

**POLITECHNIKA CZĘSTOCHOWSKA**  
**WYDZIAŁ INŻYNIERII MECHANICZNEJ I INFORMATYKI**



**PROJEKT ZESPOŁOWY**

**System Obsługi Studentów**

Grupa 2 PAI

Zespół 6

Arkadiusz Kędziora

Filip Sokół

Mateusz Łuczyński

# Spis treści

<b>1</b>	<b>WSTĘP .....</b>	<b>3</b>
1.1	CEL PRACY .....	3
<b>2</b>	<b>WYMAGANIA BIZNESOWE .....</b>	<b>4</b>
2.1	WYMAGANIA FUNKCJONALNE: .....	4
2.2	WYMAGANIE NIEFUNKCJONALNE: .....	4
2.3	OGRANICZENIA SYSTEMU .....	4
<b>3</b>	<b>DOKUMENTACJA TECHNICZNA.....</b>	<b>5</b>
3.1	BACKEND.....	5
3.1.1	<i>Wykorzystane technologie</i> .....	5
3.1.2	<i>Architektura systemu</i> .....	6
3.1.3	<i>Endpointy</i> .....	6
3.1.4	<i>Diagram klas UML</i> .....	12
3.1.5	<i>Model bazy danych</i> .....	13
3.2	FRONTEND .....	14
3.2.1	<i>Wykorzystane technologie</i> .....	14
3.2.2	<i>Dokumentacja aplikacji</i> .....	15
3.2.2.1	Widoki aplikacji - Ogólne .....	15
3.2.2.2	Widoki aplikacji - Student.....	17
3.2.2.3	Widoki aplikacji – Nauczyciel .....	18
3.2.2.4	Widoki aplikacji – Admin .....	20
3.2.3	<i>Dokumentacja techniczna</i> .....	27
3.2.3.1	Nawigacja w projekcie .....	27
3.3	APLIKACJA MOBILNA .....	28
3.3.1	<i>Wykorzystane technologie</i> .....	28
3.3.2	<i>Dokumentacja użytkownika</i> .....	29
3.3.2.1	Ekrany Aplikacji - Ogólne .....	29
3.3.2.2	Ekrany Aplikacji - Student .....	35
3.3.2.3	Ekrany Aplikacji – Nauczyciel.....	40
3.3.3	<i>Dokumentacja techniczna</i> .....	47
3.3.3.1	Nawigacja w projekcie .....	47

---

# 1 Wstęp

Projekt zespołowy dotyczy projektu i implementacji Systemu obsługi studentów (SOS). Celem tego systemu jest umożliwienie studentom łatwiejszego dostępu do informacji dotyczących ich rozwoju oraz umożliwienie wymiany informacji między studentami a wydziałem.

W ostatnich latach, coraz więcej uczelni wprowadza systemy informatyczne służące do obsługi studentów, dlatego też System Obsługi Studentów będzie dobrym odpowiednikiem dla istniejących rozwiązań.

## 1.1 Cel Pracy

System ma na celu usprawnienie pracy administracyjnej oraz ułatwienie życia studentom, udostępniając im wiele przydatnych funkcjonalności. Dzięki SOS studenci będą mieli dostęp do swoich ocen i planu zajęć.

System będzie także umożliwiał zarządzanie kalendarzem zajęć i wydarzeniami na uczelni oraz komunikację z innymi studentami i pracownikami uczelni.

Głównym celem projektu jest więc zapewnienie studentom wygodnego i szybkiego dostępu do potrzebnych im informacji.

## **2 Wymagania biznesowe**

SOS ma być nowoczesnym narzędziem, umożliwiającym zarządzanie danymi o studentach oraz udostępniającym im różnego rodzaju usługi. W ramach projektu zostaną zaimplementowane m.in. następujące funkcjonalności:

### **2.1 Wymagania funkcjonalne:**

- Użytkownik może zarejestrować się oraz zalogować do aplikacji
- Użytkownik może wyświetlić swoje dane i w zależności od roli wykonywać adekwatne akcje na stronie.
- Użytkownik z rolą Nauczyciela może wystawiać studentom oceny.
- Użytkownik z rolą Admin może przypisywać studentów do grup

### **2.2 Wymaganie niefunkcjonalne:**

- Aplikacja powinna być cross-platformowa.
- Wykorzystanie technologii ASP.NET Core 6.0 / MS SQL / React Native 0.70 / ReactJS 18.2.0
- Dokumentacja techniczna oparta na Swashbuckle Swagger.

### **2.3 Ograniczenia systemu**

Aplikacja webowa nie jest dostosowana do wyświetlania na urządzeniach mobilnych. Do obsługi na smartfonach i tabletach stworzona została oddzielna aplikacja.

## 3 Dokumentacja techniczna

### 3.1 Backend

#### 3.1.1 Wykorzystane technologie

Technologie po stronie serwera

**ASP.NET Core 6.0** - to open-source platforma do tworzenia aplikacji internetowych i usług sieciowych, która jest rozwijana przez Microsoft. Wersja 6.0 to najnowsza wersja tej platformy, która została wydana w 2021 roku. ASP.NET Core 6.0 oferuje szereg nowych i ulepszonych funkcjonalności w stosunku do poprzednich wersji.

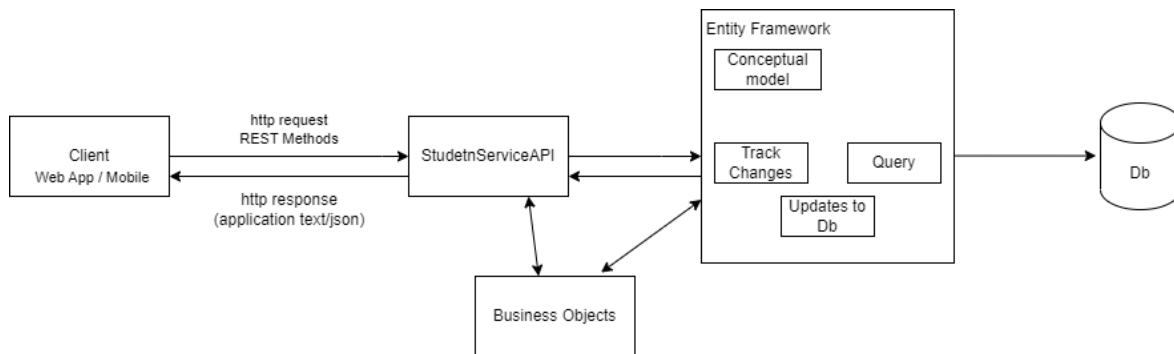
**Microsoft SQL Server (MS SQL)** - to system zarządzania bazami danych (DBMS) opracowany przez firmę Microsoft. Jest to bardzo popularne narzędzie do zarządzania dużymi zbiorami danych, które jest szeroko stosowane w różnych rodzajach organizacji, od małych firm po duże korporacje.

**Transact-SQL (TSQL)** - to język zapytań, który jest używany do zarządzania bazami danych i przetwarzania danych w systemie Microsoft SQL Server. Jest to rozszerzenie języka Structured Query Language (SQL), który jest powszechnie używany do zarządzania bazami danych w różnych systemach.

**Swagger** - to narzędzie do tworzenia dokumentacji interfejsu API (Application Programming Interface). Interfejs API to zestaw reguł, które pozwalają na komunikację między różnymi aplikacjami lub systemami. Dokumentacja interfejsu API opisuje sposób korzystania z danego interfejsu, w tym jakie są dostępne metody, jakie parametry należy przekazać i jakie dane zwracane są w odpowiedzi.

### 3.1.2 Architektura systemu

Klient poprzez aplikację webową bądź mobilną łączy się metodami REST z serwerem (StudentServiceAPI), które z wykorzystaniem frameworka Entity Framework Core wykonuje operacje na bazie danych.



[Model architektury systemu]

### 3.1.3 Endpointy

Każdy z obiektów biznesowych posiada własny endpoint umożliwiający tworzenie, odczytywanie, modyfikację oraz usuwanie obiektu z bazy danych.

