

# **Podstawy Teleinformatyki - laboratorium**

Projekt: Sprawdzanie obecności w laboratorium z  
wykorzystaniem legitymacji studenckiej



**Prowadzący : mgr inż. Przemysław Walkowiak**

## **Wykonanie Projektu :**

Sławomir Asimowicz 126854 slawomir.asimowicz@student.put.poznan.pl

Dariusz Krajewski 126855 dariusz.w.krajewski@student.put.poznan.pl

Piotr Kaszuba 126856 piotr.kaszuba@student.put.poznan.pl

# **Spis Treści**

Prowadzący : mgr inż. Przemysław Walkowiak.....	1
Wykonanie Projektu :.....	1
Temat projektu.....	3
Podział prac.....	4
Piotr Kaszuba.....	4
Dariusz Krajewski.....	4
Sławomir Asimowicz.....	4
Funkcjonalności zaimplementowanych aplikacji.....	5
Aplikacja prowadzącego.....	5
Prowadzący ma podgląd przedmiotów, do których jest przypisany. Widok umożliwia podejrzenie frekwencji studentów na jego zajęciach. Ponieważ lista obecności jest dokumentem wiążącym prowadzący ma jedynie podgląd aktywności. Nie może edytować wpisów odnośnie obecności. Prowadzący może sprawdzać obecność wybierając zajęcia po ich temacie. Umożliwia to w prosty sposób podglądarki żądanych wydarzeń i zajęć.....	9
Aplikacja stanowiska roboczego(studenta).....	10
Baza danych.....	12
Serwer MySQL.....	13
Wybrane technologie.....	14
Front-end prowadzącego.....	14
Front-end stacji roboczych(studentów).....	15
Back-end.....	16
Architektura rozwiązania.....	17
Instrukcja użytkowania.....	18
Instalacja systemu.....	18
Instalacja bazy danych.....	18
Aplikacja Prowadzącego.....	20
Aplikacja stanowiska roboczego.....	21
Interesujące problemy i rozwiązania.....	21
Uruchamianie aplikacji w tle.....	21
Komunikacja z kartą.....	21
Efekty ubocznie powstawania projektu.....	23
Perspektywy rozwoju:.....	24
Przebieg pracy.....	25
Podsumowanie.....	27
Bibliografia.....	28

## Temat projektu

Tematem tego projektu jest aplikacja umożliwiająca sprawdzanie obecności w laboratorium z wykorzystaniem legitymacji studenckiej. Obecnie logowanie obecności odbywa się w tradycyjny sposób, przy pomocy kartki, długopisu i listy ocenności. By usprawnić ten proces można wykorzystać dostępne każdemu studentowi zasoby: stanowisko komputerowe oraz legitymację studencką. Legitymacja jest wyposażona w chip umożliwiający odczytanie informacji wystarczających do zweryfikowania tożsamości studenta. Projekt zakłada stworzenie oprogramowania dedykowanego czytaniu danych z kart inteligentnych, rejestrówaniu obecności studentów na zajęciach i przeglądaniu stanów frekwencji poprzez aplikację sprzągniętą z bazą danych.

Wybranie tego tematu umożliwiło zapoznanie się z technologiami wykorzystywany na całym świecie od kart dostępu aż po systemy bankowości. Technologie tak zwanych *smart cards*. Dodatkowo poszerzyły umiejętności z dziedziny definiowania i programowania baz danych a także projektowania interfejsów użytkownika(ang. GUI) z użyciem szeroko stosowanych technologii platformy .NET. Dodatkowym powodem wyboru tego tematu było zainteresowanie programowaniem niskopoziomowym, wymaganym do wykorzystania możliwości technologii kart elektronicznych *smart cards*.



Ilustracja 1: Przykładowy czytnik kart inteligentnych na interfejsie USB

Czytniki kart inteligentnych mogą być zintegrowane z obudową komputera, jak ma to często miejsce w przypadku laptopów oraz podłączane po interfejsie USB. Taki czytnik wymaga własnego sterownika oraz programu, który będzie czytał odpowiednio dane z karty.

## **Podział prac**

Prace zostały rozdzielone między członków zespołu projektowego według zainteresowań oraz wcześniejszego doświadczenia. Każdy miał swoje zadanie do zrealizowania między kolejnymi spotkaniami, na których prezentowane były postępy i integrowane z już zaimplementowanymi funkcjonalnościami. Projekt nie powstawał zgodnie z żadną z metodologii zwinnych(ang. agile) lecz w ramach wspólnie ustalanych terminów oraz celu implementacyjnych.

### **Piotr Kaszuba**

Lider projektu - wyznaczanie zadań, wykorzystywanych technologii, ustalanie terminów spotkań i celów długofałcowych i krótkoterminowych.

Programowanie graficznego interfejsu użytkownika - Język C# z wykorzystaniem *WPF*(Windows Presentation Foundation) platformy .NET.

Oprogramowanie funkcjonalności przechwytywania informacji z bazy danych w aplikacji prowadzącego z użyciem technologii LINQ oraz C#.

Uruchamianie aplikacji stanowiska roboczego w tle - Wykorzystanie skompilowanego kodu w języku *Python* jako aplikacji działającej w tle i oczekującej na sygnały z kart inteligentnych przy użyciu programu *bat2exe*.

### **Dariusz Krajewski**

Definicja bazy danych do przechowywania informacji o studentach - Zaprojektowanie bazy danych oraz zaimplementowanie bazy z użyciem *MariaDB* (fork *MySQL*). Baza implementowana była na systemie operacyjnym GNU/Linux i przeniesiona w celach prezentacji na system Windows 7.

Grafika - Opracowanie wyglądu aplikacji po stronie prowadzącego, dobór grafik i odpowiednich okien dialogowych oraz ich komunikatów.

Komunikacja aplikacji czytającej z bazą danych - Kod w języku *Python* z wykorzystaniem biblioteki *MySQLdb* umożliwiający wywoływanie odpowiednich zapytań SQL w bazie danych odpowiedzialnych za logowanie obecności.

### **Sławomir Asimowicz**

Oprogramowanie czytające informacje z legitymacji studenckiej - Implementacja programu działającego w pętli i oczekującego na sygnał wsunięcia karty do czytnika, po czym zczytanie odpowiednich danych i wyświetlenie komunikatu o pomyślnym zarejestrowaniu obecności,

Testowanie aplikacji - Wyszukiwanie błędów w kodzie, wykorzystanie *debuggera* do znajdowania nieobsłużonych wyjątków. Próba wywoływania błędów aplikacji i dokumentacja natury pojawiających się problemów.

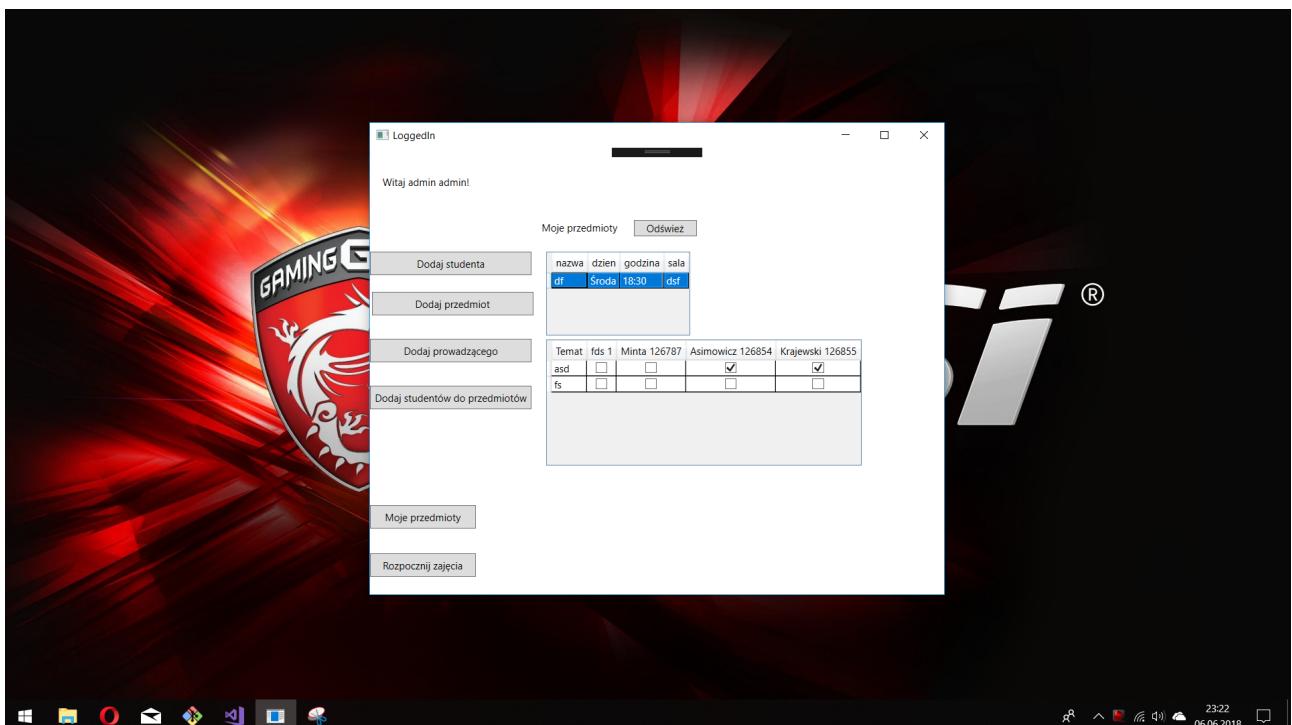
Usuwanie błędów aplikacji stanowiska roboczego- Poprawa kodu aplikacji stanowiska roboczego celem wyeliminowania wszystkich znalezionych problemów i obsługi wyjątków.

# Funkcjonalności zaimplementowanych aplikacji

Ze względu na potrzebę wykonywania różnych operacji po stronie prowadzącego i studenta projekt składa się z dwóch odrębnych aplikacji. Obie aplikacje są skomunikowane z jedną bazą danych zarządzającą danymi wszystkich studentów oraz wszystkich prowadzących. Przyjęcie takiego rozwiązania podyktowane jest tym, że jedna baza ułatwia komunikację aplikacji roboczych oraz prowadzących. Nie będzie potrzeby synchronizowania stanów baz danych. Co więcej przechowywane dane nie zajmują sporo miejsca.

## Aplikacja prowadzącego

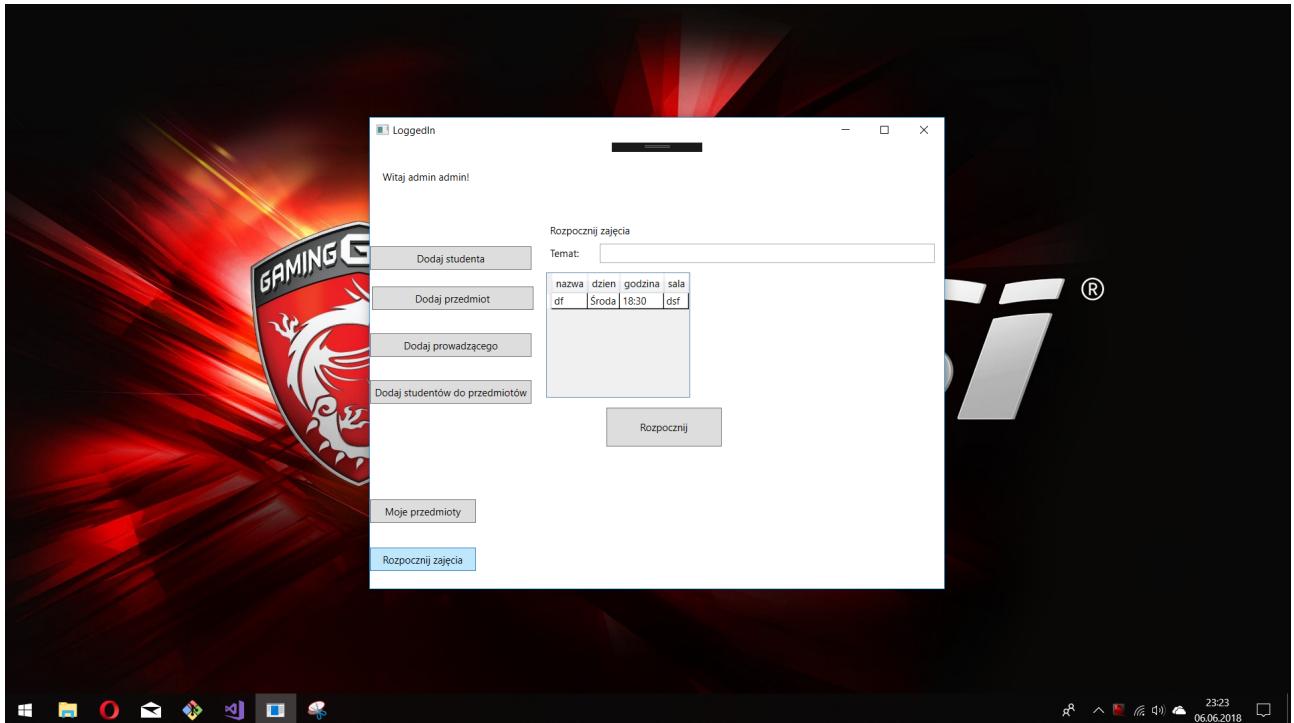
Od strony prowadzącego zajęcia aplikacja umożliwia w prosty i czytelny sposób podgląd obecności studentów na zajęciach.



