

Wykonanie:  
Piotr Pietras  
Witold Parysek

## SPRAWDZANIE OBECNOŚCI W LABORATORIUM Z WYKORZYSTANIEM LEGITYMACJI STUDENCKIEJ.

# SPIS TREŚCI

SPIS TREŚCI .....	2
1. Opis i uzasadnienie dlaczego wybrany został dany temat. ....	3
2. Podział prac pomiędzy członków zespołu: .....	4
3. Wylistowanie i opisanie funkcjonalności oferowanej przez aplikację:.....	5
4. Wybrane technologie wraz z uzasadnieniem dlaczego: .....	6
5. Architektura rozwiązania (jak jest zbudowane). ....	7
6. Interesujące problemy i rozwiązania ich na jakie się .....	10
natknęliśmy. ....	10
7. Instrukcja użytkowania aplikacji.....	10
7.1. Instrukcja dla studenta:.....	11
7.2. Instrukcja dla prowadzącego: .....	11
7.3. Instrukcja dla administratora: .....	13
8. Pomysły na dalszy rozwój aplikacji. ....	16

# **1. Opis i uzasadnienie dlaczego wybrany został dany temat.**

Temat jakie wybraliśmy “Sprawdzanie obecności z w laboratorium z wykorzystaniem legitymacji studenckiej” w założeniu miał dawać możliwość scentralizowanego zarządzania obecnościami studentów. Poniżej znajduje się opis samego rozwiązania oraz założeń, które la realizacji projektu przyjęliśmy.

## **Zarządzanie obecnościami przy użyciu kart inteligentnych:**

Zastanawiając się nad różnorodnymi możliwościami, postanowiliśmy przyjąć, że obecność będzie sprawdzana z czytnika, który znajduje się sali laboratoryjnej, na której uruchomiony jest program. Brakuje tutaj walidacji urządzenia, z jakiego dokonywany jest odczyt karty, jednakże można ze spokojem przyjąć, że taki odczyt będzie tylko możliwy w sali laboratoryjnej, w wewnętrznej sieci akademickiej, gdzie tylko stamtąd byłaby możliwość podłączenia się do ustawionej bazy danych. W obecnej wersji programu odczyt możliwy jest tylko dla osoby, która ma lokalnie utworzoną bazę danych.

## **Scentralizowany system zarządzania obecnością:**

W warunkach domowych pracowaliśmy na lokalnie dostępnych bazach danych, jednakże zmieniając parametry dla connection stringa można podłączyć się do bazy zdalnej, gdzie można mówić już o scentralizowanym zarządzaniu obecnością. Importowanie/wczytywanie listy studentów jest możliwe z poziomu administratora aplikacji. Może on przygotować plik z odpowiednimi identyfikatorami.

## **Obszerna baza danych:**

Nasze rozwiązanie obsługuje obecność na różnych przedmiotach. Owe przedmioty zostały podzielone na laboratoria i wykłady, przy odpowiednim przypisaniu różnych sal czy też prowadzących. Aplikacja oferuje także podział z uwagi na wydział i kierunek, na którym znajduje się dany przedmiot. Administrator jest jedyną osobą, która zarządza studentami i ich statusami, a prowadzący osobą, która może modyfikować obecność dla wszystkich swoich zajęć. Z naszego projektu może także korzystać student, który może sprawdzić swoją obecność na wszystkich swoich zajęciach. Zaoferowaliśmy także możliwość eksportowania danych z bazy danych.

## **Nasza motywacja:**

Temat związany ze sprawdzaniem obecności przy użyciu czytnika ELS, został przez nas wybranych z uwagi na możliwości i wyzwania jakie ze sobą niośł. Widzieliśmy w tym szansę na możliwość wykorzystania wiedzy i umiejętności z przedmiotu bazy danych w zakresie stworzenia bazy studentów z odpowiednimi relacjami.

Sam czytnik kart inteligentnych oraz możliwość zajrzenia pod pretekstem projektu do elektronicznej legitymacji studenckiej były kolejnymi powodami dla wybrania tego tematu. Z perspektywy czasu okazało się niestety, że wiedza wyniesienia z podstaw technologii mikroprocesorowych oraz podstaw technologii, przy jednoczesnym braku dokumentacji z “zawartością” legitymacji czy też dokumentacją od prowadzących, utrudniła znacząco zgłębianie i wykorzystywanie wszystkich możliwości jakie niesie ze sobą obsługa karty, a nawet przyniosła kilka porażek.

Podsumowując, początkowe oczekiwania zostały w większości zrealizowane, jednak nie obyło się bez niespodzianek, które zostały opisane w jednym z poniższych punktów. Jednakże udało nam się dostarczyć wygodną formę zarządzania obecnościami dla prowadzących zajęcia, gdzie zostaje z nich ściągnięta potrzeba własnoręcznego tworzenia systemu do sprawdzania obecności na tak popularnych kartkach puszcanych na laboratoriach, bądź poświęcając czas samego prowadzącego, który w bólach musi odczytywać nabazgrołone nazwiska studentów, które niejednokrotnie zostają przeinaczane wystawiając biednego studenta na pośmiewisko grupy.