Poniżej przedstawiono endpointy wraz z metodami:

Parametr	Typ	Opis
TBD JWT Bearer	string	Wymagany. Dla wszystkich operacji wymagany jest token autentykacyjny

#### Departments:

##### Wyświetl wszystkie obiekty typu Department:

GET /api/departments

##### Wyświetl obiekt typu Department o podanym Id

GET /api/departments/{id}

Parametr	Typ	Opis
id	int	Wymagane. Id wydziału

##### Dodaj obiekt typu Department

POST /api/departments

Body	Typ	Opis
------	-----	------

CreateDepartmentDto	serialized json	Wymagane. Id wydziału
---------------------	-----------------	-----------------------

#### Przykład body CreateDepartmentDto

```
{
  "Name": "Wydział Inżynierii",
  "Address": "Zielona 3",
  "City": "Częstochowa",
  "PostalCode": "42-700"
}
```

#### Edytuj obiekt typu Department

PUT /api/departments/{id}

Body	Typ	Opis
UpdateDepartmentDto	serialized json	Wymagane. Id wydziału

#### Przykład body UpdateDepartmentDto

```
{
  "Name": "Wydział Inżynierii",
  "Address": "Zielona 3",
  "City": "Częstochowa",
  "PostalCode": "42-700"
}
```

#### Usuń obiekt typu Department o podanym Id

DELETE /api/departments/\${id}

Parametr	Typ	Opis
id	int	Wymagane. Id wydziału

#### Groups:

#### Wyświetl wszystkie obiekty typu Group:

GET /api/departments/{departmentId}/groups

Parametr	Typ	Opis
departmentId	int	Wymagane. Id wydziału

#### Wyświetl obiekt typu Group o podanym Id

GET /api/departments/{departmentId}/groups/{groupId}

Parametr	Typ	Opis
departmentId	int	Wymagane. Id wydziału
groupId	int	Wymagane. Id grupy

#### Dodaj obiekt typu Group

POST /api/departments/{departmentId}/groups

Parametr	Typ	Opis
departmentId	int	Wymagane. Id wydziału
Body	Typ	Opis
CreateGroupDto	serialized json	Wymagane. Dane tworzonego Departamentu

### Przykład body CreateDepartmentDto

```
{  
  "Name": "group_2022_2",  
}
```

### Edytuj obiekt typu Group

PUT /api/departments/{departmentId}/groups/{groupId}

Parametr	Typ	Opis
departmentId	int	Wymagane. Id wydziału
groupId	int	Wymagane. Id grupy
Body	Typ	Opis
UpdateGroupDto	serialized json	Wymagane. Dane tworzonego Departamentu

### Przykład body UpdateGroupDto

```
{  
  "Name": "group_2022_2",  
}
```

### Usuń obiekt typu Group o podanym Id

DELETE /api/departments/{departmentId}/groups/{groupId}

Parametr	Typ	Opis
departmentId	int	Wymagane. Id wydziału
groupId	int	Wymagane. Id grupy

### Usuń wszystkie obiekty typu Group dla danego wydziału

DELETE /api/departments/{departmentId}/groups

Parametr	Typ	Opis
departmentId	int	Wymagane. Id wydziału

### Students:

### Wyświetl wszystkie obiekty typu Student:



GET /api/departments/{departmentId}/groups/{groupId}/students

Parametr	Typ	Opis
departmentId	int	Wymagane. Id wydziału
groupId	int	Wymagane. Id grupy

### Wyświetl obiekt typu Student o podanym Id

GET /api/students/{id}

Parametr	Typ	Opis
id	int	Wymagane. Id studenta

### Dodaj obiekt typu Student

POST /api/departments/{departmentId}/groups/{groupId}/students

Parametr	Typ	Opis
departmentId	int	Wymagane. Id wydziału
groupId	int	Wymagane. Id grupy
Body	Typ	Opis
CreateStudentDto	serialized json	Wymagane. Dane tworzonej grupy

### Przykład body CreateStudentDto

```
{
  "FirstName": "Jan",
  "LastName": "Kowalski"
}
```

### Edytuj obiekt typu Student

PUT /api/departments/{departmentId}/groups/{groupId}/students/{id}

Parametr	Typ	Opis
departmentId	int	Wymagane. Id wydziału
groupId	int	Wymagane. Id grupy
id	int	Wymagane. Id studenta
Body	Typ	Opis
UpdateStudentDto	serialized json	Wymagane. Dane tworzonej grupy

### Przykład body UpdateStudentDto

```
{
  "FirstName": "Jan",
  "LastName": "Kowalski"
}
```

### Usuń obiekt typu Student o podanym Id

DELETE /api/students/{id}

Parametr	Typ	Opis
id	int	Wymagane. Id studenta

## Marks:

### Wyświetl wszystkie obiekty typu Mark:

GET /api/students/{studentId}/marks

Parametr	Typ	Opis
studentId	int	Wymagane. Id studenta

### Wyświetl obiekt typu Mark o podanym Id

GET /api/students/{studentId}/marks/{id}

Parametr	Typ	Opis
studentId	int	Wymagane. Id studenta
id	int	Wymagane. Id oceny

### Dodaj obiekt typu Mark

POST /api/students/{studentId}/marks

Parametr	Typ	Opis
studentId	int	Wymagane. Id studenta
Body	Typ	Opis
CreateMarkDto	serialized json	Wymagane. Dane tworzonej oceny

### Przykład body CreateMarkDto

```
{
  "DateOfIssue": "2022-01-01:14:42:34",
  "SubjectId": 41,
  "Description": "This is a mark!",
  "StudentId": 132412,
  "MarkValue": 5
}
```

### Edytuj obiekt typu Mark

PUT /api/students/{studentId}/marks/{id}

Parametr	Typ	Opis
studentId	int	Wymagane. Id studenta
id	int	Wymagane. Id oceny

Body	Typ	Opis
UpdateMarkDto	serialized json	Wymagane. Dane tworzonej oceny

### Przykład body UpdateMarkDto

```
{
  "DateOfIssue": "2022-01-01:14:42:34",
  "SubjectId": 41,
  "Description": "This is a mark!",
  "StudentId": 132412,
  "MarkValue": 5
}
```

### Usuń obiekt typu Mark o podanym Id

DELETE /api/students/{studentId}/marks/{id}

Parametr	Typ	Opis
studentId	int	Wymagane. Id studenta
id	int	Wymagane. Id oceny

### Subjects:

### Wyświetl wszystkie obiekty typu Subject:

GET /api/subjects

### Wyświetl obiekt typu Subject o podanym Id

GET /api/subjects/{id}

Parametr	Typ	Opis
id	int	Wymagane. Id przedmiotu

### Dodaj obiekt typu Subject

POST /api/subjects

Body	Typ	Opis
CreateSubjectDto	serialized json	Wymagane. Dane tworzonego przedmiotu

### Przykład body CreateSubjectDto

```
{
  "Name": "Projekt zespołowy",
  "Description": "This is a subject",
  "StartTime": "2022-01-01:14:42:34",
  "EndTime": "2022-01-01:15:42:34",
}
```

```
"WeekDaysId": 3,  
"ECTS": 4,  
"TeacherId": 5  
}
```

### Edytuj obiekt typu Subject

PUT /api/subjects

Body	Typ	Opis
UpdateSubjectDto	serialized json	Wymagane. Dane tworzonego przedmiotu

### Przykład body UpdateSubjectDto

```
{  
  "Name": "Projekt zespołowy",  
  "Description": "This is a subject",  
  "StartTime": "2022-01-01:14:42:34",  
  "EndTime": "2022-01-01:15:42:34",  
  "WeekDaysId": 3,  
  "ECTS": 4,  
  "TeacherId": 5  
}
```

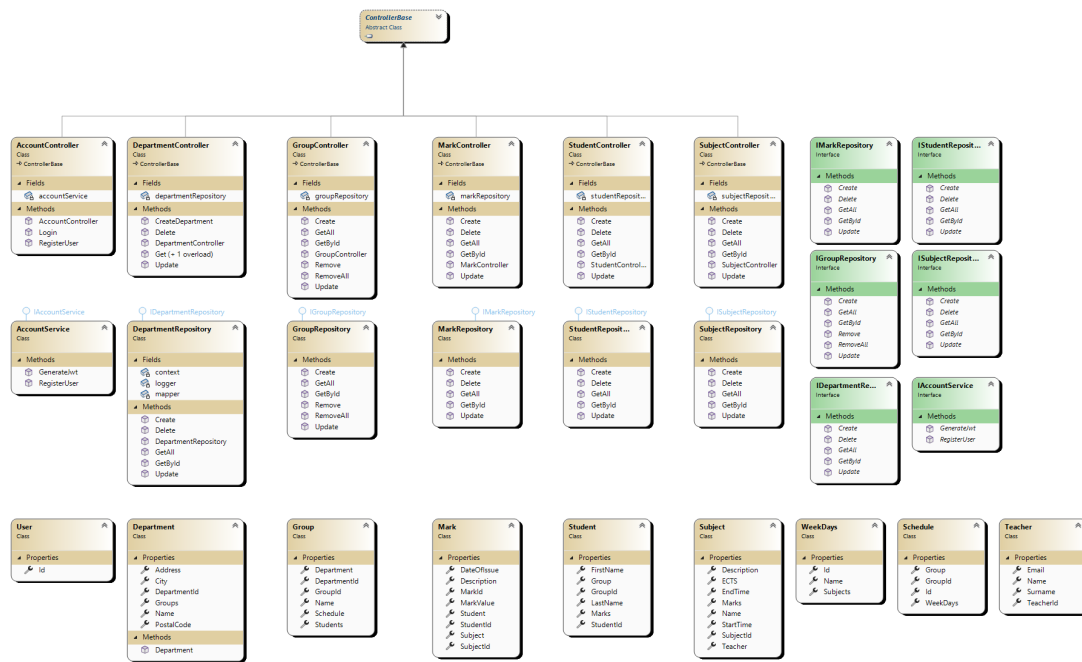
### Usuń obiekt typu Subject o podanym Id

DELETE /api/subjects/{id}

Parametr	Typ	Opis
id	int	Wymagane. Id przedmiotu

## 3.1.4 Diagram klas UML

Diagram klas opisujący główne klasy wykorzystane w API aplikacji.