Ilustracja 2: Widok obecnych na zajęciach

W oknie podglądu obecności widoczny jest temat obecnych zajęć oraz tych, które już się wcześniej odbyły. Obecność studentów jest wyświetlana również dla wszystkich poprzednich zajęć. Jest to domyślny widok dla każdego prowadzącego na zajęciach. Wyświetlane są informacje odnośnie rodzaju i tematu zajęć, godziny rozpoczęcia oraz sali, w której zajęcia się odbywają.

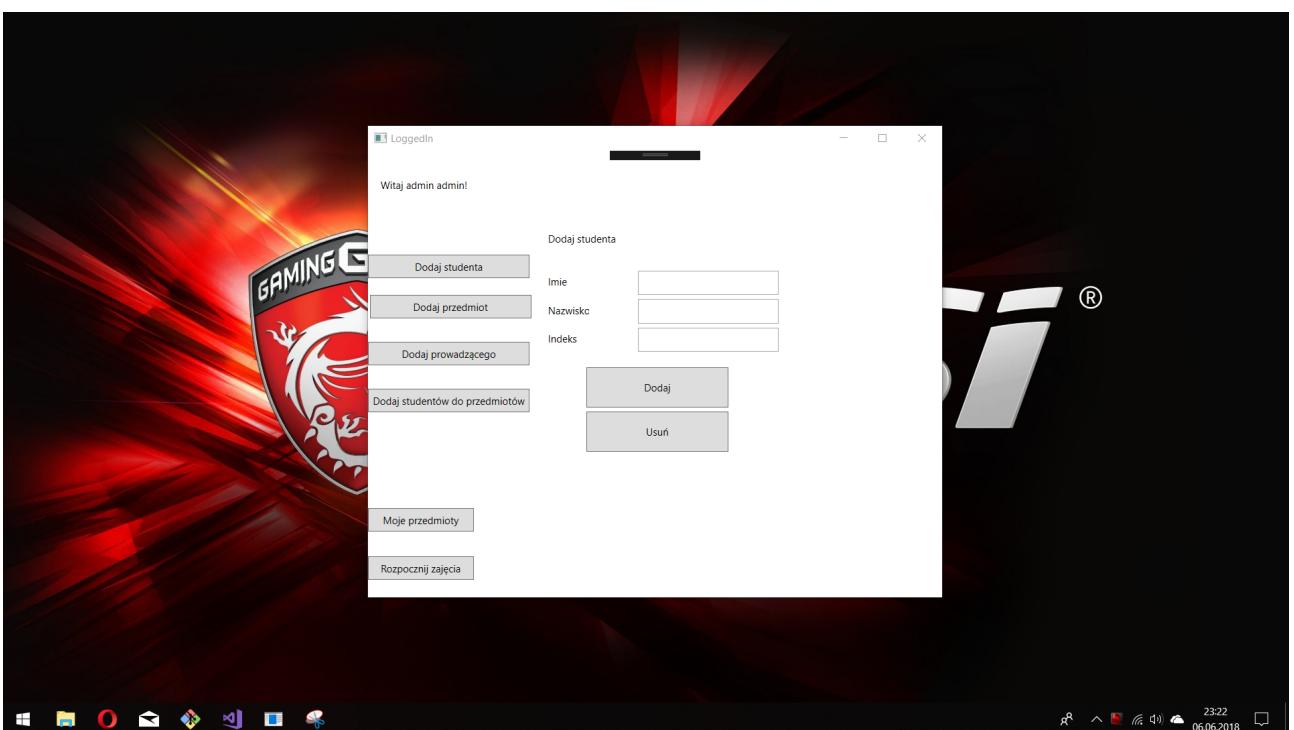
Prowadzący może wprowadzać nowe zajęcia do systemu z dedykowanego widoku.



Ilustracja 3: Widok rozpoczynania zajęć

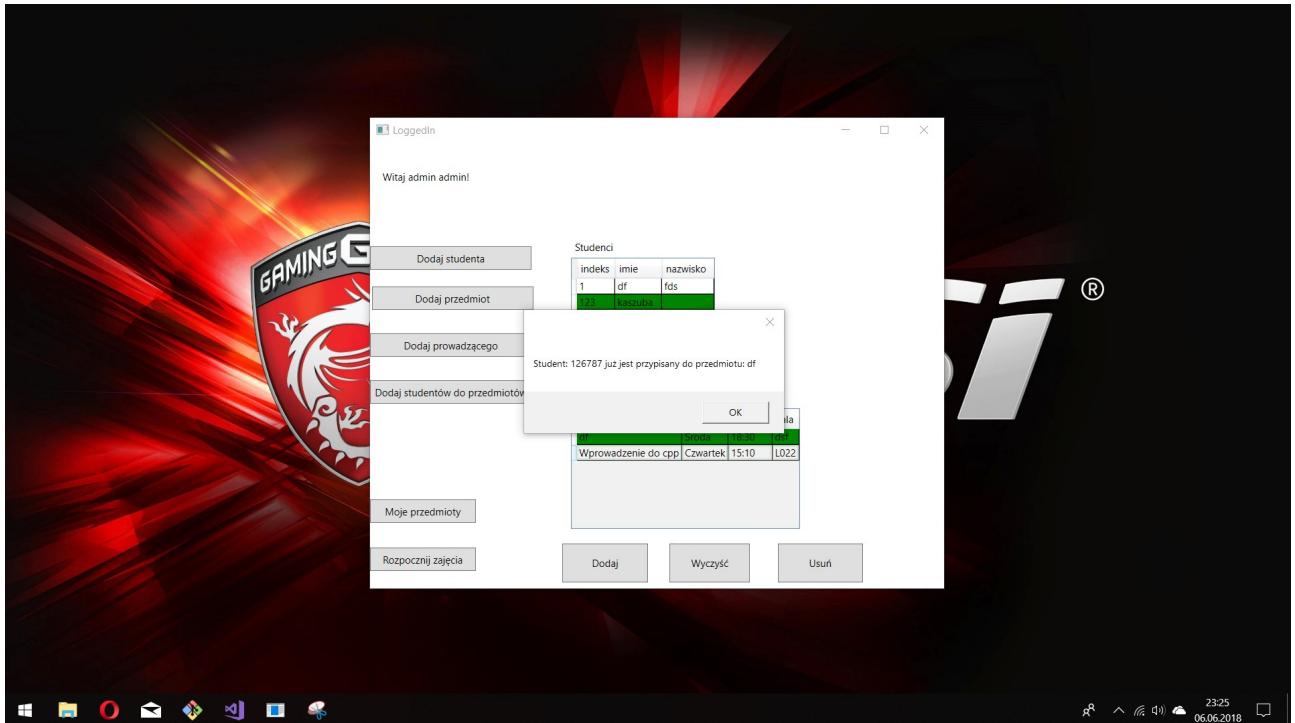
Wybór dostępnych do rozpoczęcia zajęć przez prowadzącego wyświetlany jest w postaci tabelki. Każde zajęcia można opisać tematem. Prowadzący w ten sposób nie może prowadzić zajęć, które nie są mu przypisane. Dodatkowo zawsze może zobaczyć, które z zajęć są dla niego dostępne. Wystarczy załadować predefiniowane ustawienia zamiast "wyklikawać" kolejne pozycje i zatwierdzić rozpoczęcie zajęć. Po zakończeniu tej operacji studenci mogą się zacząć rejestrować.

Prowadzący ma również możliwość dodawania nowych studentów do zajęć. Będzie to wymagane przy pierwszych zajęciach by zdefiniować kto powinien się pojawiać.



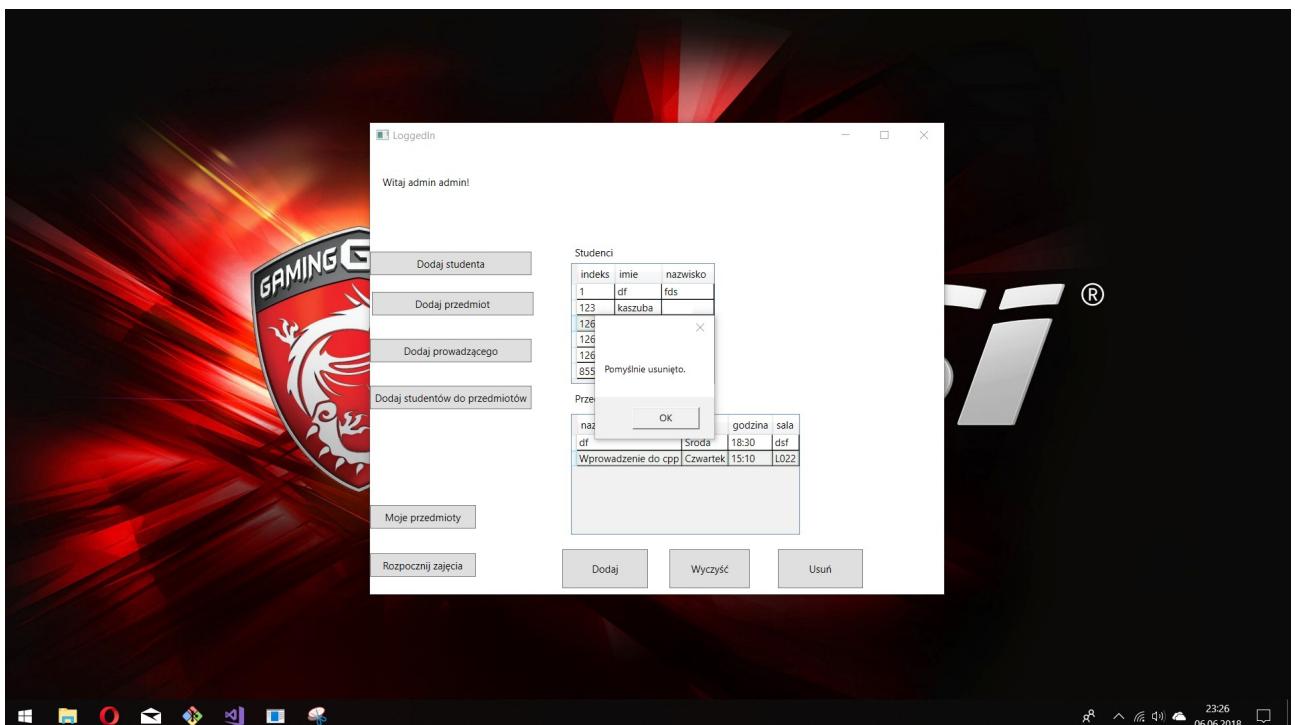
Ilustracja 4: Widok dodawania studenta

Dodawanie studentów, którzy powinni brać udział w zajęciach wymaga podania ich imion, nazwisk oraz indeksów. Raz zdefiniowana lista pozwala na dalsze korzystanie z aplikacji bez konieczności definiowania odrębnych grup osób dla kolejnych zajęć.