## **2. Podział prac pomiędzy członków zespołu:**

### **Piotr Pietras:**

- Utworzenie bazy danych MS SQL. Konfiguracja relacji pomiędzy wszystkimi tabelami. wraz ze schematami. Utworzenie wszystkich kwerend, używanych w kodzie.
- Utworzenie schematu Entity Data Model.
- Utworzenie dokumentacji.
- Stworzenie prezentacji.
- Widok Studenta: pozwala na sprawdzanie przedmiotów, na które student jest zapisany, wraz z ilością odbytych zajęć i ilością obecności na nich. Wyświetlanie listy zajęć.
- Widok Prowadzącego: Pozwala na sprawdzanie obecności na wszystkich przedmiotach, na których wykładowca jest ustawiony jako osoba prowadząca zajęcia. Ręczne sprawdzanie obecności. Widok podsumowania zajęć i frekwencji.
- Widok Administratora: Możliwość wczytywania danych studentów i przedmiotów. Możliwość eksportu danych do pliku. Możliwość importu danych dla studentów z pliku.
- Utworzenie funkcjonalności dla trzech grup osób: student, prowadzący, admin.

### **Witold Parysek:**

- Komunikacja aplikacji z czytnikiem kart
- Utworzenie graficznego interfejsu użytkownika
- Utworzenie interfejsu do ręcznego modyfikowania obecności
- Testowanie działania aplikacji

### **3. Wylistowanie i opisanie funkcjonalności oferowanej przez aplikację:**

#### **Odczytywanie danych z elektronicznej legitymacji studenckiej za pomocą czytnika kart inteligentnych:**

Aplikacja pozwala na odczyt unikatowego numeru karty (UID), która służy do połączenia informacji z bazy danych dla obiektu student i zaznaczenia obecności w jego planie zajęć, jeżeli w tym czasie ma przypisane zajęcia w odpowiedniej sali.

#### **Uwierzytelnianie użytkowników:**

Dostęp do aplikacji możliwy jest tylko po uprzednim zalogowaniu. Aplikacja rozróżnia na tej podstawie możliwe dostępy i odpowiednio ładuje widok przeznaczony dla użytkownika danego typu (administrator, prowadzący, student).

#### **Eksportowanie i importowanie danych z i do pliku:**

Możliwe jest załadowanie danych do systemu z poziomu aplikacji. Usługa przeznaczona jest tylko dla informacji związanych ze studentem. Można dodać nowych studentów do bazy, wraz z przypisaniem odpowiedniego kierunku i wydziału. Możliwe jest także modyfikowanie istniejących danych studenta podczas importu.

#### **Panel studenta:**

Panel ten dostarcza głównie funkcjonalność planu zajęć. Po zalogowaniu do systemu student ma dostępny widok, gdzie wyświetlane są informacje z przedmiotami, na które student jest zapisany. Również może skorzystać z funkcjonalności wylistowania wszystkich zajęć dla danego przedmiotu, wraz z datami oraz salami gdzie zajęcia się odbywają.

#### **Panel prowadzącego:**

Panel dostępny po zalogowaniu. W obecnej wersji jest to najbardziej rozbudowany panel. Pozwala on na podgląd wszystkich dostępnych przedmiotów na uczelni oraz wylistowanie zajęć prowadzonych przez pracownika akademickiego.

Główna funkcjonalność polega na możliwości podglądu obecności do wszystkich prowadzonych zajęć, oraz na ich ręcznej modyfikacji.

Dostarczona jest także na bieżąco odświeżająca się wersja widoku z podsumowaniem frekwencji studenta, co pomoże na bieżąco kontrolować poczynania studentów.

#### **Panel administratora:**

Umożliwia ładowanie oraz eksportowanie danych z aplikacji. Dostępna funkcjonalność przeznaczona obecnie tylko dla tabeli studentów wraz z ich przypisaniem na wykłady i kierunki.

## **4. Wybrane technologie wraz z uzasadnieniem dlaczego:**

### **Visual Studio 2013/2015 Community:**

Jest to popularne środowisko programistyczne posiadające wiele przydatnych narzędzi ułatwiających programowanie, w szczególności bardzo intuicyjny edytor graficznego interfejsu użytkownika oraz możliwość łączenia aplikacji z bazą danych znajdującej się Sql Server.

### **Dodatkowo użyliśmy w VS:**

Entity Framework.

ExcelDataReader.