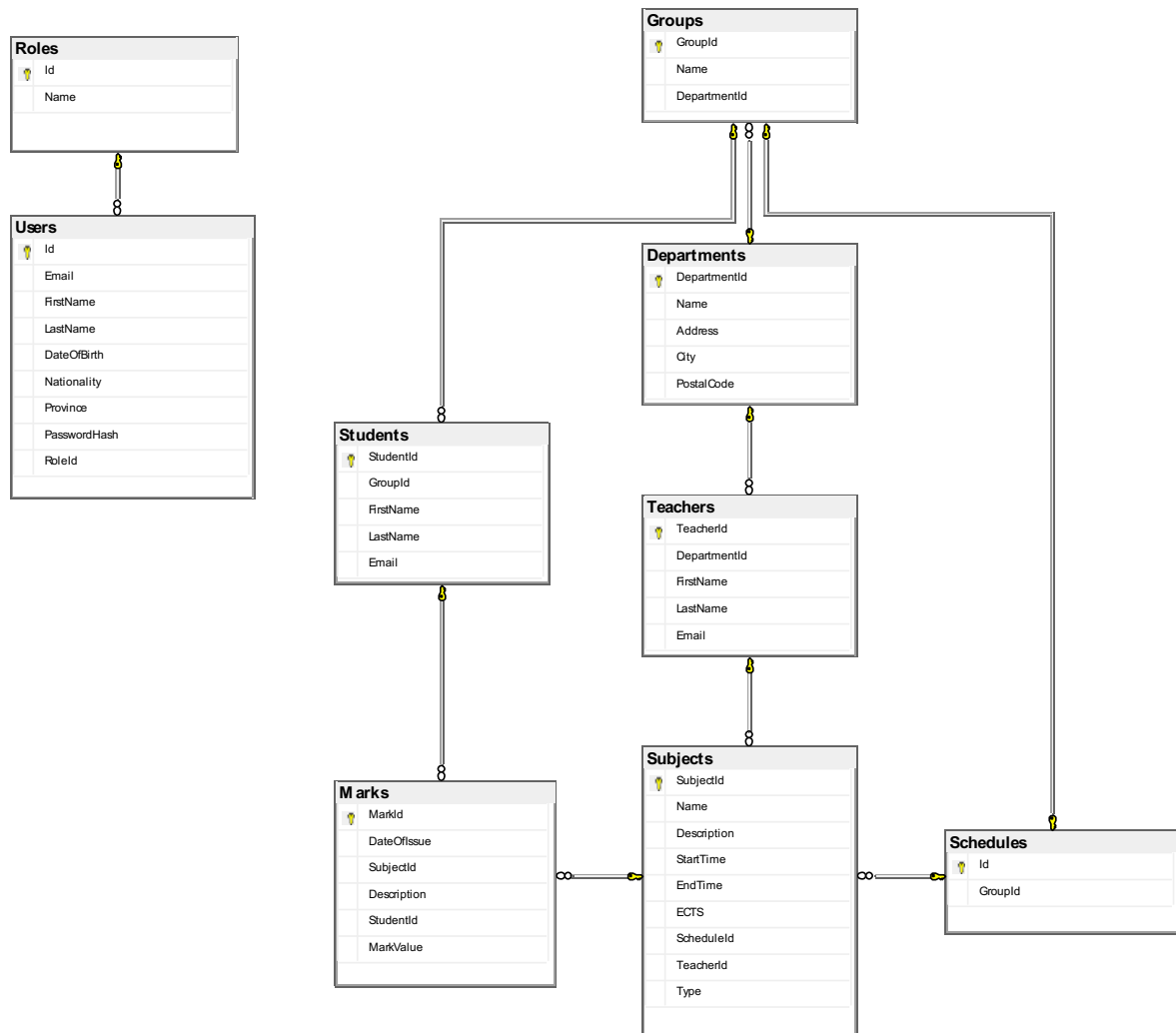


[Diagram klas API]

### 3.1.5 Model bazy danych

Model bazy danych to sposób reprezentowania struktury bazy danych oraz zależności między jej elementami. Model bazy danych jest używany do opisywania sposobu przechowywania danych oraz ich powiązań w bazie danych.

W Systemie Obsługi Studentów model bazy danych wygląda następująco:



[Model bazy danych]

## 3.2 Frontend

### 3.2.1 Wykorzystane technologie

Technologie po stronie aplikacji internetowej

**React** - to biblioteka JavaScript stworzona przez firmę Facebook, która służy do tworzenia interfejsów użytkownika dla aplikacji internetowych. Opiera się ona wykorzystaniu komponentów, które są małymi, odizolowanymi fragmentami kodu odpowiedzialnymi za wyświetlenie określonej części interfejsu. Komponenty są łatwe w utrzymaniu i rozszerzaniu, co sprawia, że React jest dobrym wyborem dla dużych projektów z rozbudowanymi interfejsami użytkownika.

**SCSS** - język preprocesora CSS, który umożliwia zapisywanie stylów dla stron internetowych w bardziej zaawansowanej i zoptymalizowanej formie. SCSS jest rozszerzeniem języka CSS i umożliwia używanie takich funkcji jak zmienne, selektory niestandardowe, mixiny (czyli funkcje, które pozwalają na "mieszanie" kilku stylów w jednym selektorze) i inne narzędzia, które ułatwiają i usprawniają tworzenie stylów dla stron internetowych.

**Ant Design** - to biblioteka komponentów React stworzona przez firmę Alibaba, która służy do tworzenia interfejsów użytkownika dla aplikacji internetowych. Zawiera szeroką gamę gotowych komponentów, takich jak przyciski, formularze, tabele, menu itp., które umożliwiają szybkie i łatwe tworzenie interfejsów użytkownika.

**Vite** - narzędzie deweloperskie służące do tworzenia aplikacji internetowych. Vite zostało zaprojektowane z myślą o szybkim uruchamianiu aplikacji i zapewnieniu lepszej wydajności niż inne narzędzia tego typu. Główną cechą Vite jest to, że nie wymaga on kompilacji kodu przed uruchomieniem aplikacji. Zamiast tego używa on specjalnego mechanizmu pozwalającego na dynamiczne ładowanie kodu podczas działania aplikacji, co pozwala na szybsze uruchamianie i lepszą wydajność. Vite również automatycznie generuje plik mapy źródeł (source map), co umożliwia łatwiejsze debugowanie kodu.

**Axios** - biblioteka JavaScript służąca do wysyłania i odbierania danych z serwerów przez protokół HTTP. Może być używana zarówno w aplikacjach internetowych, jak i w aplikacjach mobilnych. Axios umożliwia wysyłanie zapytań HTTP za pomocą metod takich jak GET, POST, PUT, DELETE itp., a także umożliwia konfigurację zapytań za pomocą opcji takich jak nagłówki, ciasteczka, autentykacja itp. Axios automatycznie parsuje odpowiedzi z serwera do formatu JSON, co umożliwia łatwe ich wykorzystanie w aplikacji.

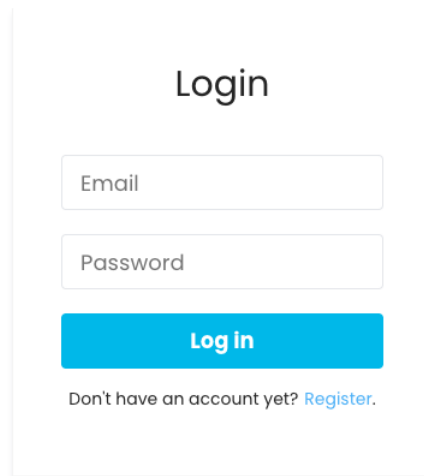
### **3.2.2 Dokumentacja aplikacji**

W tym podrozdziale sporządzono dokumentację aplikację internetową.