Ilustracja 5: Widok przypisywania studenta do przedmiotu

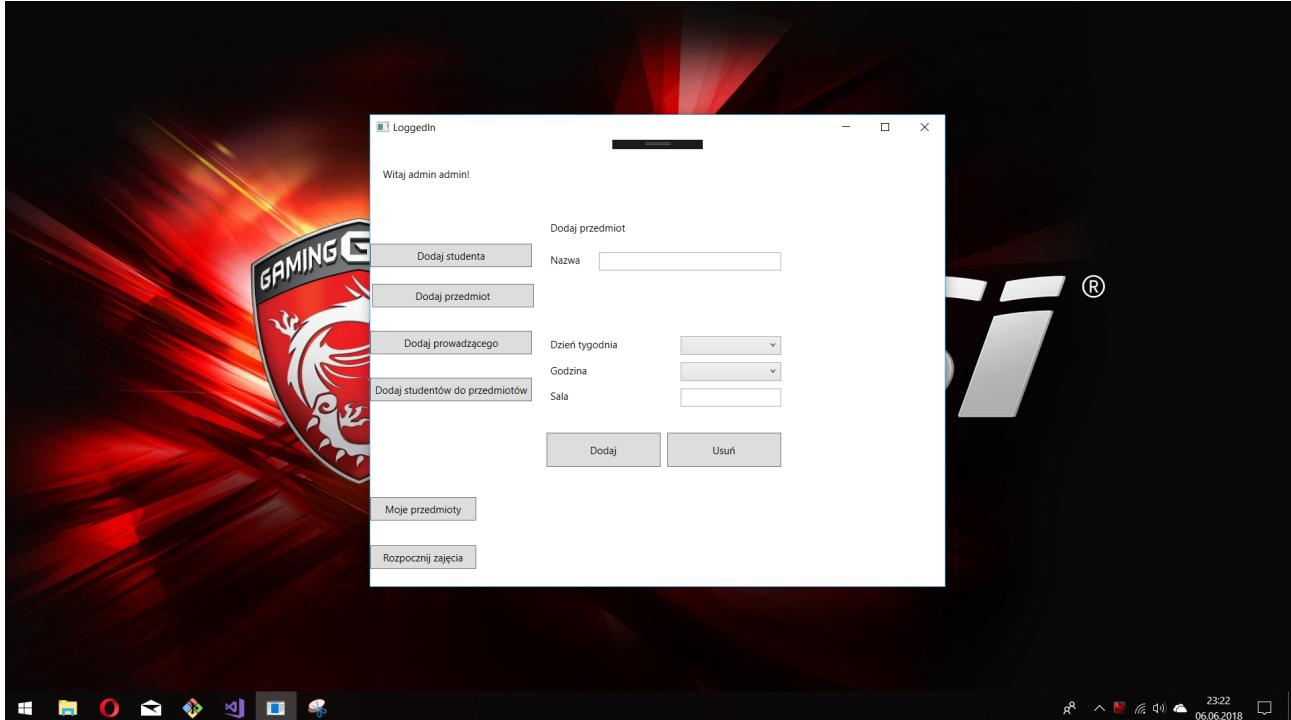
Dodawanie studentów do zajęć odbywa się przez wybranie z listy wszystkich studentów tych, którzy powinni na dane zajęcia uczęszczać. Okno dialogowe informuje prowadzącego o statusie operacji.



Ilustracja 6: Komunikat o usunięciu studenta z przedmiotu

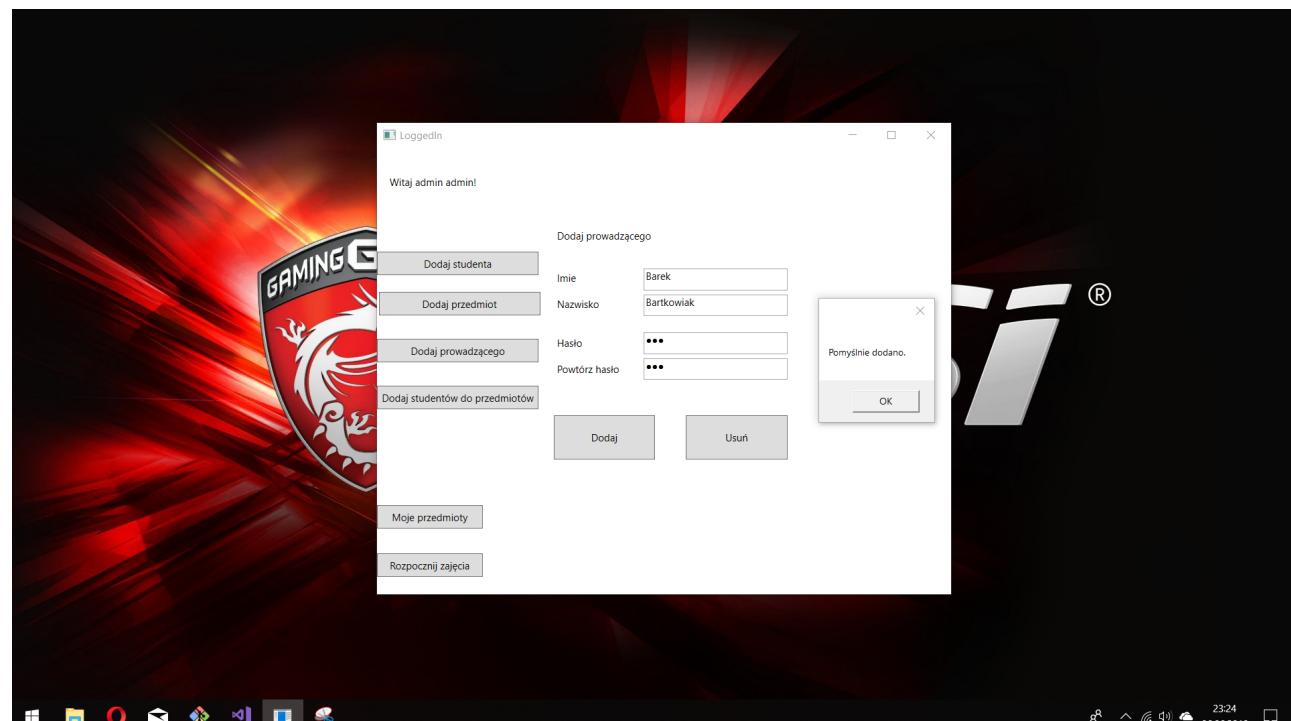
Jeśli przez pomyłkę dodano nieodpowiednią osobę zawsze można tą operację odwrócić.

Administrator może wprowadzać do bazy danych kolejne typy zajęć, które później prowadzący będą mogli zdefiniować jako odbywające się. Jeśli jakieś zajęcia przepadają po prostu nie są tworzone w bazie danych. Rejestracja jedynie odbywających się zajęć.



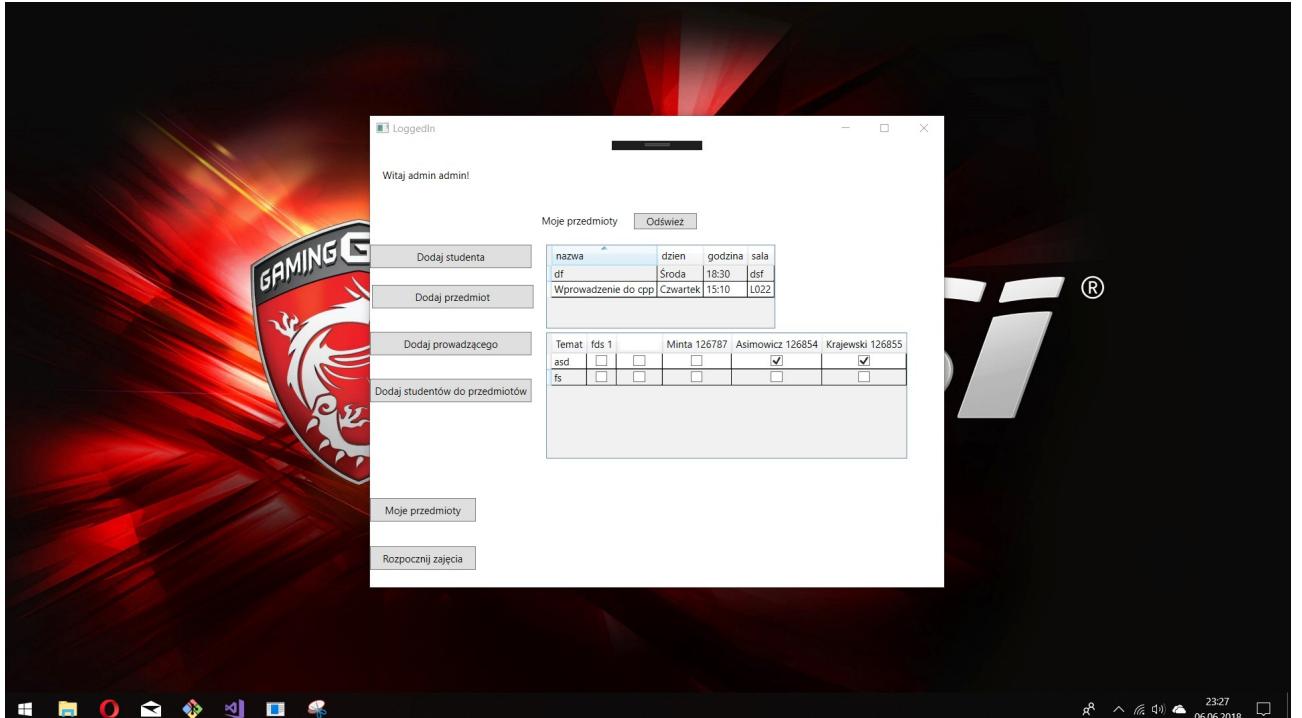
Ilustracja 7: Widok dodawania przedmiotów

Z poziomu tego widoku administrator może dodać typ zajęć jaki będzie dostępny do prowadzenia później w trakcie semestru. Administrator podaje nazwę, dzień tygodnia, w jaki zajęcia powinny być prowadzone, godzinę rozpoczęcia, gdyż system zakłada, że zajęcia trwają 1,5h oraz salę, w której zajęcia się będą odbywać.



Ilustracja 8: Widok dodawania prowadzącego

Każde zajęcia muszą być przez kogoś prowadzone. Projekt zakłada, że to prowadzący rozpoczyna zajęcia i z domysłu zajęcia nie zarejestrowane w systemie to takie, które się nie odbyły. Dodawanie prowadzącego to mechanizm podobny do tworzenia konta w każdej innej usłudze. Należy ustawić dane logowania, czyli imię i nazwisko prowadzącego oraz hasło prowadzącego. Każdy prowadzący powinien zmienić domyślnie założone przez administratora hasło przy pierwszym logowaniu.



Ilustracja 9: Podgląd prowadzonych zajęć i studentów przypisanych do zajęć

Prowadzący ma podgląd przedmiotów, do których jest przypisany. Widok umożliwia podejrzenie frekwencji studentów na jego zajęciach. Ponieważ lista obecności jest dokumentem wiążącym prowadzący ma jedynie podgląd aktywności. Nie może edytować wpisów odnośnie obecności. Prowadzący może sprawdzać obecność wybierając zajęcia po ich temacie. Umożliwia to w prosty sposób podglądarki żądanych wydarzeń i zajęć.

## Aplikacja stanowiska roboczego(studenta)

Od strony studentów aplikacja jest niewidoczna. Działa w tle nasłuchując na zeskanowanie legitymacji.

W momencie zarejestrowania obecności użytkownika jest informowany odpowiednim oknem dialogowym. Okno wyświetla odpowiedni komunikat w zależności od odbywających się zajęć i powiązania studenta z nimi. Przykładowo student nie może się zarejestrować na zajęcia, które nie odbywają się dla jego grupy.