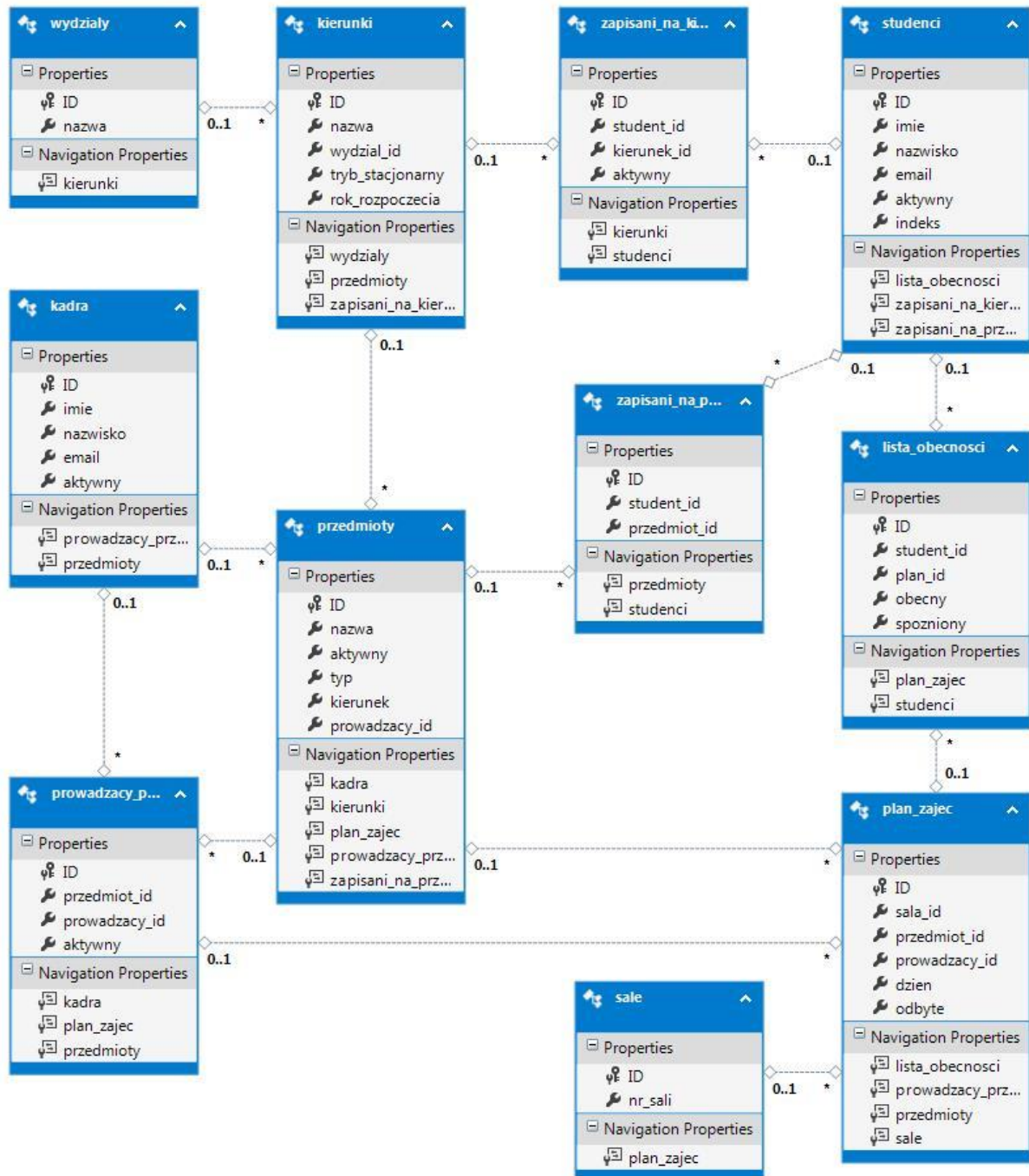
### **Stykowy czytnik kart inteligentnych firmy Manhattan, Model 102049**

Jest to model czytnika otrzymany przez nas od prowadzącego oraz dostępny na zajęciach laboratoryjnych, jednak aplikacja jest w stanie współpracować z dowolnym czytnikiem tego typu kompatybilnym ze standardem ISO 7816 i interfejsem PC/SC bez dodatkowej ingerencji w kod źródłowy.

### **Sql Server 2014 Management Studio**

## 5. Architektura rozwiązania (jak jest zbudowane).

Poniżej widoczny jest Entity Data Model odpowiadający relacją w bazie danych:



Nasze rozwiązanie jest obecnie dostępne lokalnie na maszynie uruchamiającej. Aby móc uzyskać dostęp do danych z bazy danych należy mieć lokalny serwer MS SQL oraz utworzoną bazę w MS SQL.

Wymagane jest utworzenie bazy. Aby ułatwić to zadanie na repozytorium zostały umieszczone odpowiednie pliki, które należy wykonać w Microsoft SQL Server Management Studio kod.

Poniższy kod tworzy bazę danych i wszystkie tabele wraz z ograniczeniami.

```
CREATE DATABASE PTproject;
CREATE TABLE studenci
(
    ID int primary key,
    imie varchar(255) not null,
    nazwisko varchar(255) not null,
    email varchar(255),
    aktywny bit default 1,
    indeks int UNIQUE CHECK (indeks>=0)
);

CREATE TABLE lista_obecnosci
(
    ID int primary key,
    student_id int FOREIGN KEY REFERENCES studenci(ID),
    plan_id int FOREIGN KEY REFERENCES plan_zajec(ID),
    obecny bit default 0,
    spozniony bit default 0
);
```

Dane do tabeli ładowane są w ten sam sposób. Należy załadować dane w MSSQL Server Mangement Studio:

```
INSERT INTO studenci (id, imie, nazwisko, aktywny, indeks)
VALUES ('1', 'Piotr', 'Pietras', '1', '122022')
INSERT INTO studenci (id, imie, nazwisko, aktywny, indeks)
VALUES ('2', 'Witold', 'Parysek', '1', '100357')
INSERT INTO studenci (id, imie, nazwisko, aktywny, indeks)
VALUES ('3', 'Jan', 'Beziemii', '1', '111222')
```

Od testera aplikacji wymaga się aby ustawił odpowiedni connectionstring w kodzie:

```
connetionString = "Data Source=WHYNOT-KOMPUTER\\SQLEXPRESS;Initial
Catalog=PTproject;Integrated Security=True";
sql = "select id, imie, nazwisko, indeks from studenci";
```

W naszym projekcie łączymy ze sobą przede wszystkim te elementy: Bazę danych, która jest obsługiwana w DataSetach w Visual Studio. Aktualizacje danych są wysyłane do bazy a dane w programie na nowo ładowane do DataSetu.