#### **3.2.2.1 Widoki aplikacji - Ogólne**

## Logowanie

Logujemy się podając email oraz hasło, następnie klikamy przycisk login.

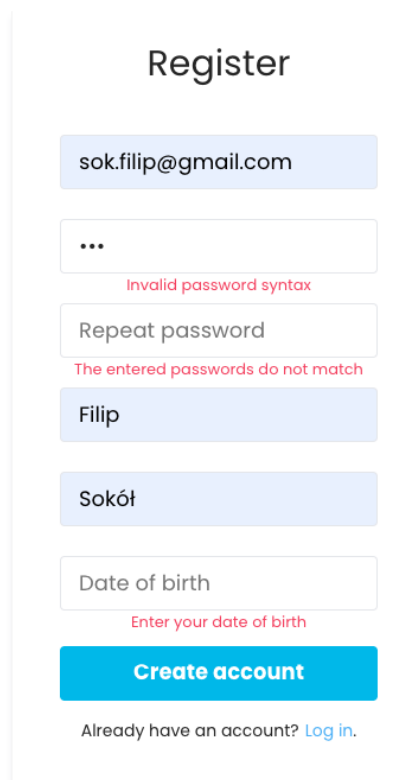


A login form with a white background and a light gray border. At the top, the word "Login" is centered in a dark gray font. Below it are two input fields: "Email" and "Password", both with light gray borders. Under the "Password" field is a blue button with the text "Log in" in white. At the bottom, there is a link that says "Don't have an account yet? [Register.](#)" in a small, dark gray font.

Zdjęcie 1 Widok logowania

## Rejestracja

Rejestracji dokonujemy poprzez wprowadzenie danych, które będą zgadzać z wymogami walidacji. W przeciwnym przypadku użytkownik zostaje poinformowany o wprowadzeniu niepoprawnych danych.



A registration form with a white background and a light gray border. At the top, the word "Register" is centered in a dark gray font. Below it are several input fields and buttons. The first field contains the email "sok.filip@gmail.com". The second field contains three dots "...", with a red error message "Invalid password syntax" below it. The third field is labeled "Repeat password" and has a red error message "The entered passwords do not match" below it. Below these are two more fields: "Filip" and "Sokół". The next field is labeled "Date of birth" and has a red error message "Enter your date of birth" below it. At the bottom is a blue button with the text "Create account" in white. Below the button is a link that says "Already have an account? [Log in.](#)" in a small, dark gray font.

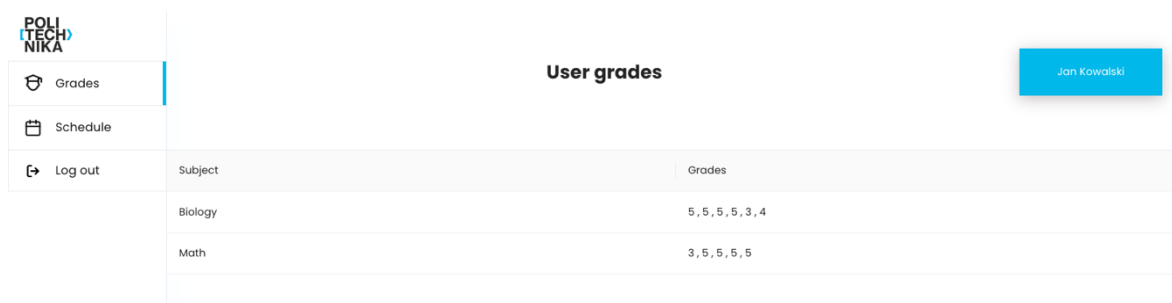
Zdjęcie 2 Widok rejestracji



### 3.2.2.2 Widoki aplikacji - Student

#### Panel ocen studenta

Po zalogowaniu się w aplikacji użytkownik z rolą student zostaje przeniesiony do panelu ocen studenta. Użytkownik może dowiedzieć się w nim na jakie konto jest zalogowany oraz zobaczyć wszystkie oceny, które dostał z poszczególnych przedmiotów.

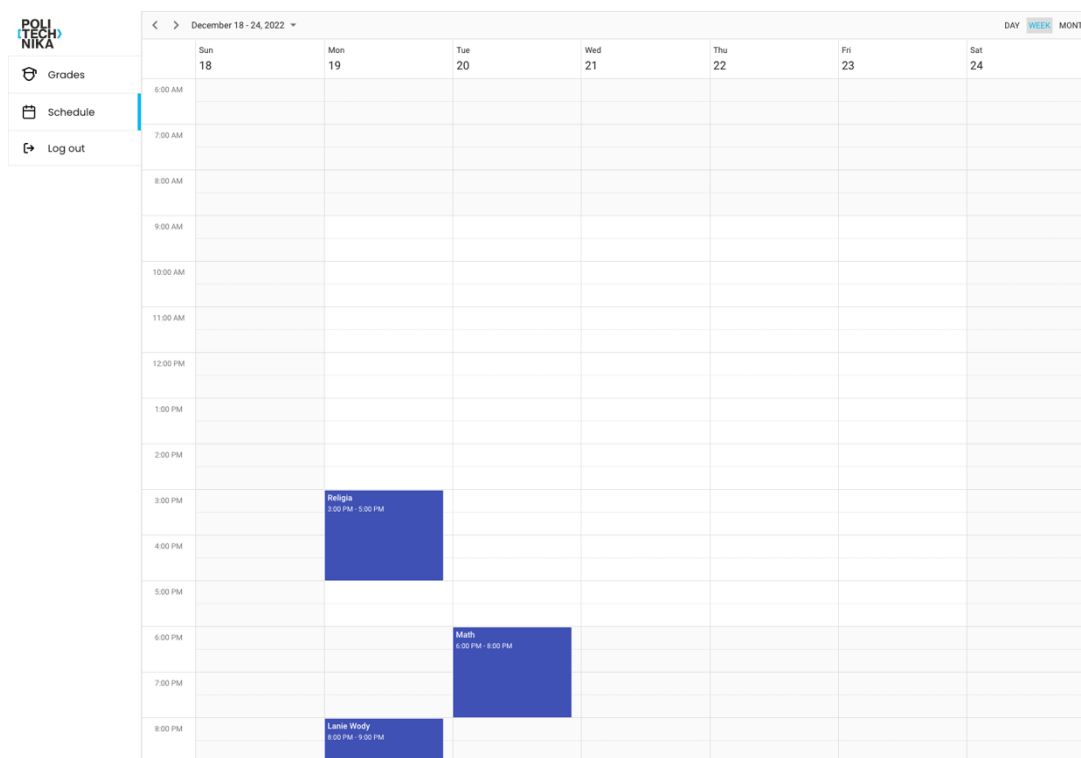


Subject	Grades
Biology	5, 5, 5, 5, 3, 4
Math	3, 5, 5, 5, 5

Zdjęcie 3 Widok panelu ocen

#### Panel planu zajęć

Użytkownik klikając w przycisk Schedule przeniesiony zostaje do panelu zajęć studenta. Może zobaczyć w nim plan grupy, do której jest został przypisany.