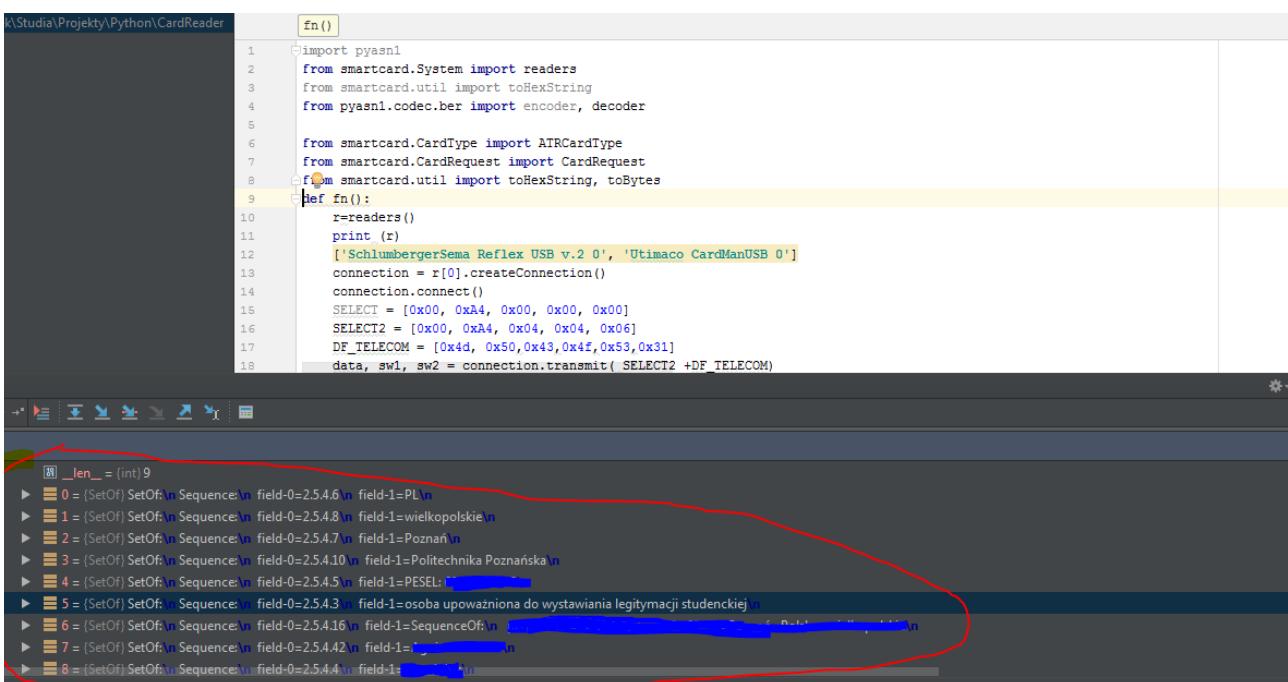
Czytanie z kart inteligentnych odbywa się z użyciem APDU(Application Protocol Data Unit) czyli jednostek komunikacji między czytnikiem kart inteligentnych a tymi właśnie kartami. Wysyłane polecenia mają charakter ciągów bajtów.

APDU commands are byte arrays containing the following:

APDU Command						
Header				Body		
CLA	INS	P1	P2	Lc	Data	Le

Ilustracja 10: Budowa APDU

Jednostka składa się z dwóch części. Nagłówka i ciała. Pierwszy bajt CLA określa dziedzinę rozkazu wysyłaną do przez czytnik do karty. Bajt INS określa dokładnie jakie polecenie z uprzednio zadeklarowanej dziedziny należy wykonać, na przykład "odczytaj z pliku". Pole P1 i P2 to dwa bajty będące argumentami wywołanego polecenia. W ciele jednostki pierwszy bajt określa liczbę wysyłanych bajtów danych. DATA to dane przesyłane do karty. LE to bajt określający ile bajtów danych oczekuje się w odpowiedzi.



```
\\"\\Studio\\Projekty\\Python\\CardReader
fn():
    import pyasn1
    from smartcard.System import readers
    from smartcard.util import toHexString
    from pyasn1.codec.ber import encoder, decoder

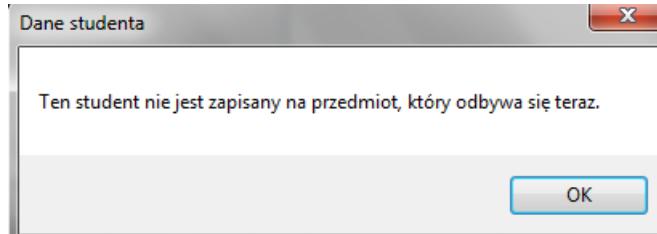
    from smartcard.CardType import AIRTCardType
    from smartcard.CardRequest import CardRequest
    from smartcard.util import toHexString, toBytes

    def fn():
        r=readers()
        print (r)
        ['SchlumbergerSema Reflex USB v.2.0', 'Utimaco CardManUSB 0']
        connection = r[0].createConnection()
        connection.connect()
        SELECT = [0x00, 0xA4, 0x00, 0x00, 0x00]
        SELECT2 = [0x00, 0x44, 0x04, 0x04, 0x06]
        DF_TELECOM = [0x4d, 0x50, 0x43, 0x4f, 0x53, 0x31]
        data, sw1, sw2 = connection.transmit( SELECT2 +DF_TELECOM)

    _len_ = [int] 9
    0 = {SetOf: Sequence field-0=2.5.4.6 field-1=P1
    1 = {SetOf: Sequence field-0=2.5.4.8 field-1=wielkopolskie
    2 = {SetOf: Sequence field-0=2.5.4.7 field-1=Poznan
    3 = {SetOf: Sequence field-0=2.5.4.10 field-1=Politechnika Poznanska
    4 = {SetOf: Sequence field-0=2.5.4.5 field-1=PESEL
    5 = {SetOf: Sequence field-0=2.5.4.3 field-1=osoba upoważniona do wystawiania legitymacji studenckiej
    6 = {SetOf: Sequence field-0=2.5.4.16 field-1=SequenceOf
    7 = {SetOf: Sequence field-0=2.5.4.42 field-1=Jednostka
    8 = {SetOf: Sequence field-0=2.5.4.4 field-1=Jednostka
```

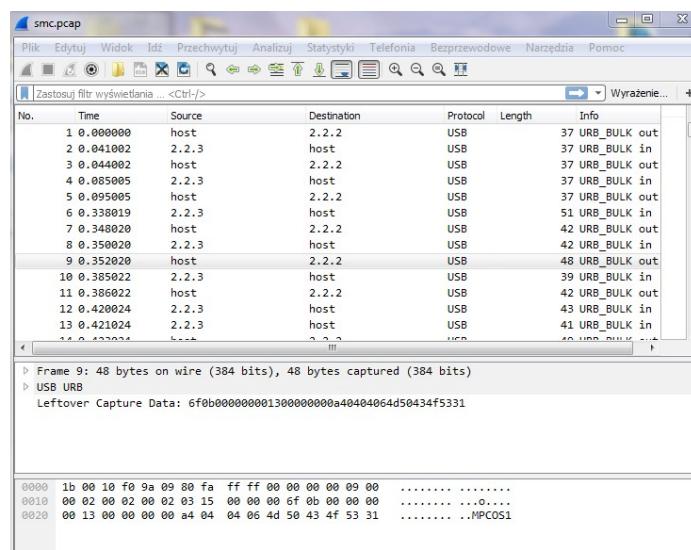
Ilustracja 11: Przykładowe zapytanie i odpowiedź od karty inteligentnej

Wywoływanie w kodzie zapytań do karty inteligentnej odbywa się z użyciem biblioteki *pyscard*. Umożliwia ona na wykrywanie wsunięcia karty do komputera lub odpowiedniego czeytnika na gnieździe USB. Widocza na zrzucie ekranu funkcja *fn* posiada deklaracje tablic SELECT oraz SELECT2. Są to zapytania kierowane do karty. Wysyłanie odbywa się z użyciem funkcji *connection.transmit()*. W odpowiedzi, w dolnej części ekranu wyświetlane są dane przesłane przez kartę.



*Ilustracja 12: Przykładowy komunikat po stronie aplikacji stanowiska roboczego*

Powyższa ilustracja przedstawia przypadek, gdy student nie powinien być obecny. Podobne okienka wyskakują w momencie pomyślnego logowania obecności na zajęciach, na których student powinien się zarejestrować. Inną możliwością było wykorzystanie "dymków" systemowych do informowania studenta o odnotowaniu jego obecności. Jednak z uwagi na subtelność tej metody student mógłby przegapić i zapomnieć taki monit, po czym i tak trzeba by poinformować studenta, że już jest zalogowany. Zrezygnowano zatem z "dymków informacyjnych" na rzecz okienka wymagającego interakcji.

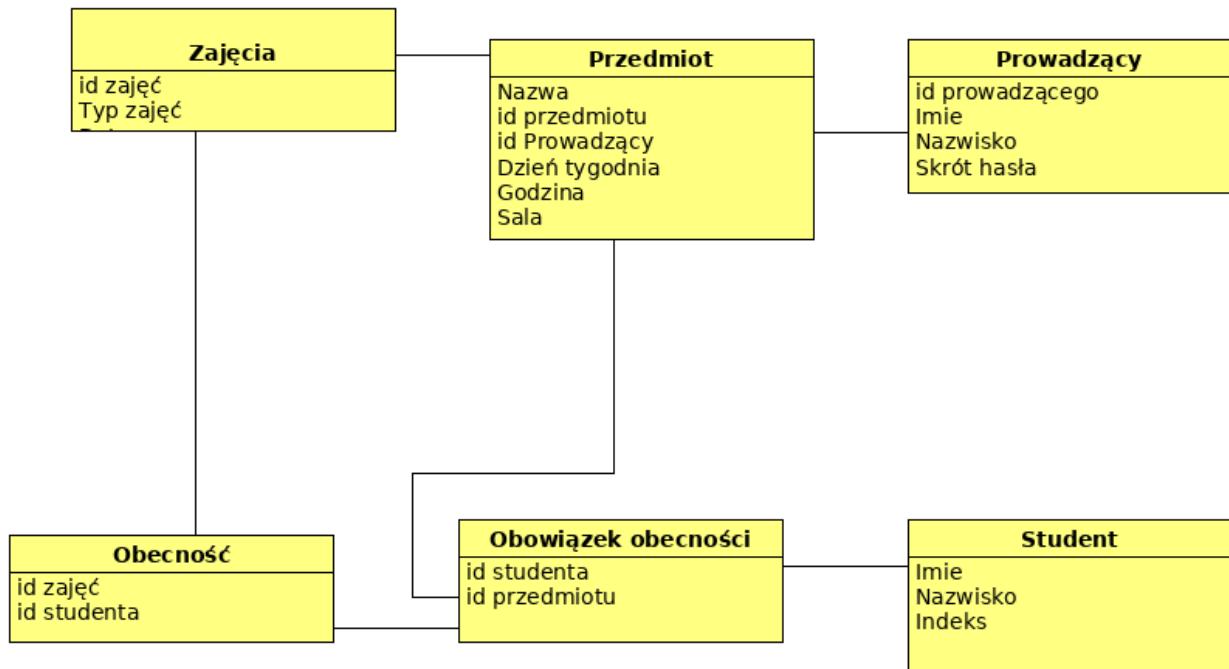


*Ilustracja 13: Widok komunikacji czytnika z aplikacją stanowiska roboczego*

Cała ineligenca aplikacji skupia się na przesyłaniu danych w tle i automatyzacji procesu sprawdzania obecności. Z tego wynika jedynie konieczność informowania użytkownika o działaniu programu i odnotowaniu interakcji. Ponieważ interfejsu graficznego jako takiego nie ma użytkownik musi chociaż otrzymać informację zwrotną o funkcjonowaniu programu.

## Baza danych

Baza danych jest elementem niezbędnym do funkcjonowania aplikacji prowadzącego oraz stacji roboczej. Baza ma charakter scentralizowany, jedna baza obsługuje wszystkie obecności, typy zajęć, wszystkich prowadzących.