W projekcie mamy trzy główne elementy, które tworzą cały system: czytnik legitymacji studenckich, bazę danych oraz aplikacje

Diagram Entity Data Model dla obsługiwanej bazy danych:

Postawiliśmy tutaj akcent na sporą bazę danych, która kryje w sobie wiele możliwości rozwoju zarządzania danymi. Rozwiązanie jakie kryje się za wszystkimi formami jest bardzo podobne.

Wszystkie dane są ładowane przy pomocy kwerend stworzonych w kodzie programu i następnym ich wykonaniu oraz wczytaniu do datasetów. Reprezentatywnym przykładem



mogą być poniższe linie kodu:

```
sql = "select pz.id, p.nazwa as Przedmiot, p.typ as Rodzaj, kadra.imie+\'  
\'+'+kadra.nazwisko as Prowadzacy, s.nr_sali, pz.dzien from przedmioty as p"  
    + " join kierunki as k on k.id = p.kierunek "  
    + " join wydzialy as w on w.id = k.wydzial_id "  
    + " join kadra on kadra.id = p.prowadzacy_id "  
    + " join plan_zajec as pz on pz.przedmiot_id=p.id "  
    + " join sale as s on s.id=pz.sala_id"  
    //+ " where p.prowadzacy_id =1 "  
    // + " where p.id = 0";  
    + " where p.id =" + dataGridView1.SelectedCells[0].Value.ToString();  
    zm = dataGridView1.SelectedCells[0].Value.ToString();
```

Powyższa kwerenda jest aktualizowana dodatkowo o wartość sczytywaną z DGV.

Po ustawieniu stringów, kod jest wykonywany w poniższy sposób, gdzie następuje wpisanie danych do datasetów i wyświetlenie w DGV.

```
connection.Open();  
command = new SqlCommand(sql, connection);  
adapter.SelectCommand = command;  
sDs = new DataSet();  
cofnijSet = new DataSet();  
dataGridView1.DataSource = null;  
adapter.Fill(sDs, "przedmioty");  
adapter.Fill(cofnijSet, "cofnij");  
adapter.Dispose();  
command.Dispose();
```

Warto wspomnieć o krótkim kodzie, który jest wykorzystywana do aktualizowania danych. Kod ten w zasadzie wystarcza do wprowadzenia dalszych funkcjonalności związanych z aktualizowaniem informacji w bazie danych.

```
using (SqlConnection connection = new SqlConnection(connetionString))  
{  
    using (SqlCommand command = new SqlCommand())  
    {  
        command.Connection = connection;  
        connection.Open();  
        for (int i = 0; i < dataGridView1.Rows.Count; i++)  
        {  
            StrQuery = @"update lista_obecnosci set obecny='" +  
dataGridView1.Rows[i].Cells["obecny"].Value.ToString() + "' where id='" +  
dataGridView1.Rows[i].Cells["ID"].Value.ToString() + "';";  
  
            command.CommandText = StrQuery;  
            command.ExecuteNonQuery();  
        }  
    }  
}
```

Powyższy kod w różnych wariacjach dostarcza możliwość do obsłużenia całej aplikacji i dodania nowych funkcjonalności. Programistycznie jest to gotowe rozwiązanie, które wymaga już tylko zmodyfikowania do odpowiednich tabel, kolumn i pól, które chcemy wczytać bądź zaktualizować.

## 6. Interesujące problemy i rozwiązania ich na jakie się natknęliśmy.

Niestety nie udało się w pełni zrealizować odczytywania danych z karty. W związku z napotkanymi problemami nie udało nam się stworzyć aplikacji poprawnie odczytującej UID legitymacji. Przeanalizowaliśmy różne przykłady kodów dostępne w sieci, poradniki oraz dyskusje na forach, jednak żadne przedstawione tam rozwiązanie albo nie dawało interesujących nas wyników, albo nie działało w naszym domowym środowisku poprawnie. Obszerne dokumentacje dotyczące obsługi kart inteligentnych nie przedstawiały sprawy zbyt intuicyjnie, wymagały dużego nakładu czasowego do zrozumienia jednocześnie przynosząc małe owoce. W związku ze zbliżającym się terminem oddania projektu musieliśmy porzucić poszukiwania rozwiązania tego problemu i poprzestać na tym, że czytnik odczytuje numer ATR karty, który niestety nie jest unikatowy. Jednakże posiadając legitymacje z różnych uczelni, mogliśmy w unikatowy sposób rozwiązać ten problem aby możliwość automatycznego sprawdzania obecności przy włożeniu karty, mogła mieć miejsce.

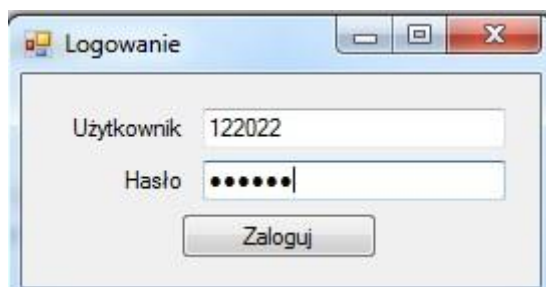
Zarządzanie danymi z plików xls ixlsx, stworzyło trochę problemów, jednak tutaj pomocne okazują się dwa rozwiązania.

```
using Excel = Microsoft.Office.Interop.Excel;  
using ExcelDataReader;
```

ExcelDataReader jest pomocny, jednakże po testach nie wprowadzaliśmy tej funkcjonalności. W Internecie jest niestety bardzo dużo nieaktualnych informacji do użycia tego kodu (nawet na stronie twórców). Dlatego wykorzystaliśmy rozwiązanie Microsoft.Office.Interop.Excel; dostępne po dodaniu referencji do biblioteki Microsoft Excel 11.0 Object Library.