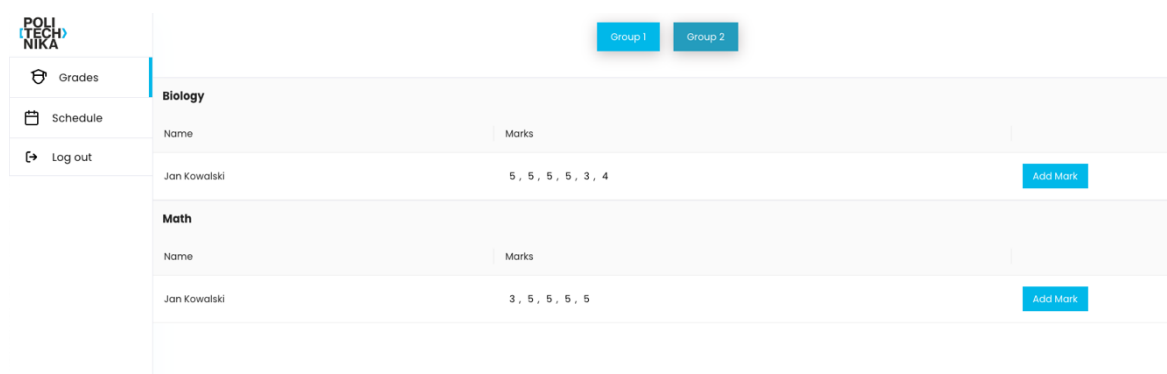
	Sun 18	Mon 19	Tue 20	Wed 21	Thu 22	Fri 23	Sat 24
6:00 AM							
7:00 AM							
8:00 AM							
9:00 AM							
10:00 AM							
11:00 AM							
12:00 PM							
1:00 PM							
2:00 PM							
3:00 PM		Religia 3:00 PM - 5:00 PM					
4:00 PM							
5:00 PM							
6:00 PM				Math 6:00 PM - 8:00 PM			
7:00 PM							
8:00 PM		Lane Wody 8:00 PM - 9:00 PM					

Zdjęcie 4 Widok panelu zajęć

### 3.2.2.3 Widoki aplikacji – Nauczyciel

#### Panel ocen nauczyciela

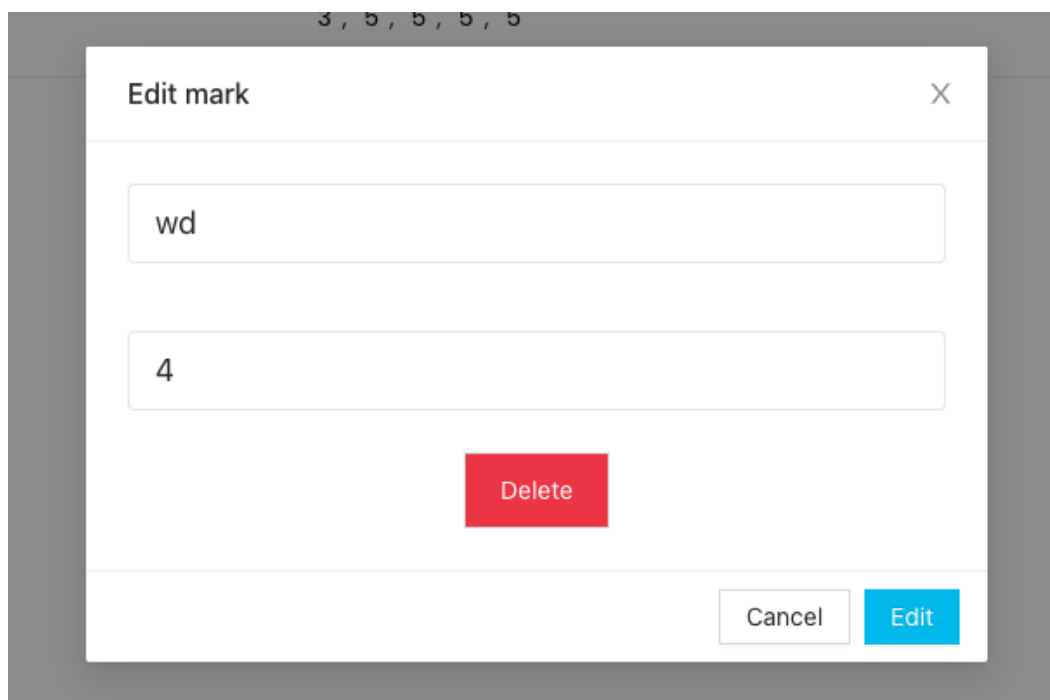
Po zalogowaniu się w aplikacji użytkownik z rolą teacher zostaje przeniesiony do panelu ocen studentów w poszczególnej grupie. U góry posiada on sekcje nawigacyjną dzięki której może przemieszczać się po grupach, w których uczy.



Zdjęcie 5 Widok panelu ocen nauczyciela

#### Widok okna edycji oceny

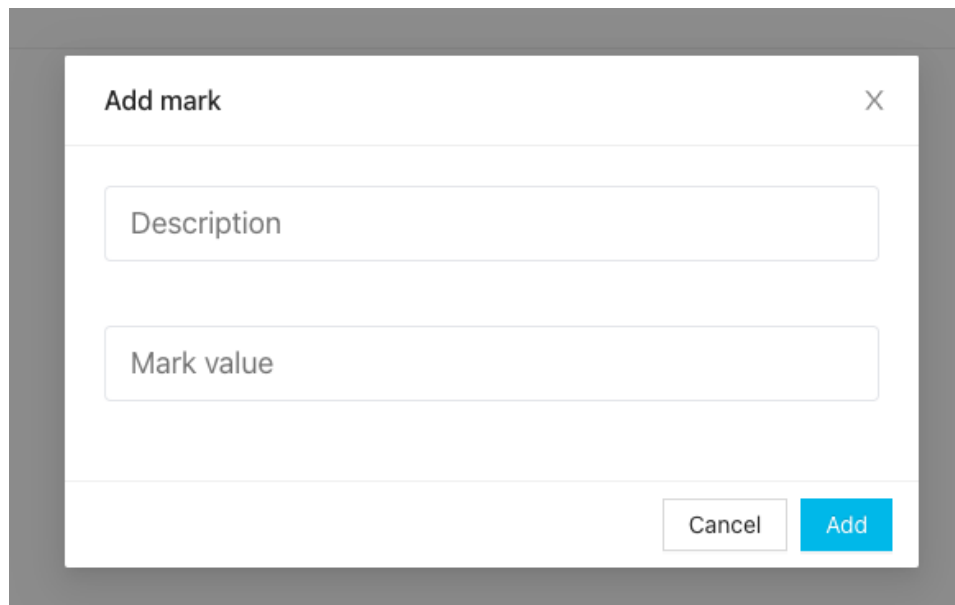
Nauczyciel może kliknąć na daną ocenę przez co otworzy wyskakujące okno z jej szczegółami. Posiada on tam dostęp do edycji danych oraz usunięcia całkowicie oceny. Podane pola nie mogą być puste oraz ocena musi znajdować się w zakresie 2-5, aby możliwe było potwierdzenie edycji.



Zdjęcie 6 Widok okna edycji oceny

### Widok okna tworzenia oceny

Nauczyciel może kliknąć na niebieski przycisk obok studenta, aby dodać do jego puli nową ocenę. Tak samo jak w przypadku edycji pola zostają walidowane i muszą one zostać konkretnie wypełnione, aby umożliwione zostało utworzenie oceny.

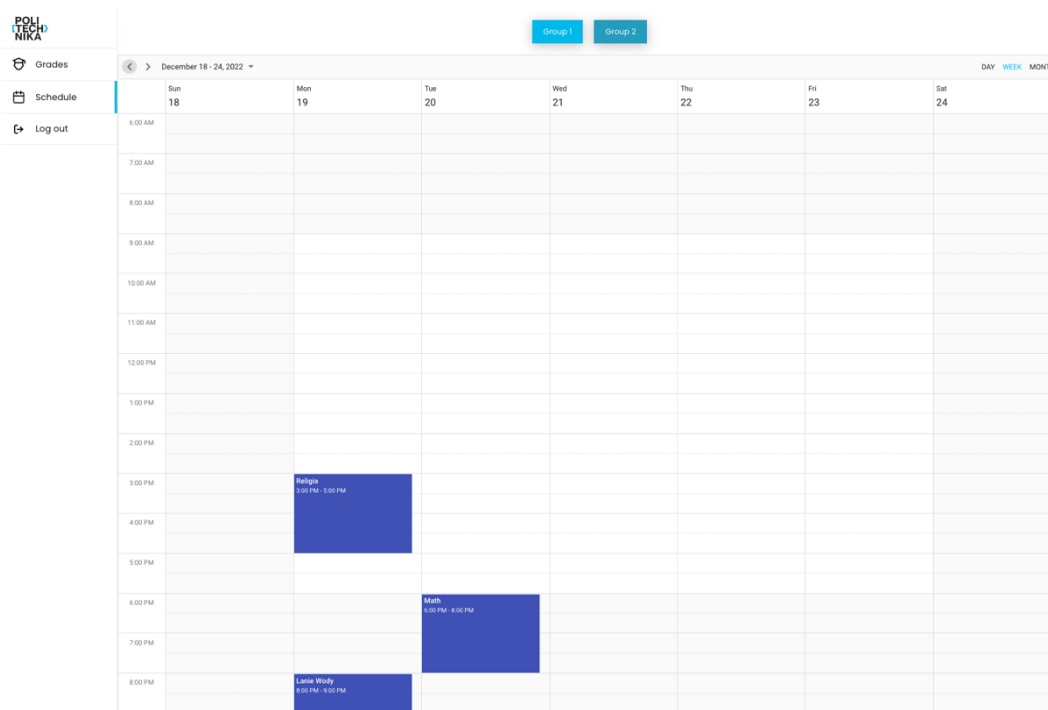


The image shows a modal window titled "Add mark". It contains two text input fields: "Description" and "Mark value". At the bottom right, there are two buttons: "Cancel" and "Add".