Ilustracja 14: Diagram bazy danych na potrzeby projektu

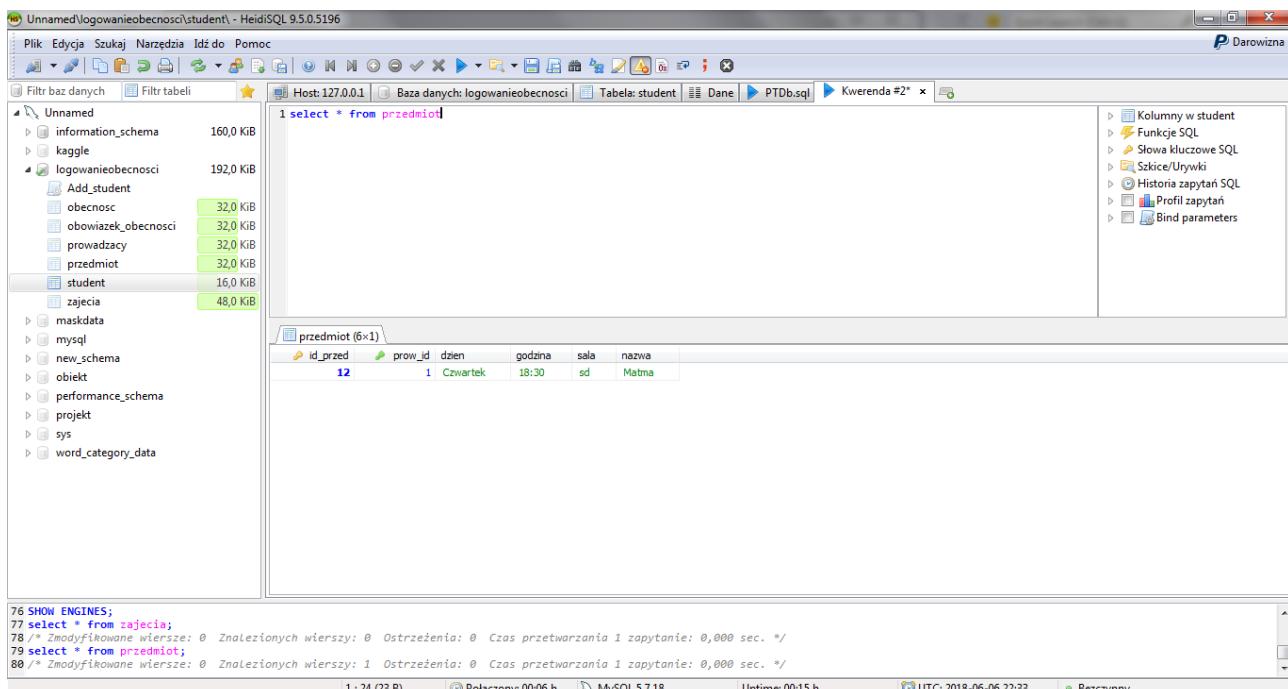
Aplikacje stacji roboczych oraz prowadzącego kontaktują się z bazą odpowiednio aby zarejestrować obecność na zajęciach i przeglądać obecny stan frekwencji.

Dodatkowo w bazie przechowywane są informacje o wszystkich prowadzących i możliwych typach zajęć.

## Serwer MySQL

W każdej chwili administrator ma dostęp do bazy danych działającej na potrzeby systemu.

Ponieważ prowadzący nie może zarejestrować prowadzenia zajęć bez zalogowania się ważnym jest aby baza danych działała nieprzerwanie. Ryzyko przepelnienia bazy danych zapytaniami jest bardzo niewielkie. Ilość wysyłanych i zbieranych danych jest bardzo mała. Podczas logowania wysyłane są jedynie proste polecenia do bazy danych celem odnotowania obecności. Prowadzący również wysyła proste zapytanie do bazy podczas rejestrowania odbywających się zajęć i pobierania lub aktualizacji obecnego stanu obecności. W aplikacji odświeżanie jest na żądanie a nie stałe gdyż nie można ocenić jak często trzeba odpytywać bazę danych by odnotować studentów, którzy przyszli spóźnieni.



Ilustracja 15: Widok bazy danych w HeidiSQL

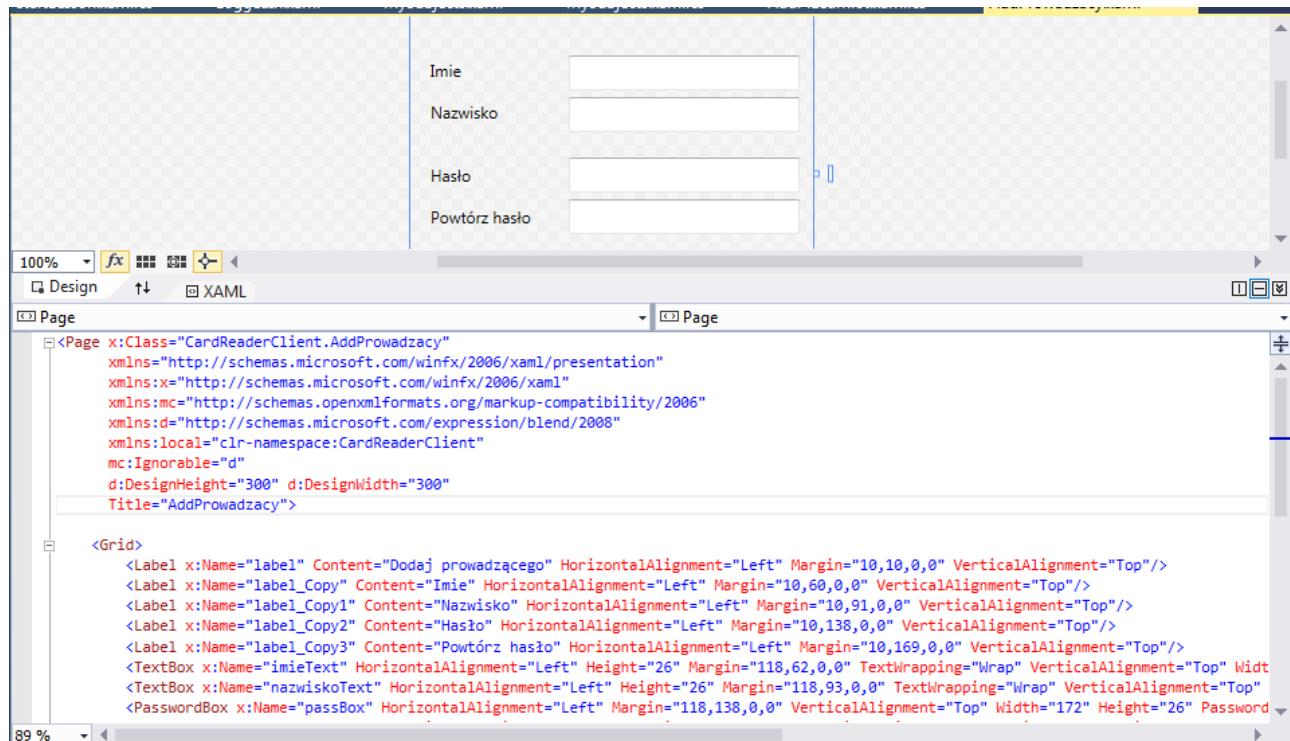
Do funkcjonowania aplikacji wymagany jest działający serwer MySQL. Baza może działać w ramach jednej sieci w stanowiskami prowadzącego i stanowiskami roboczymi lub w osobnej sieci. Funkcjonalności MySQL są wystarczające do prowadzenia operacji, których wymaga implementacja. Dodatkowo MySQL tworzony był z myślą o szybkości co dodatkowo przemawia na korzyść tego rozwiązania.

# Wybrane technologie

Pod względem technologii wykonania projekt jest bardzo zróżnicowany. Do każdego realizowanego zadania wykorzystano różny język i metody wykonania. Podział całego projektu na dedykowane do zadania aplikacje umożliwiło dobranie odpowiednich technologii do każdego z nich pod kątem skomplikowania. Wiodące w projekcie technologie to języki C# oraz Python. Framework .NET z wykorzystaniem WPF. Baza danych oparta o MySQL, rozwijana przez społeczność oraz deweloperów MariaDB.

## Front-end prowadzącego

Część dostępna dla prowadzącego jest napisana w języku C# z użyciem technologii LINQ do obsługi zapytań i przetwarzania odpowiedzi bazy danych oraz WPF do zaprojektowania interfejsu użytkownika. WPF to nazwa silnika graficznego i API bazującego na .NET 3, wchodzącego w skład WinFX. WPF integruje interfejs użytkownika, grafikę 2D i 3D, multimedia, dokumenty (nazwa kodowa Metro) oraz generowanie/rozpoznawanie mowy (do aplikacji sterowanych głosem).

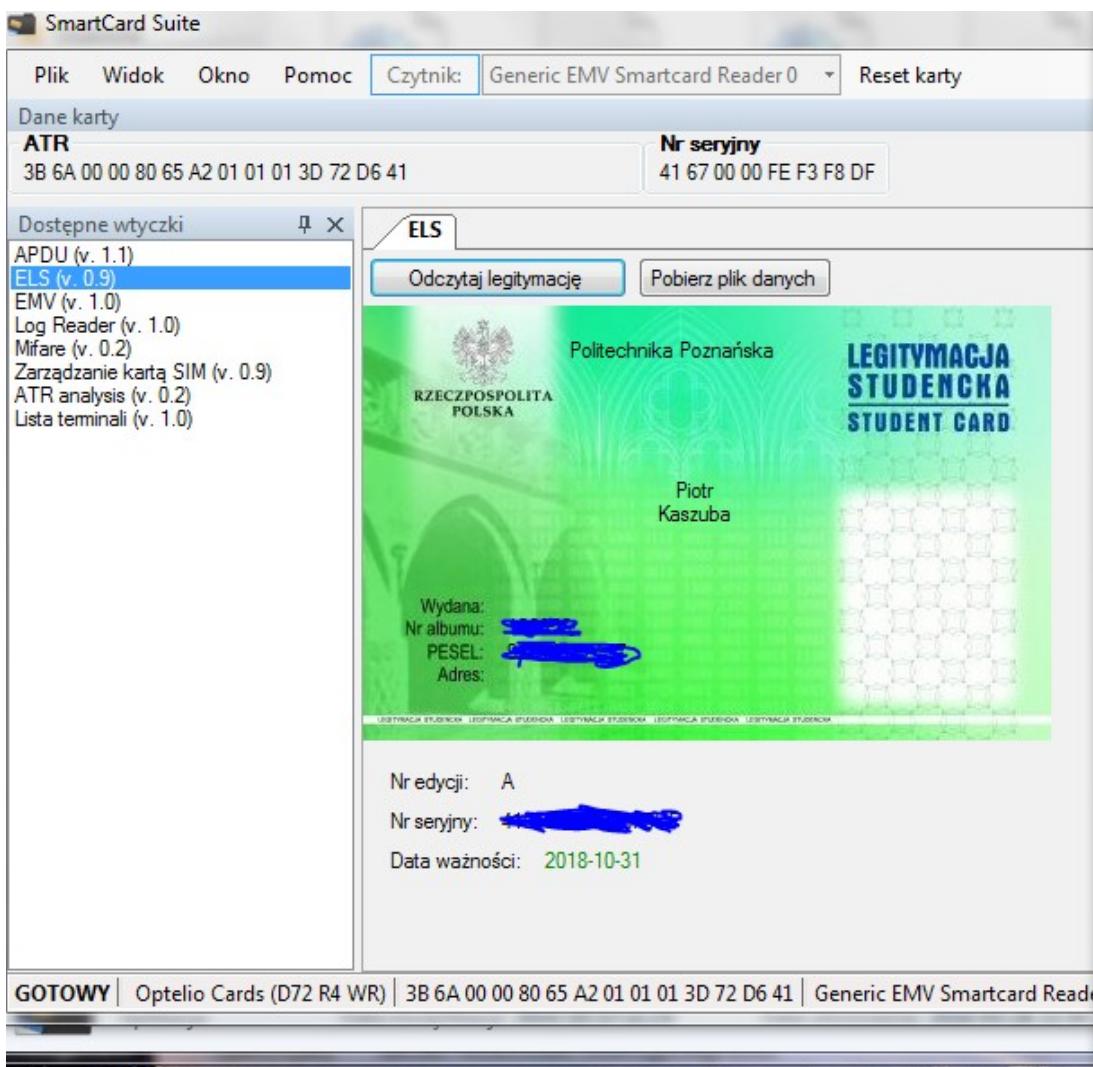


Ilustracja 16: Definiowanie grafiki w WPF przy pomocy XML

API w WPF opiera się na języku XML, dokładniej na jego implementacji o nazwie XAML. Całość jest zawarta w nowym API WinFX, zaś graficzna część GUI wykorzystuje grafikę wektorową, budowaną z użyciem akceleratorów grafiki 3D i efektów graficznych udostępnianych przez WGF. Rozwiązanie to jest podobne do Quartz z Mac OS X.