## 7. Instrukcja użytkowania aplikacji.

Dostępne są trzy instrukcje dla administratora, studenta oraz prowadzącego. Aby uzyskać dostęp do konkretnego panelu, należy zalogować się do systemu:



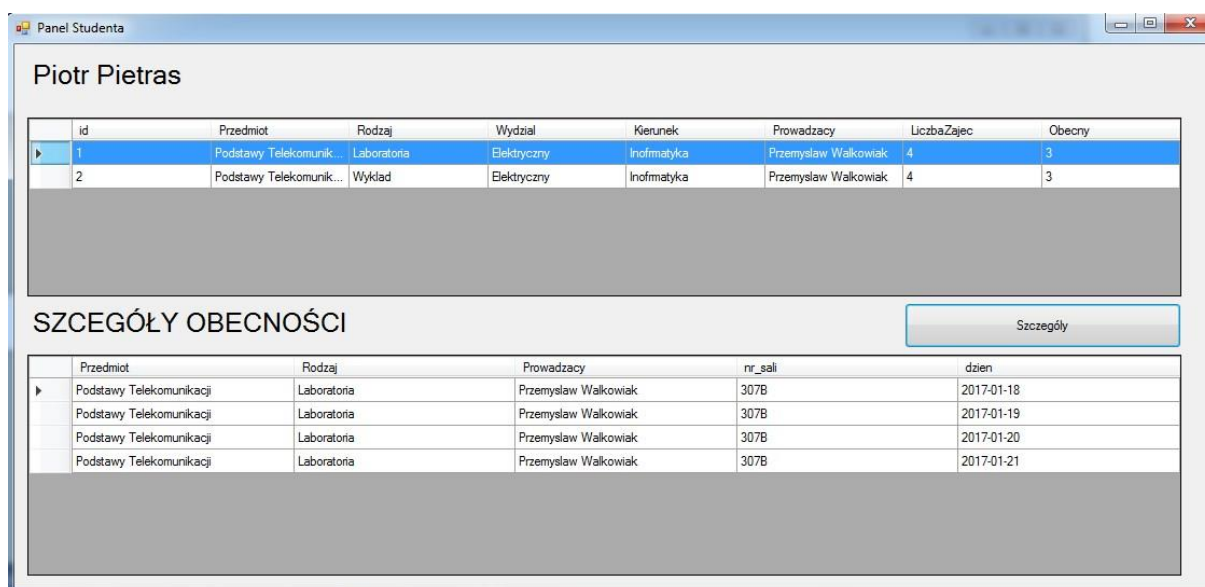
Po zalogowaniu odpowiednim użytkownikiem pojawi się odpowiednio jeden ze stworzonych paneli.

## 7.1. Instrukcja dla studenta:

Poniżej dostępny jest widok GUI studenta. Ładuje się on odpowiednio danymi dostępnymi z jego poziomu.

Student w aplikacji ma dostęp read-only. Możliwy jest podgląd wszystkich dostępnych zajęć. Każdy przedmiot na który student jest zapisany jest podzielony na wykłady i laboratoria. Istnieje tutaj możliwość zobaczenia ilości zajęć, które się odbyły oraz jego obecność na tych zajęciach.

Dodatkowo dostępny jest szczegółowy wykaz zajęć (plan zajęć). Aby uzyskać do niego dostęp należy zaznaczyć przedmiot z pierwszego DataGridView, a następnie kliknąć przycisk szczegóły.



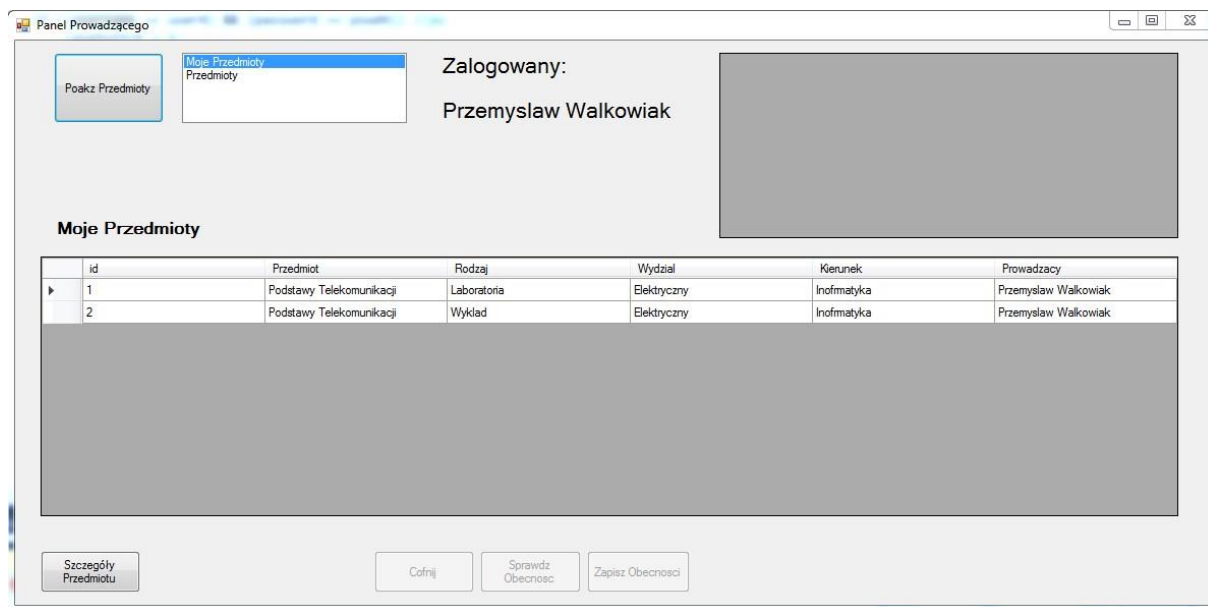
## 7.2. Instrukcja dla prowadzącego:

Panel prowadzącego w domyślnym widoku daje dostęp do załadowania przedmiotów.

Przedmioty, które można zobaczyć to wszystkie dostępne przedmioty bądź wyfiltrowany widok, własnych przedmiotów.

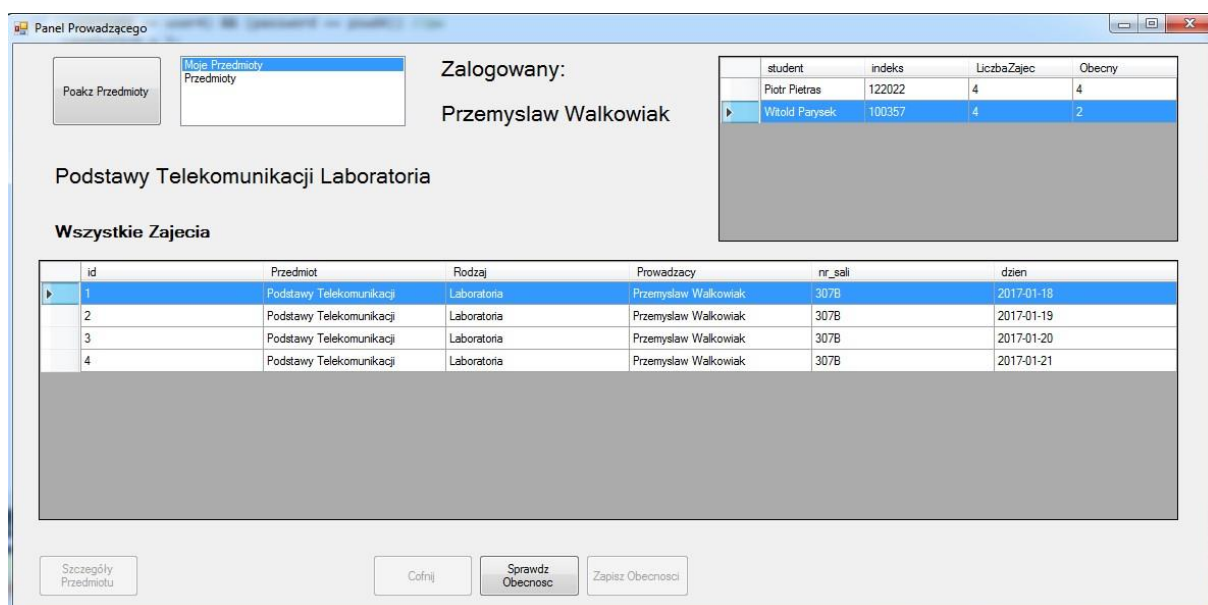
Aby zobaczyć przedmioty należy zaznaczyć widok z box i kliknąć przycisk „Pokaż Przedmioty”.

Poniżej jest przedstawiony efekt wykonania powyższej instrukcji.