Zdjęcie 7 Widok okna tworzenia oceny

### Widok planu grup

Przechodząc do podstrony Schedule nauczyciel może zobaczyć plany wszystkich grupy, w których uczy. Korzystając z panelu nawigacji może zmieniać prezentowane dane.



The image shows a weekly schedule view for the week of December 18-24, 2022. The left sidebar contains 'Grades', 'Schedule', and 'Log out'. The top navigation bar shows 'Group 1' and 'Group 2'. The schedule grid shows the following classes:

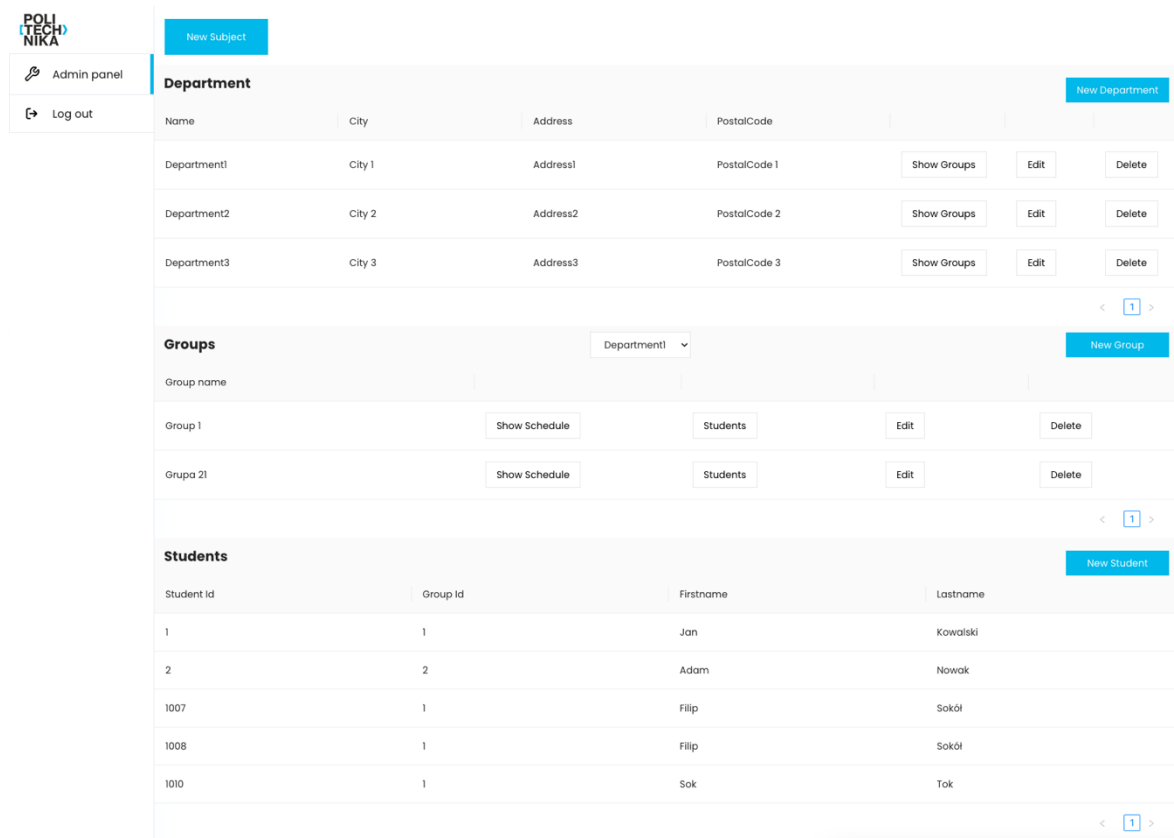
Time	Sun 18	Mon 19	Tue 20	Wed 21	Thu 22	Fri 23	Sat 24
6:00 AM							
7:00 AM							
8:00 AM							
9:00 AM							
10:00 AM							
11:00 AM							
12:00 PM							
1:00 PM							
2:00 PM							
3:00 PM		Religia 3:00 PM - 5:00 PM					
4:00 PM							
5:00 PM							
6:00 PM			Math 6:00 PM - 8:00 PM				
7:00 PM							
8:00 PM		Lecture Work 8:00 PM - 9:00 PM					

Zdjęcie 8 Widok planu grup

### 3.2.2.4 Widoki aplikacji – Admin

#### Widok panel admina

Użytkownik z rolą admin posiada dostęp tylko jednej strony, do której zostaje przeniesiony od razu po zalogowaniu. Umożliwione zostało mu tam kontrolowanie większości danych poprzez wbudowane specjalne funkcjonalności.



The screenshot displays the Admin Panel interface for POLITECHNIKA. The left sidebar contains links for 'Admin panel' and 'Log out'. The main content area is divided into three sections: 'Department', 'Groups', and 'Students'.

**Department Section:** Features a 'New Subject' button at the top left and a 'New Department' button at the top right. The table lists departments with columns: Name, City, Address, PostalCode, Show Groups, Edit, and Delete.

Name	City	Address	PostalCode	Show Groups	Edit	Delete
Department1	City 1	Address1	PostalCode 1	Show Groups	Edit	Delete
Department2	City 2	Address2	PostalCode 2	Show Groups	Edit	Delete
Department3	City 3	Address3	PostalCode 3	Show Groups	Edit	Delete

**Groups Section:** Includes a 'New Group' button at the top right. A dropdown menu shows 'Department1'. The table lists groups with columns: Group name, Show Schedule, Students, Edit, and Delete.

Group name	Show Schedule	Students	Edit	Delete
Group 1	Show Schedule	Students	Edit	Delete
Grupa 21	Show Schedule	Students	Edit	Delete

**Students Section:** Features a 'New Student' button at the top right. The table lists students with columns: Student Id, Group Id, Firstname, and Lastname.

Student Id	Group Id	Firstname	Lastname
1	1	Jan	Kowalski
2	2	Adam	Nowak
1007	1	Filip	Sokół
1008	1	Filip	Sokół
1010	1	Sok	Tok

Zdjęcie 9 Widok panelu admina

#### Widok okna tworzenia nowego przedmiotu

Poprzez kliknięcie przycisku „New Subject” użytkownikowi zostaje otwarte wyskakujące okno, w którym może dodać nowy przedmiot. Pola muszą zostać wypełnione, aby umożliwione zostało dodanie go.

**Zdjęcie 10 Widok okna tworzenia nowego przedmiotu**

### **Widok sekcji wydziałów w panelu admina**

W tej sekcji użytkownik może dodawać oraz zarządzać wydziałami.

Department					New Department	
Name	City	Address	PostalCode			
Department1	City 1	Address1	PostalCode 1	Show Groups	Edit	Delete
Department2	City 2	Address2	PostalCode 2	Show Groups	Edit	Delete
Department3	City 3	Address3	PostalCode 3	Show Groups	Edit	Delete

**Zdjęcie 11 Widok sekcji wydziałów w panelu admina**

### **Widok okna tworzenia lub edycji wydziału**

Okna tworzenia oraz edycji wydziału są bliźniaczo do siebie podobne, dlatego zamieszczone zostają w jednym podpunkcie. Użytkownik aby potwierdzić wprowadzone dane musi wypełnić wszystkie pola.

Address3 PostalCode 3

Add new department X

Name

Address

City

PostalCode

Cancel OK

**Zdjęcie 12 Widok okna tworzenia lub edycji wydziału**

### **Widok okna wszystkich grup w wydziale**

Korzystając z przycisku „Show Groups” otwarte zostaje okno wszystkich grup w danym wydziale. Zaimplementowany został tam również przycisk umożliwiający usunięcie grupy.