## Front-end stacji roboczych(studentów)

Aplikacja do czytania danych z legitymacji jest napisana w języku Python ze względu na prostotę składni oraz dostępność bibliotek do obsługi komunikacji z czytnikami kart elektronicznych *smart cards* oraz komunikacji z bazą danych MySQL. Ponieważ sama aplikacja nie zawiera wielu linii kodu i jest nieskomplikowana nie było potrzeby wykorzystywania języków z rozszerzoną funkcjonalnością, takich jak C, C++.



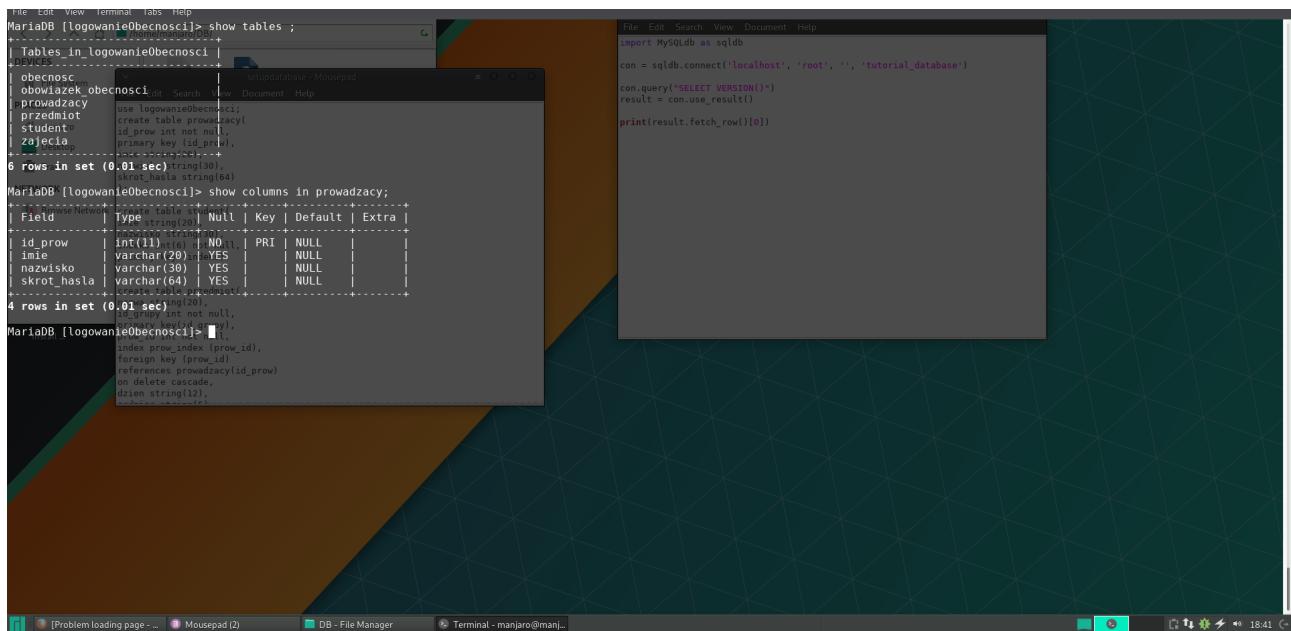
Ilustracja 17: Przykładowa aplikacja do czytania danych z kart inteligentnych

Istnieje wiele aplikacji do czytania danych z kart inteligentnych, na których można oprzeć aplikację po stronie stanowiska roboczego. Nie jest jednak potrzebne za każdym razem wyświetlanie informacji dla studenta o jego legitymacji. Program działa w sposób podobny do sterownika. Czeka w tle na wywołanie odpowiedniego zdarzenia, po którym wysyła zapytanie do czytnika kart a później do bazy danych.

## Back-end

Baza danych oparta o technologię MySQL. MySQL jest szybkim silnikiem baz danych i posiada wszystkie niezbędne funkcjonalności do prowadzenia ewidencji obecności na zajęciach.

Dodatkowo jest kompatybilna z framework'iem LINQ dostępnym w języku C# oraz dostępność bibliotek dla języka Python jest bardzo wysoka. Implementacja bazy danych nastąpiła przy użyciu mariaDB.

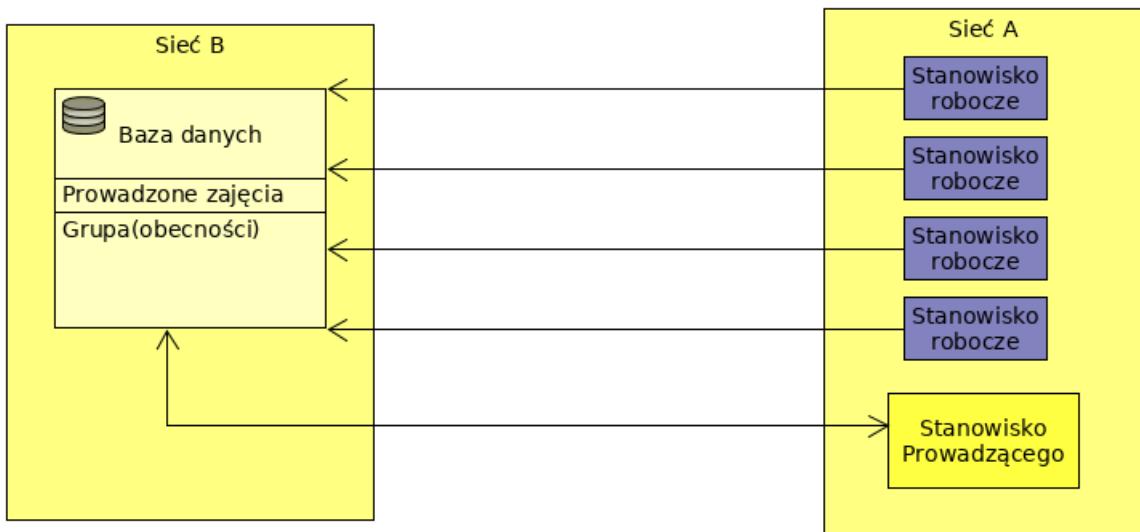


Ilustracja 18: Konsola bash z widokiem bazy danych MariaDB

MariaDB jest rozwiązaniem *open-source* z szerokim i dobrym wsparciem ze strony deweloperów oraz społeczności. Umożliwia tworzenie stabilnych baz danych z funkcjonalnościami MySQL. Implementowanie bazy danych w ten sposób umożliwia na późniejsze uruchamianie bazy danych na komputerach z systemem zarówno Windows jak i *GNU/Linux*.

## Architektura rozwiązania

Do poprawnego działania system wymaga trzech elementów. Pierwszym jest baza danych zbierająca dane o obecności studentów. Baza ma charakter zcentralizowany, jedna baza danych zarządza rejestraniem obecności na wszystkich zajęciach. Z bazą komunikują się klienci prowadzący. Jedno stanowisko prowadzącego na których zajęciach. Na każdym stanowisku komputerowym dostępnym dla studenta działa oprogramowanie oczekujące na wczytanie legitymacji. Po zczytaniu i wysłaniu informacji do bazy prowadzący może na bieżąco monitorować stan obecności na swoich zajęciach.



*Ilustracja 19: Przykładowa konfiguracja architektury systemu*

Na powyższym przykładzie zakładamy, że komputer prowadzącego oraz stanowiska robocze znajdują się w jednej sieci(np. w jednej sali obsługiwanej przez jeden ruter) podczas gdy baza danych działa na serwerze w innym budynku. Stanowiska robocze nie muszą kontaktować się z stanowiskiem prowadzącego, gdyż logują wszystkie obecności bezpośrednio w serwerze. W ten sposób pomija się dodatkowe kroki sprawdzania listy gdyż celem projektu jest automatyzacja i przyspieszenie procesu sprawdzania obecności.

# Instrukcja użytkowania

Projekt był pisany z myślą o prostym wdrożeniu do korzystania w ramach sieci. By rozpocząć pracę z systemem zaimplementowanym w ramach tego projektu należy zainstalować serwer bazy danych MySQL oraz aplikacje prowadzącego i aplikacje stacji roboczych. Do prawidłowego funkcjonowania system wymaga działania jednej bazy danych oraz dowolnej liczby aplikacji prowadzącego i stacji roboczych. Poniższa instrukcja przedstawia kroki niezbędne do wdrożenia systemu i rozpoczęcia pracy.

## Instalacja systemu

Poniższa instrukcja przeprowadza administratora systemu przez proces instalacji całego systemu.

### Instalacja bazy danych

By system działał należy rozpocząć od uruchomienia bazy danych w ramach danej sieci. Jest ona niezbędna do uruchomienia aplikacji stacji roboczych oraz prowadzącego.

Poniżej przedstawione są kroki do uruchomienia bazy danych w ramach systemu GNU/Linux.

1. W konsoli systemu GNU/Linux należy wpisać

```
$: mysql --version
```

Jeśli konsola odpowie użytkownikowi nazwą oraz wersją działającego silnika baz danych MySQL można wykonać następne kroki. Jeśli nie, należy zainstalować taki silnik użyciem domyślnego dla systemu menedżera paczek (ang. package manager).

2. W działającym silniku baz danych MySQL należy utworzyć bazę danych do obsługi funkcjonalności projektu. Się zalogować do silnika baz danych:

```
$: mysql -u <nazwa użytkownika>
```

W powyższym przypadku nastąpi bezpieczne logowanie, wartość wpisywanego hasła zostanie przesłonięta. Alternatywnie można wykonać polecenie

```
$: mysql -u <nazwa użytkownika> -p<hasło>
```

Wykonanie tego polecenia zostawi w historii konsoli widoczne hasło, zaleca się wykonanie dodatkowego polecenie

```
$: history -c
```

3. Po udanym logowaniu użytkownik powinien zobaczyć domyślną wiadomość powitalną. Będąc zalogowanym można utworzyć bazę danych. Należy wprowadzić polecenie

```
> create database logowanieobecnosci;
```

Należy zwrócić uwagę, że polecenia wydawane silnikowi baz danych MySQL zakończone są średnikiem.

4. W odpowiedzi użytkownik powinien zobaczyć wiadomość potwierdzającą utworzenie bazy danych. Teraz można zakończyć pracę w silniku. Należy wprowadzić polecenie  
> quit;

5. Dodanie bazy danych projektu wymaga utworzenia jej z wykorzystaniem gotowego, dostarczonego pliku *.sql*. Z poziomu terminala(już nie silnika bazy danych) należy wprowadzić polecenie

```
$: mysql -u <nazwa użytkownika> -p logowanieobecnosci < logowanieobecnosci.sql
```

Ponownie, wywołanie takiego polecenia wywoła bepieczne logowanie z przesłoniętym hasłem. By wykonać to samo polecenie lecz wprowadzić hasło w widoczny sposób należy wprowadzić polecenie:

```
$: mysql -u <nazwa użytkownika> -p<hasło> logowanieobecnosci < logowanieobecnosci.sql
```

6. Należy pozwolić bazie danych dokończyć odtwarzanie wszystkich tabel. Proces nie powinien zająć dużo czasu.
7. Po zobaczeniu wiadomości potwierdzającej pomyślne ukończenie operacji można się upewnić, że wszystko zostało odtworzone poprawnie. Poniższe kroki są opcjonalne.
8. Należy wkomponować logowanie do silnika bazy danych *MySQL* w sposób opisany w kroku 2.
9. Będąc w widoku konsoli silnika baz danych *MySQL* należy wprowadzić polecenie:

```
> show databases;
```

10. W tekstowym podglądzie baz danych powinna być widoczna nazwa *logowanieobecnosci*.

11. By wybrać bazę danych należy wprowadzić polecenie:

```
> use logowanieobecnosci;
```

12. Silnik baz danych powinien wyświetlić informację potwierdzającą. Teraz można się upewnić, że zostały stworzone wszystkie tabele. Należy wprowadzić polecenie:

```
> show tables;
```

13. By sprawdzić, czy wszystkie tabele zostały stworzone poprawnie należy skonfrontować wszystkie tabele z schematem bazy danych. Podgląd struktury tabeli o dowolnej nazwie w ramach danej bazy danych wykonuje się poleceniem:

```
> show columns in <nazwa tabeli>;
```

14. Jeśli wszystkie tabele zgadzają się z schematem zamieszczonym jako *ilustracja 14* można rozpocząć instalację aplikacji stanowisk roboczych i prowadzących.