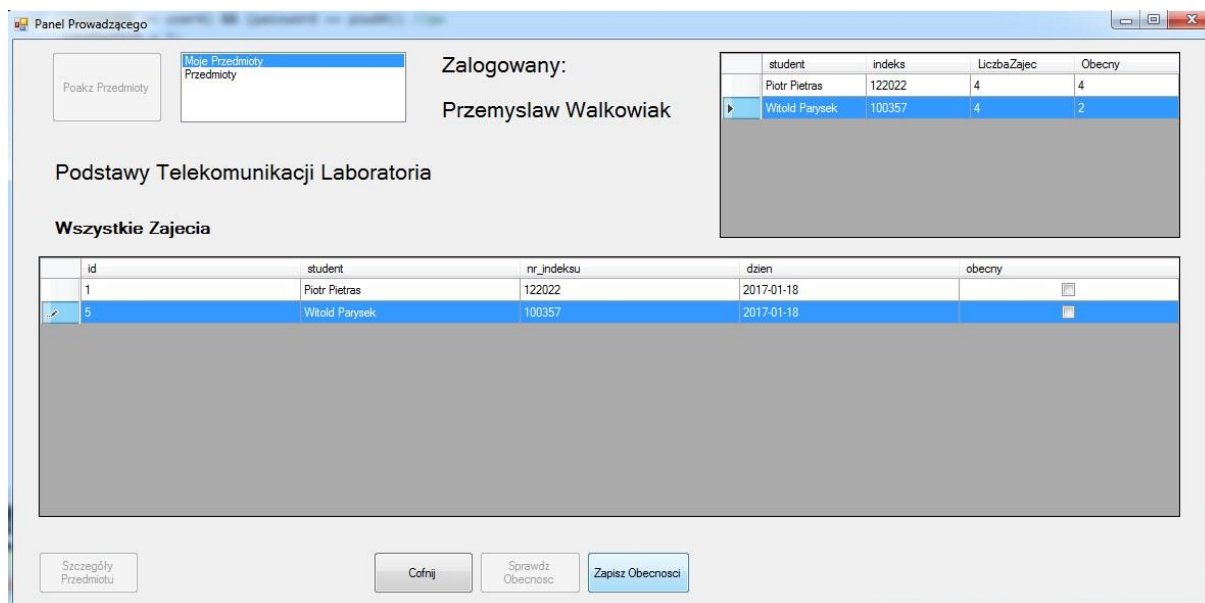
Możemy uzyskać szczegółowe informacje dla wybranych przedmiotów. Aby uzyskać takie informacje należy zaznaczyć przedmiot z DataGridView **Moje Przedmioty/Przedmioty** i kliknąć przycisk „Szczegóły Przedmiotu”. Wykonanie tej instrukcji spowoduje załadowanie danych do dwóch DGV. Większy DGV ładuje plan zajęć z informacjami id, przedmiot, rodzaj, prowadzący, nr\_sali oraz dzień.

Drugie okno wypełni się podsumowaniem obecności studentów na tym przedmiocie. Jest to widok pogrupowany z uwagi na studenta, indeks, liczbę zajęć oraz ilość obecności. Widok ten stworzony jest z myślą o sprawnym zarządzaniu obecnościami i ustalaniem czy student, który jest zapisany na zajęć, powinien uważać na swoje obecności. Poniżej zamieszczony jest widok aplikacji po załadowaniu szczegółów dane przedmiotu.



Powyższe okno daje tylko dwie możliwości dalszego poruszania się po programie. Funkcją jaką daje to szczegółowe okno przedmiotu to możliwość sprawdzenia obecności dla wskazanego rekordu. Przykładowo zaznaczając rekord o id równym 1, możemy kliknąć „Sprawdź Obecność”. Załaduje nam to dane ze wszystkimi studentami zapisanymi na konkretny przedmiot, dla których możemy sprawdzić obecność. Aby sprawdzić obecność należy posłużyć się checkboxem „obecny” i zaznaczyć odpowiednią wartość.

Jeżeli zakończyliśmy sprawdzanie obecności, należy kliknąć „Zapisz Obecności” a następnie użyć przycisku cofnąć aby wyjść z widoku sprawdzania obecności. Możemy też wrócić do wyświetlenia wszystkich/własnych przedmiotów.



student	indeks	LiczbaZajec	Obecny
Piotr Pietras	122022	4	4
Witold Parysek	100357	4	2

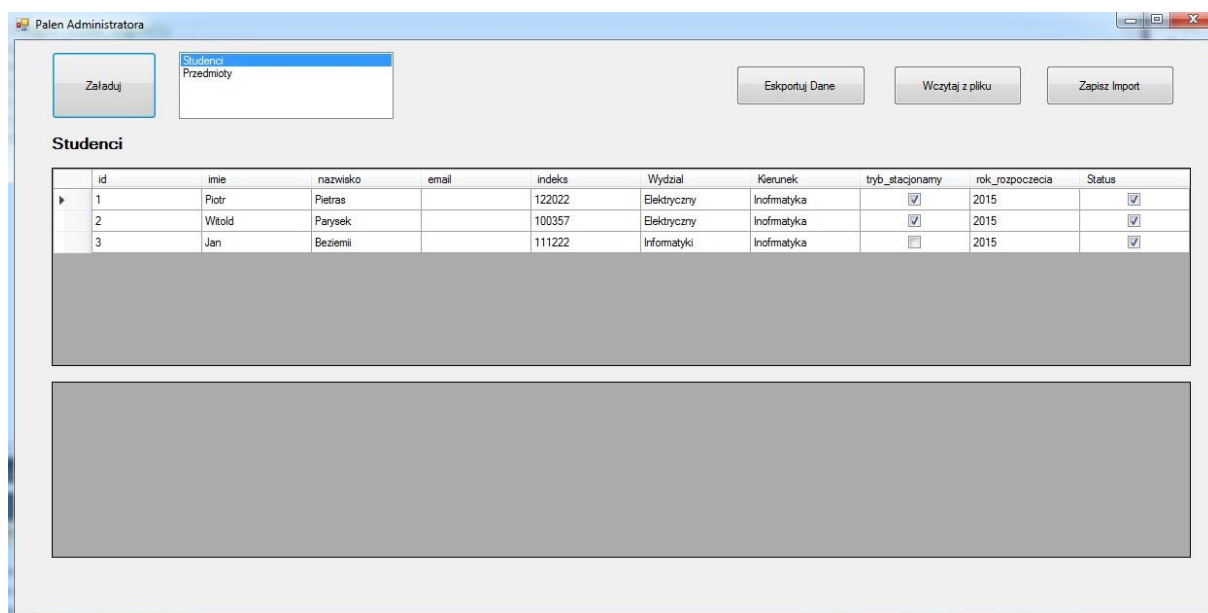
id	student	nr_indeksu	dzien	obecny
1	Piotr Pietras	122022	2017-01-18	<input type="checkbox"/>
5	Witold Parysek	100357	2017-01-18	<input type="checkbox"/>

Dodatkowo prowadzący ma możliwość edytowania obecności wstecz, dla wszystkich zajęć dostępnych w tym widoku.