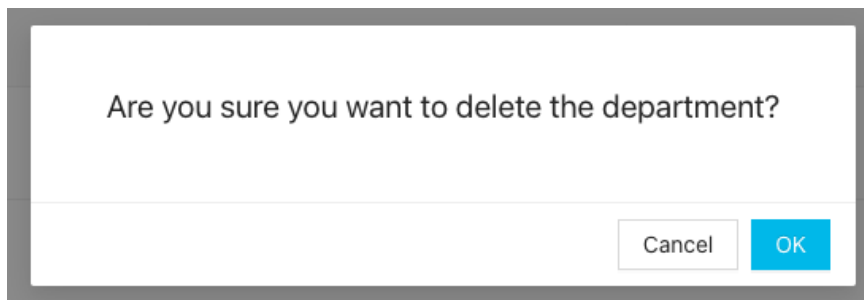
Groups assigned to a department X

Id	Name	
1	Group 1	Delete
2	Grupa 21	Delete

**Zdjęcie 13 Widok okna wszystkich grup w wydziale**

### **Widok okna usuwania wydziału**

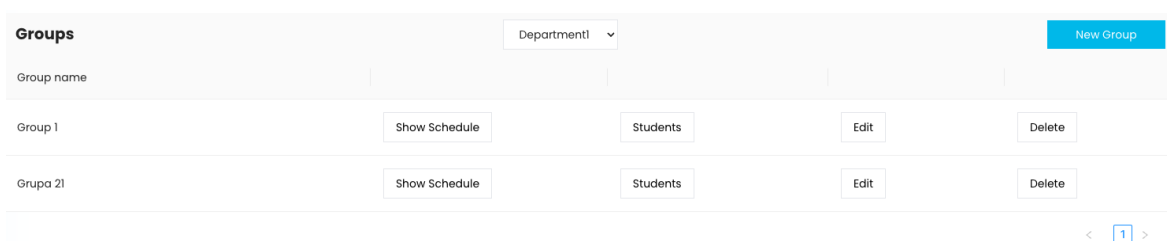
Użytkownik może również usunąć wydział klikając w odpowiedni przycisk obok jego danych. Otwarte zostaje wtedy okno służące do upewnienia użytkownika czy chce dokonać zmian.



**Zdjęcie 14 Widok okna usuwania wydziału**

### **Widok sekcji grup w panelu admina**

Poprzez pole wyboru pośrodku sekcji użytkownik może wybrać prezentowane grupy w konkretnym wydziale.



**Zdjęcie 15 Widok sekcji grup w panelu admina**

### **Widok okna tworzenia nowej grupy**

Użytkownik poprzez kliknięcie „New Group” wywołuje okno, w którym może wpisać nazwę nowej grupy w danym wydziale. Pole jest walidowane.



**Zdjęcie 16 Widok okna tworzenia nowej grupy**

### **Widok okna edycji grupy**

W panelu edycji grupy może on zmienić dane odnośnie jej wprowadzając nową nazwę lub przypisać ją do innego wydziału.

Dialog box titled "Edit group" with a close button (X). It contains two input fields. The first field contains the text "Group 1". The second field contains the number "1". At the bottom right, there are two buttons: "Cancel" and "Edit".

**Zdjęcie 17 Widok okna edycji grupy**

### **Widok okna studentów w grupie**

Klikając przycisk „Student” użytkownikowi zostaje otwarte okno ze wszystkimi studentami w danej grupie. Posiada on tam możliwość usuwania poszczególnych studentów.

Dialog box titled "All students in group" with a close button (X). It displays a table of students in the group. Each row includes a "Delete" button.

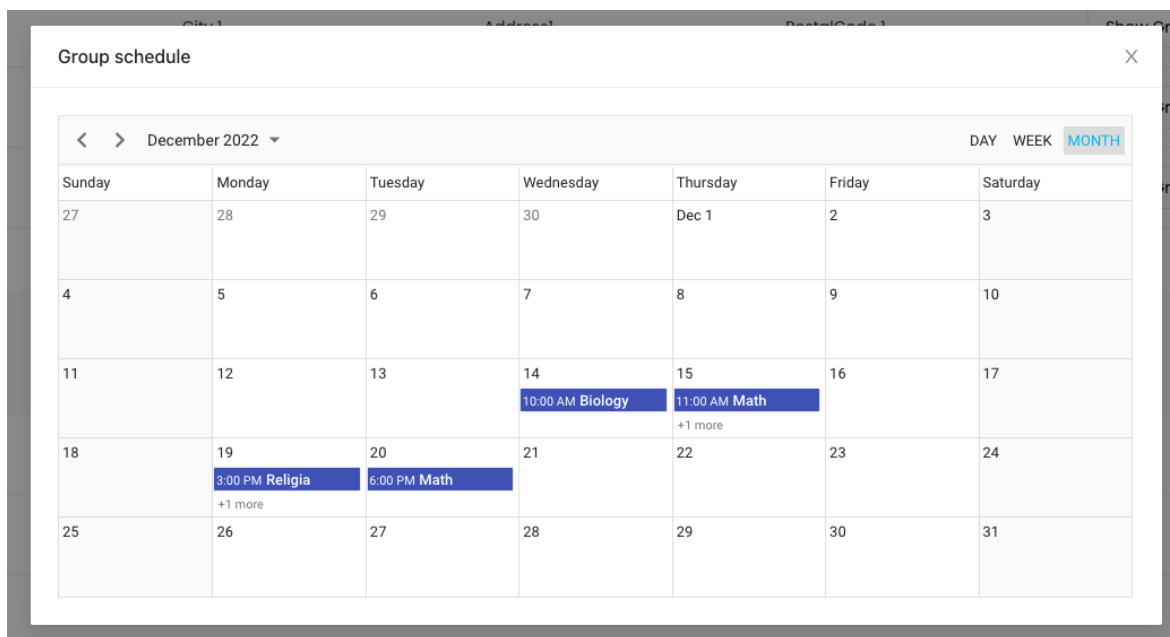
Student Id	First name	Last name	Email	
1	Jan	Kowalski	jan@kowalski.com	Delete
1007	Filip	Sokół	sok.ddfilip@gmail.com	Delete
1008	Filip	Sokół	sok.filipp@gmail.com	Delete
1010	Sok	Tok	studentTest@gmail.com	Delete

**Zdjęcie 18 Widok okna studentów w grupie**



### Widok planu zajęć danej grupy

Umożliwione zostało również otwarcie okna z planem grupy poprzez przycisk „Schedule”.



**Zdjęcie 19 Widok planu zajęć danej grupy**

### Widok sekcji wszystkich studentów

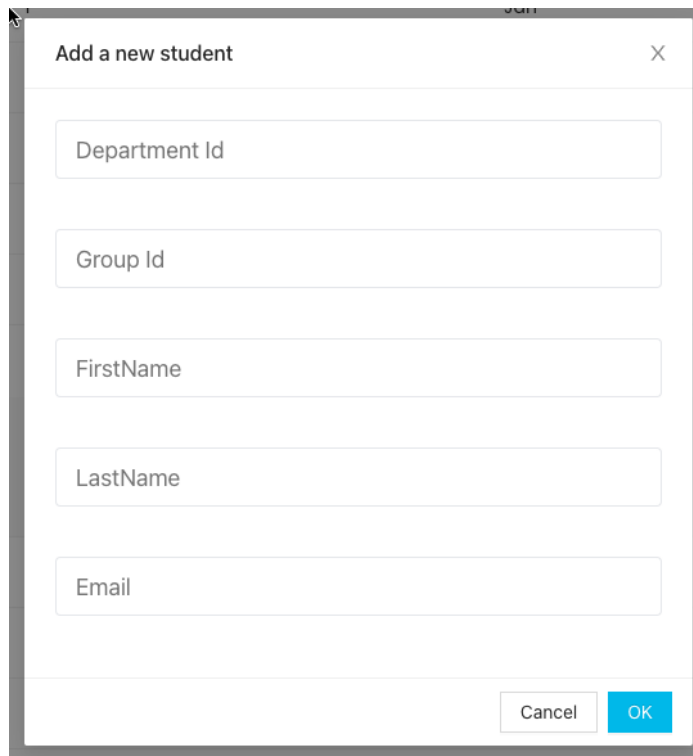
W sekcji Students użytkownikowi prezentowane zostają dane o wszystkich studentach w systemie.

Students				New Student
Student Id	Group Id	Firstname	Lastname	
1	1	Jan	Kowalski	
2	2	Adam	Nowak	
1007	1	Filip	Sokół	
1008	1	Filip	Sokół	
1010	1	Sok	Tok	
				< 1 >

**Zdjęcie 20 Widok sekcji wszystkich studentów**

### Widok okna tworzenia nowego studenta

Klikając w przycisk „New Student” użytkownik może utworzyć nowego studenta. Wymagane jest wprowadzenie wszystkich danych, aby potwierdzić akcję.



Dialog box titled "Add a new student" with a close button (X) in the top right corner. The dialog contains five input fields for student information: Department Id, Group Id, FirstName, LastName, and Email. At the bottom right, there are two buttons: "Cancel" and "OK".