By rozpocząć pracę z systemem logowania obecności potrzebna jest baza danych działająca w ramach sieci, oprogramowanie prowadzącego oraz oprogramowanie działające w ramach stanowiska roboczego(studenta).

Teraz, kiedy baza danych jest dostępna w sieci należy skonfigurować jej podstawowe wartości.

## Aplikacja Prowadzącego

Aplikacja prowadzącego działa jak każdy inny klient bazy danych. Po uruchomieniu aplikacji użytkownik witany jest prośbą o podanie informacji służących zweryfikowaniu jego tożsamości. Po podaniu prawidłowych danych logowania można podglądać zajęcia, tworzyć nowe czy sprawdzać obecność studentów. Aplikacja nie wymaga instalacji. Wystarczy uruchomić plik .exe.

*Uwaga: Aplikacja do prawidłowego funkcjonowania wymaga najnowszej wersji oprogramowania Microsoft .Net Framework. Oprogramowanie to można pobrać z oficjalnej strony Microsoft'u.*

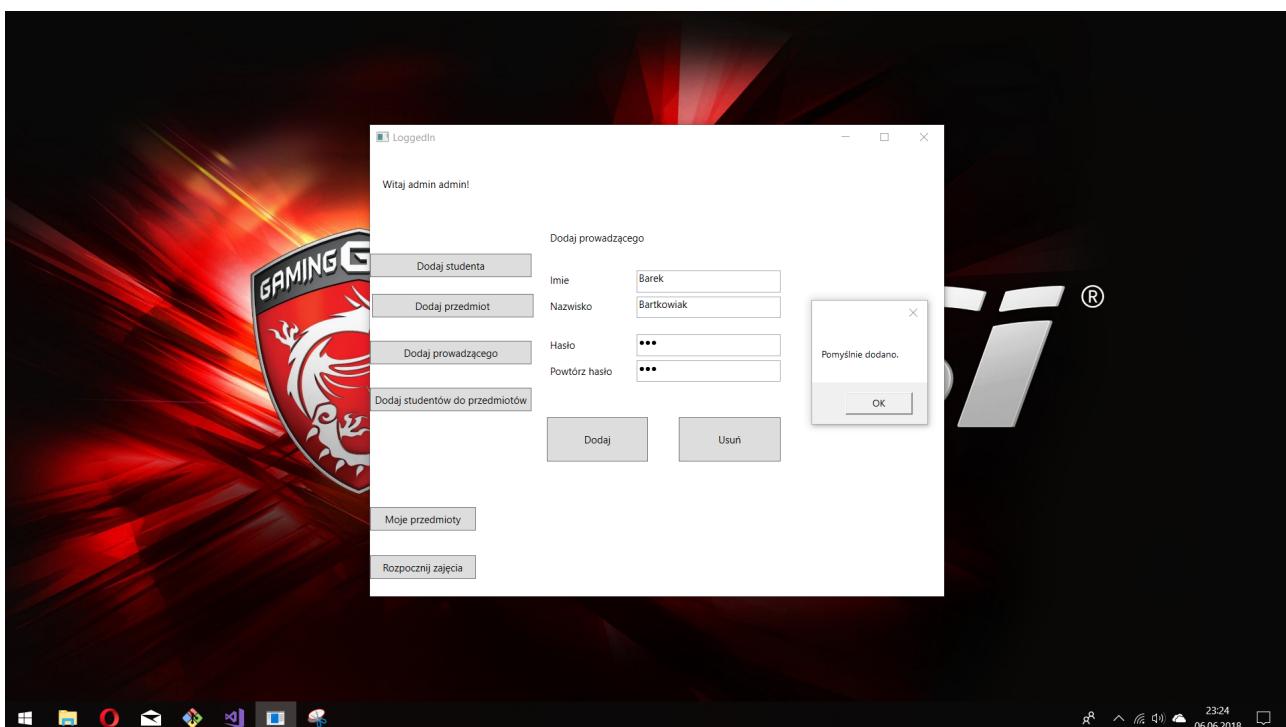
Ponieważ baza danych w pierwotnej postaci jest pusta pierwsze logowanie należy wykonać na zdefiniowanym koncie administratora.

- Imię: admin
- Nazwisko: admin
- Hasło: admin

Ze względów bezpieczeństwa zaleca się zmianę hasła administratora a nawet danych logowania po pierwszym uruchomieniu.

Teraz kiedy administrator ma dostęp do bazy danych może zacząć dodawać potrzebne informacje do działania systemu. Pierwszym, najważniejszym zadaniem jest dodanie listy prowadzących.

Prowadzący za swoje dane logowania przyjmują imię, nazwisko oraz hasło. Administrator powinien ustawić hasła jednorazowe i pouczyć prowadzących o konieczności zmiany tegó hasła po pierwszym logowaniu i ustaleniu odpowiedniej polityki haseł.



Ilustracja 20: Widok dodawania prowadzącego do bazy danych

Administrator może w tym momencie zdefiniować możliwe przedmioty i dodać studentów do listy wszystkich studentów na uczelni.

Może również rozproszy wykonanie tego zadania na prowadzących. Prowadzący mogą dodawać własne przedmioty i studentów do listy, później definiując, którzy studenci muszą uczęsczać na jaki przedmiot.

## Aplikacja stanowiska roboczego

Aplikacja stacji roboczej jest minimalna i działa w tle systemu operacyjnego. By zaszła jakakolwiek widoczna interakcja użytkownika z programem należy wsunąć legitymację do czytnika w obudowie komputera lub tego podłączonego przez USB.

*Uwaga: program stacji roboczej do poprawnego funkcjonowania wymaga zainstalowanego środowiska języka Python w wersji 3 i wyżej.*

Program działa w sposób zbliżony do sterownika. Po uruchomieniu programu z pliku .exe należy zweryfikować jego działanie z poziomu dowolnego menedżera procesów systemu.

Jeśli widoczny jest proces aplikacji można rozpoczęć pracę z systemem. Jeśli wszystko zostało poprawnie skonfigurowana wszystkie elementy systemu powinny się ze sobą komunikować.

Zaleca się dodanie pliku wykonywalnego aplikacji stacji roboczej do programów uruchamianych przy starcie systemu (ang. autostart).

## Interesujące problemy i rozwiązania

W trakcie fazy implementacyjnej projektu zespół natrafił na różne problemy. Ponizej znajduje się ich opis wraz z zastosowanymi obejściami.

## Uruchamianie aplikacji w tle

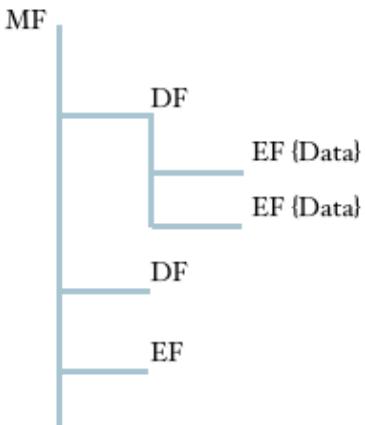
W przypadku aplikacji stacji roboczej wykorzystanie języka *Python* nie umożliwia w prosty sposób definiowania aplikacji działających w tle. Język Python skupia się na pisaniu głównie skryptów wywoływanych w konsoli.

By umożliwić działanie programu w sposób podobny do sterownika wykorzystano narzędzie *bat2exe*. Pozwoliło ono na przekonwertowanie aplikacji do postaci wykonywalnej(.exe) i uruchamianie jej w tle systemu operacyjnego przy jednoczesnym nie zawieszaniu wykonywania jej funkcji. Pomimo tego, że program nie jest widoczny jego główna pętla jest stale wykonywana.

## Komunikacja z kartą

Pierwotnie projekt zakładał wykorzystanie języka niskiego poziomu, takiego jak C lub C++. Ostatecznie ze względu na dostępność bibliotek i prostotę użycia zastosowano język *Python*.

Największym wyzwaniem okazało się odczytywanie rekordów z karty intelligentnej. Biblioteki udostępniają jedynie funkcje do komunikacji z kartą jednak nie posiadają zdefiniowanych gotowych funkcji do pobierania danych z kart. Wysyłane jednostki zapytań (APDU) należy konstruować samemu w postaci tablic bajtów i wysyłać do karty, licząc, że zadeklarowana spodziewana ilość informacji zwrotnych jest wystarczająca.



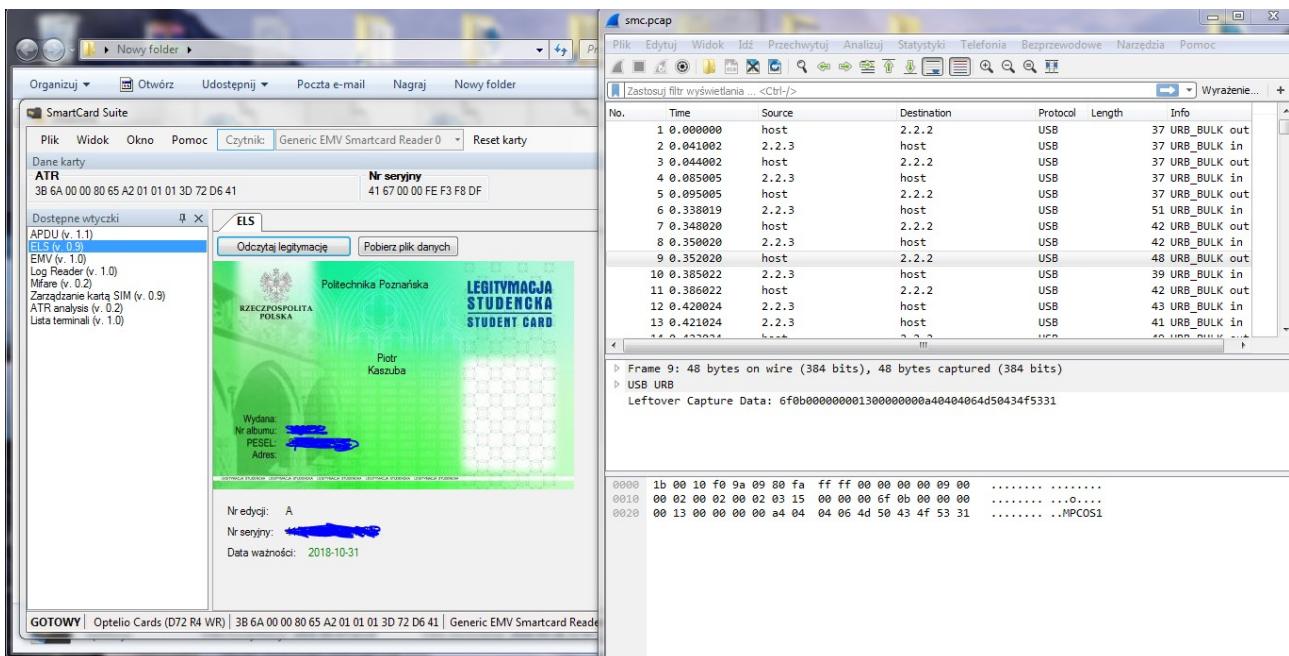
Data Structure as described in the ISO7816

### Ilustracja 21: Struktura katalogów z danymi na kartach inteligentnych

Samo zrozumienie standardu ISO opisującego sposób przechowywania danych i komunikacji z kartami inteligentnymi jest dość skomplikowany. Karta posiada tak zwany *Master File*, który posiada rekordy danych, które tak naprawdę są interesujące na potrzeby opisywanej aplikacji.

Ponieważ komunikacja z kartami inteligentnymi jest zagadnieniem dobrze udokumentowanym lecz mało wspominanym w sieci ciężko było znaleźć przykłady komunikacji z taką kartą.