### 7.3. Instrukcja dla administratora:

Administrator ma możliwość wczytania danych wszystkich studentów. Widok ten pozwala na wyświetlenie imienia, nazwiska, emaila, indeksu, wydziału, kierunku, trybu na którym jest student (checkbox zaznaczony wskazuje, że jest to tryb stacjonarny), rok rozpoczęcia studiów oraz status studenta. Zakłada się tutaj, że dane studenta nie są usuwane z bazy, tylko dezaktywowane.

Administrator, może wyeksportować dane studentów do pliku. Istnieje także możliwość eksportu danych dla przedmiotów.



Jeżeli chodzi o dane w DGB, to Administrator ma dostępny widoku read-only. Jednakże może zarządzać wszystkimi danymi poprzez wczytanie danych z pliku za pomocą przycisku „Wczytaj z pliku”. Powoduje to załadowanie danych do tabeli tymczasowej zamieszczonej pod główną tabelą.

Aby przygotować dane należy użyć pliku, który zawiera w sobie 9 kolumn ze wszystkimi wskazanymi danymi:

id	imie	nazwisko	email	indeks	Wydzial	Kierunek	tryb_stacjonarny	rok_rozpoczecia	Status
1	Piotr	Pietras		122022	Elektryczny	Informatyka	PRAWDA	2015	PRAWDA
2	Witold	Parysek		100357	Elektryczny	Informatyka	PRAWDA	2015	PRAWDA
3	Jan	Beziemii		111222	Informatyki	Informatyka	FAŁSZ	2015	PRAWDA
4	Karol	Głoski		111111	Informatyki	Informatyka	PRAWDA	2015	PRAWDA
5	Adam	Michałek		222222	Informatyki	Informatyka	FAŁSZ	2015	PRAWDA
6	Michał	Adamak		333333	Elektryczny	Budowa Mas	PRAWDA	2015	PRAWDA

Poniżej dostępny jest widok z dwiema tabelami. Tabela tymczasowo czeka już tylko na naszą aprobatę i użycie przycisku „Zapisz Import”. Chwilowo funkcjonalność ta niestety nie działa i dodawanie nowych danych jest możliwe tylko poprzez MSSQL.

Palen Administratora

Zaladuj

Studenci

Przedmioty

Esportuj Dane

Wczytaj z pliku

Zapisz Import

Studenci

	id	imie	nazwisko	email	indeks	Wydział	Kierunek	tryb_stacjonarny	rok_rozpoczecia	Status
►	1	Piotr	Pietras		122022	Elektryczny	Informatyka	<input checked="" type="checkbox"/>	2015	<input checked="" type="checkbox"/>
	2	Witold	Parysek		100357	Elektryczny	Informatyka	<input checked="" type="checkbox"/>	2015	<input checked="" type="checkbox"/>
	3	Jan	Beziemii		111222	Informatyki	Informatyka	<input type="checkbox"/>	2015	<input checked="" type="checkbox"/>

	id	imie	nazwisko	email	indeks	Wydział	Kierunek	tryb_stacjonarny	rok_rozpoczecia
►	1	Piotr	Pietras		122022	Elektryczny	Informatyka	True	2015
	2	Witold	Parysek		100357	Elektryczny	Informatyka	True	2015
	3	Jan	Beziemii		111222	Informatyki	Informatyka	False	2015
	4	Karol	Głoski		111111	Informatyki	Informatyka	True	2015
	5	Adam	Michalek		222222	Informatyki	Informatyka	False	2015
	6	Michał	Adamek		333333	Elektryczny	Budowa Maszyn	True	2015

## 8. Pomysły na dalszy rozwój aplikacji.

Aplikacja może zostać dalej rozbudowana dając więcej funkcjonalności do zarządzania danymi przez administratora. Docelowo administrator mógłby zarządzać wszystkimi danymi z bazy danych za pomocą aplikacji.

Aby stworzyć taką możliwość interesującą, należałoby skorzystać z gotowych picklist, które byłyby dostępne w polach aby łatwiej edytować dane.

W takim wypadku odejdzie potrzeba tworzenia plików do importów danych, a funkcja edycji rekordów w przyjemny sposób będzie dostępna.

Funkcja kalendarza. Widok prowadzącego/studenta dla planu zajęć, może zostać rozszerzony o kalendarz. Głównym beneficjentem takiego rozwiązania będzie jednak administrator, który będzie miał możliwość edycji planu zajęć.

Nowe triggerzy. W rozwiązaniu brakuje obecnie triggerów, które robiły by autozaładunki. Z uwagi na to, że obecnie administrator, nie przypisuje studentów do przedmiotów z poziomu aplikacji, takie rozwiązanie nie zostało wprowadzone. Jednakże, przewidujemy, że aplikacja zostanie o taką opcję rozszerzona.

Funkcjonalność, która za tym się kryje to na przykład: możliwość stworzenia rekordów w planie zajęć zaraz po tym kiedy student został przypisany na nowy przedmiot.