By lepiej zrozumieć naturę zapytań i odpowiedzi kart *smart cards* wykorzystano gotową aplikację do czytania danych z karty w połączeniu z podsłuchem pakietów wysyłanych magistralą USB.

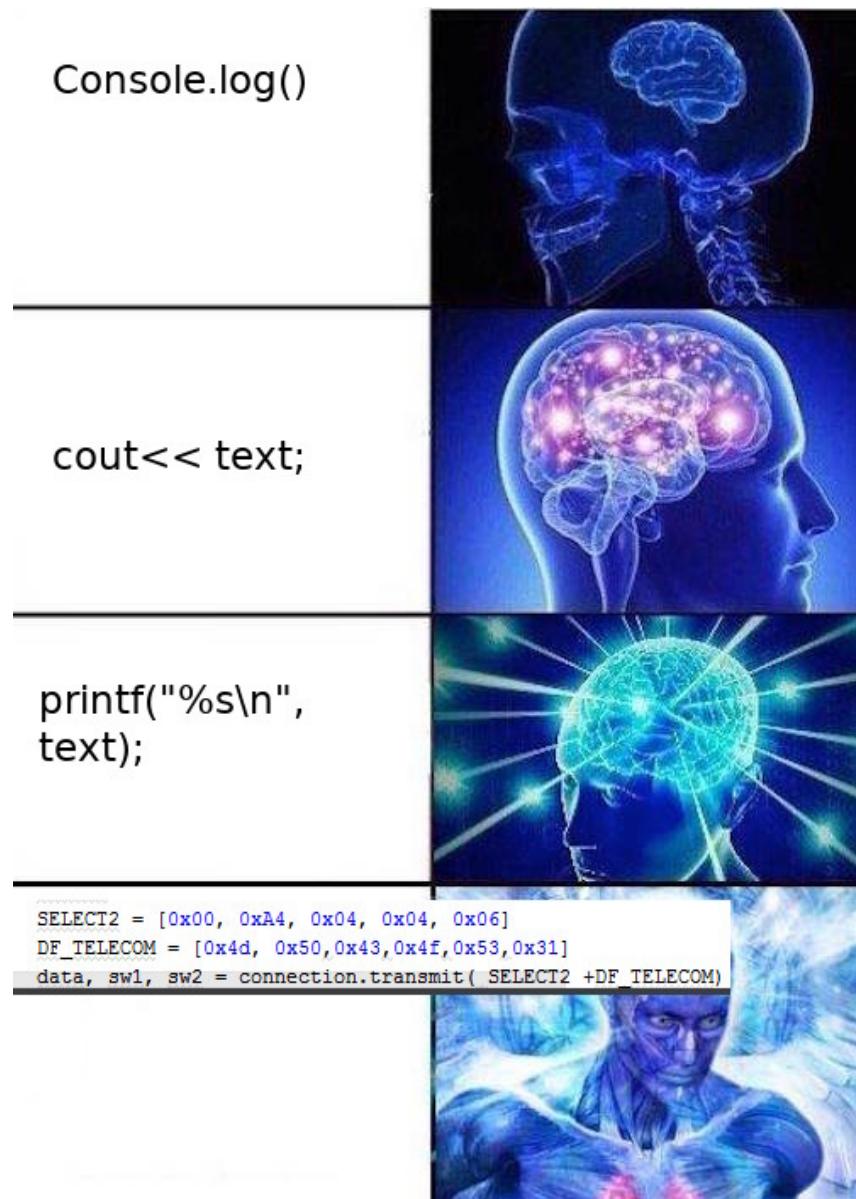


### Ilustracja 22: Połączenie aplikacji do komunikacji z kartami inteligentnymi z podsłuchem magistrali USB

Dopiero niskopoziomowa analiza komunikacji pozwoliła na zrozumienie charakteru zapytań i interpretacji odpowiedzi.

## Efekty uboczne powstawania projektu

Ponieważ w trakcie implementacji kolejnych funkcjonalności i w miarę z zapoznawaniem się z zagadnieniami wynikło wiele ciekawych sytuacji, które zaowocowały powstaniem poniższej grafiki.

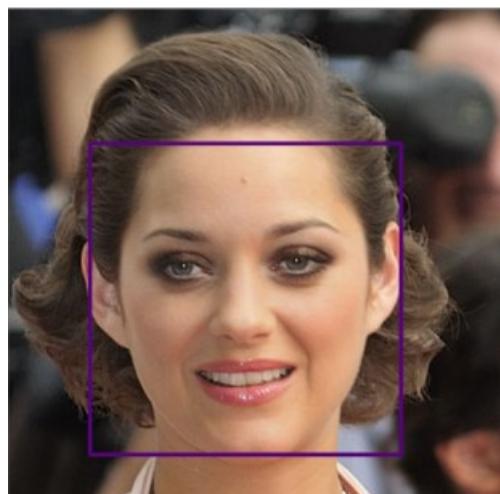


Ilustracja 23: Komunikacja użytkownik - komputer w różnych językach w porównaniu z komunikacją z kartą inteligentną

## Perspektywy rozwoju:

Projekt w swojej obecnej postaci realizuje jedynie podstawowe założenia. Ze względu na szerokie zagadnienie jakim jest automatyzacja procesu rejestracji obecności można rozważyć dodatkowe funkcje, o które można by rozszerzyć implementację.

- W celu usprawnienia aplikacji można rozwinać ją o sieć neuronową, bądź klasyfikator wyszukujący dane w karcie by dostosować ją do wszystkich rodzajów i odmian elektronicznych legitymacji, ponieważ elektroniczne legitymacje posiadają w różnych miejscach dane osobowe dotyczące właściciela przykład legitymacja prowadzącego, czyli legitymacja doktorancka w innym miejscu posiada informacje dotyczące danych osobowych niżeli legitymacja studencka na studiach inżynierskich lub magisterskich. Każda nowa iteracja legitymacji (wydawane co kilka lat) posiada inną pozycję poszczególnych informacji.
- Kolejnym możliwym do realizacji modułem jest ukazanie prowadzącemu osoby, które się spóźniły, czyli zalogowały się po czasie bądź z przyczyn usprawiedliwionych nie stawiły się na zajęciach by prowadzący mógł uzupełnić listę studentów ręcznie w systemie, to by również było rozwiązaniem w przypadku jeżeli sieć komputerowa na uczelni uległa by awarii bądź któryś z czytników byłby uszkodzony.
- Można rozwinać aplikację również o moduł rozpoznawania twarzy który mógłby rozpoznać czy dany użytkownik to właściciel legitymacji w celu uniknięcia oszustw odnośnie podszywania się pod studenta. Na pomoc w tym module pojawia się język Python wraz z biblioteką OpenCV które w prosty sposób pomagają rozpoznać twarz. Należały zaprojektować bazę zdjęć oraz wytrenować klasyfikator. W bibliotekach OpenCV jest kilka wytrenowanych klasyfikatorów w OpenCV.



*Ilustracja 24: Rozpoznana twarz na obrazie.*

Ilustracja 24 przedstawiona powyżej ukazuje nam jaki fragment zdjęcia jest oznaczony jako twarz. Skoro program rozpoznał, że na danym obrazie jest już twarz teraz należałyby wykryć cechy szczególne, które są najbardziej istotną rzeczą, gdyż po samej twarzy nie możemy zdefiniować czy ta osoba to ta osoba. Więc trzeba wybrać poszczególne części takie jak usta i oczy które określają czy pozwolą określi, że „my to my”.

Oczy i usta wykrywamy tą samą metodą co wykryliśmy twarz tylko wykorzystujemy inną kaskadę oraz dopasować wykrywanie tylko do części, którą jest twarz w celu uzyskania dokładnych pozycji oczu i tak samo ust. Następnie należałoby porównać obraz który mamy w bazie z obrazem wykonanym przez kamerkę w sali laboratoryjnej. Ponownie z pomocą przychodzi nam OpenCV dzięki któremu będziemy mogli porównać twarz z zdjęciem z bazy. Ostatecznym krokiem pozostało usprawnienie tego modułu by nie można było podłożyć zdjęcia by wykrywałoby twarz poprzez ruch by było widać na przykład uszy.

## Przebieg pracy

Zespół projektowy nie okrzystał z żadnych z tak zwanych "metodyk zwinnych" w celu usprawnienia procesu powstawania aplikacji konieczne było przyjęcie pewnego harmonogramu spotkań i podziału prac.

### 29.03 – Zapoznanie się z tematami prac

- Zapoznanie się z *APDU*
- Przygotowanie harmonogramu
- Podział pracy na członków zespołu
- Zapoznanie się z czytnikiem
- Zapoznanie się z biblioteką *pyscard*

### 12.04 – Implementacja aplikacji czytającej dane z karty

- Implementacja algorytmu
- Szukanie w karcie danych
- Sprawdzenie poprawności danych za pomocą aplikacji *SmartCard Suite*

### 26.04 – Projektowanie bazy

- Zaprojektowanie bazy odpowiedniej do wykonywanego zadania
- Realizowanie bazy w *MySql*
- Zapoznanie z biblioteką *MySQLdb* w środowisku python

### 10.05 – Projektowanie i implementacja aplikacji klienckiej C#

- Implementacja prostego formularza logowania
- Implementacja widoków podglądu bazy danych
- Implementacja procedur składowanych w bazie danych

## **24.05 – Testowanie i sprawdzanie poszczególnych modułów**

- Testowanie aplikacji czytającej dane karty
- Testowanie aplikacji klienckiej
- Testowanie poprawności procedur składowanych
- Sprawdzenie poprawności schematu bazy danych
- Poprawienie poszczególnych błędów
- Obsługa wyjątków

## **07.06 – Testowanie ostatecznej wersji aplikacji**

- Sprawdzanie aktualizacji bazy danych
- Przygotowanie interfejsu graficznego WPF
- Przygotowanie do prezentacji projektu

Na każdym spotkaniu byli obecni wszyscy członkowie zespołu implementacyjnego. Głównym założeniem spotkań było dokonanie i połączenie modułów w jedną, spójną całość. Każde spotkanie z założenia miało owocować jak największym postępem w możliwie najkrótszym czasie. Pod koniec spotkania przydzielane były kolejne zadania.

## Podsumowanie

Możliwość wykonania projektu o bardziej złożonej architekturze i z koniecznością niskopoziomowej obsługi sprzętu okazała się bardzo kusząca. Wymagało to od zespołu projektowego podejścia uwzględniającego więcej aspektów niż tylko obsługi prostego modelu klient - serwer.

W przypadku tego projektu model ten obsługuje jeden główny serwer i wielu klientów podłączonych jednocześnie. Do tego dochodzi obsługa zdarzeń w bazie danych.



Każdy moduł projektu był pisany w innym języku co dało wgląd w możliwości współczesnych języków programowania i ich współpracy między sobą.

Ilość dostępnych bibliotek koniecznych do realizacji projektu pozwoliła dobrą odpowiednio rozbudowaną bibliotekę w zależności od poziomu skomplikowania modułu oraz jego wymaganych funkcjonalności.

# Bibliografia

Do powstania projektu konieczne było zapoznanie się z strukturą danych zapisanych na kartach inteligentnych oraz metodami czytania zapisanych na nich danych. Najlepszym źródłem informacji na wszystkie potrzebne tematy były strony internetowe takie jak *Wikipedia* czy *StackOverflow*.

Opis jednostek komunikacji z kartami inteligentnymi:

[https://en.wikipedia.org/wiki/Smart\\_card\\_application\\_protocol\\_data\\_unit](https://en.wikipedia.org/wiki/Smart_card_application_protocol_data_unit)

Dokumentacja biblioteki języka *Python* do komunikacji z kartami inteligentnymi:

<https://pyscard.sourceforge.io/>

Informacje o kartach inteligentnych:

[https://en.wikipedia.org/wiki/Smart\\_card](https://en.wikipedia.org/wiki/Smart_card)

Czytanie informacji z kart inteligentnych:

<https://stackoverflow.com/questions/28472218/howto-list-files-on-a-smartcard-with-pyscard>

Przepływanie karty inteligentnej w języku JAVA:

<https://stackoverflow.com/questions/24563081/aceessing-smart-card-file-structure>

Struktura plików na kartach inteligentnych:

<https://www.cse.iitk.ac.in/users/moona/smardcard/index.php?page=smartid/smertos>