**Zdjęcie 21 Widok okna tworzenia nowego studenta**

### **Widok sekcji wszystkich nauczycieli**

Ostatnim elementem panelu admina jest sekcja wszystkich nauczycieli.

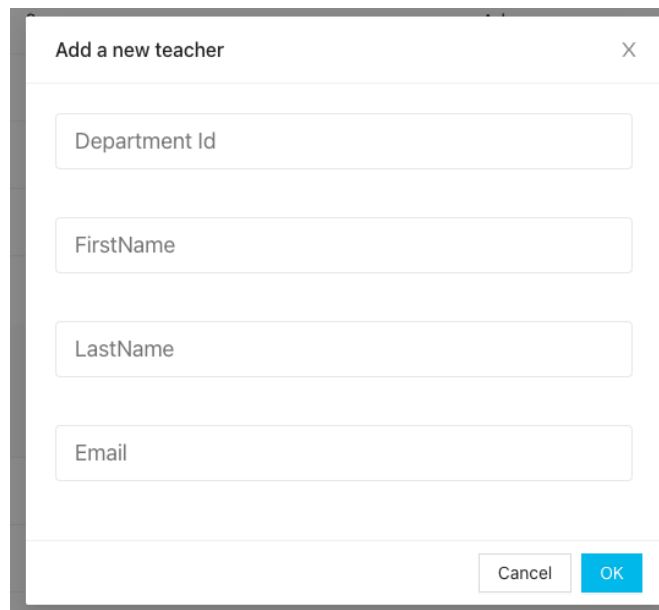
Teachers				New Teacher
Teacher Id	Department Id	Firstname	Lastname	
1	1	Jan	Kowalski	
2	1	Adam	Nowak	
3	1	Michał	Monitor	
4	1	Filip	Sokół	
5	1	Filip	Doku	
6	1	Piotr	Dokud	
7	1	Tomasz	Kot	
8	1	Filip	Sokół	

**Zdjęcie 22 Widok sekcji wszystkich nauczycieli**

### **Widok okna tworzenia nowego nauczyciela**

Klikając w przycisk „New Teacher” użytkownik może utworzyć nowego nauczyciela.

Wymagane jest wprowadzenie wszystkich danych, aby potwierdzić akcję.



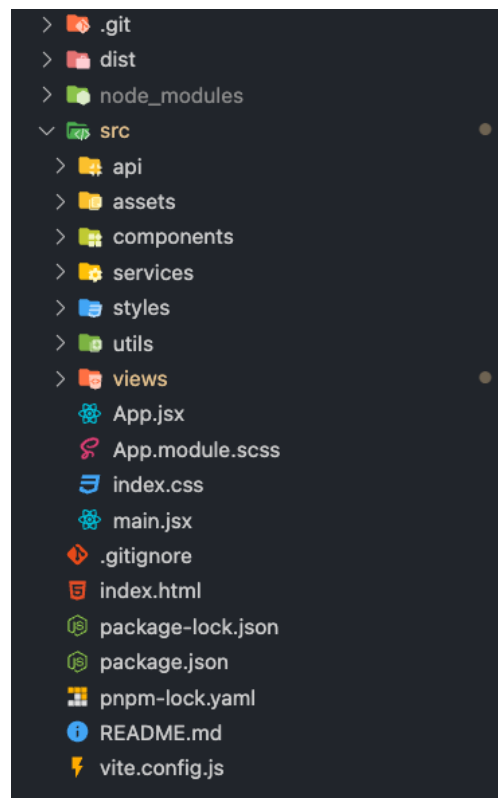
The image shows a modal dialog box titled "Add a new teacher" with a close button (X) in the top right corner. Inside the dialog, there are four text input fields stacked vertically, labeled "Department Id", "FirstName", "LastName", and "Email". At the bottom right of the dialog, there are two buttons: "Cancel" and "OK".

**Zdjęcie 23** Widok okna tworzenia nowego nauczyciela

### 3.2.3 Dokumentacja techniczna

Do procesu tworzenia aplikacji został wykorzystany edytor Visual Studio Code.

#### 3.2.3.1 Nawigacja w projekcie



**Zdjęcie 24** Struktura projektu aplikacji internetowej

Projekt zbudowany jest na bundlerze Vite, który jest nowoczesnym zamiennikiem Webpacka.

Wszystkie widoki główne znajdują się w folderze views. Komponenty użyte w nich znajdują się natomiast w osobnym folderze components. Nawigacja po stronie jest możliwa dzięki logice zawartej w pliku Layout.jsx. Oprócz wcześniej wymienionych lokalizacji w projekcie znajdują się również foldery dla stylizacji, ważnych funkcji, serwisów oraz grafiki.

Projekt uruchamiamy poleceniem „pnpm run dev”.

### **3.3 Aplikacja Mobilna**

#### **3.3.1 Wykorzystane technologie**

Technologie po stronie aplikacji Mobilnej.

**React Native** - to biblioteka JavaScript stworzona przez firmę Facebook, która pozwala tworzyć aplikacje mobilne dla systemów iOS i Android. Używa ona biblioteki React do tworzenia interfejsu użytkownika. Aplikacje stworzone za pomocą React Native są natywne dla urządzenia, co oznacza, że działają tak samo jak aplikacje napisane w językach natywnych dla danego systemu operacyjnego, takich jak Swift/Objective-c dla iOS lub Kotlin/Java dla Android.

**Expo** - to narzędzie open source, które ułatwia tworzenie aplikacji mobilnych z wykorzystaniem frameworka React Native. Jest to zestaw narzędzi, bibliotek i serwisów, które umożliwiają tworzenie aplikacji dla systemów iOS i Android bez konieczności instalowania wielu dodatkowych narzędzi lub ustawiania środowiska programistycznego.

Główną zaletą korzystania z Expo jest to, że pozwala on na szybkie i łatwe uruchomienie aplikacji na urządzeniach mobilnych bez konieczności kompilowania kodu do natywnych aplikacji. Zamiast tego, aplikacja jest uruchamiana za pomocą specjalnej aplikacji Expo, która pozwala na testowanie aplikacji na urządzeniach mobilnych bez konieczności instalowania jej w sklepach aplikacji.

**Axios** - biblioteka JavaScript służąca do wysyłania i odbierania danych z serwerów przez

protokół HTTP. Może być używana zarówno w aplikacjach internetowych, jak i w aplikacjach mobilnych. Axios umożliwia wysyłanie zapytań HTTP za pomocą metod takich jak GET, POST, PUT, DELETE itp., a także umożliwia konfigurację zapytań za pomocą opcji takich jak nagłówki, ciasteczka, autentykacja itp. Axios automatycznie parsuje odpowiedzi z serwera do formatu JSON, co umożliwia łatwe ich wykorzystanie w aplikacji.

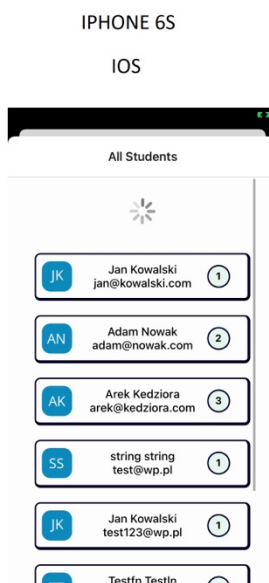
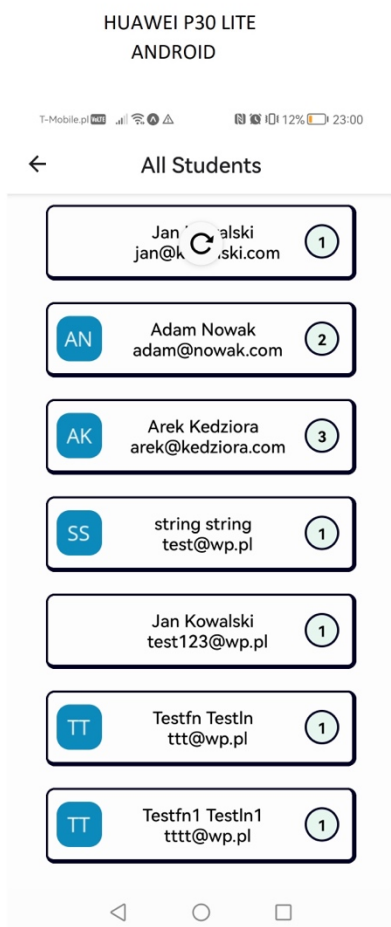
**React Hook Form** - React-hook-form to biblioteka stworzona z myślą o ułatwieniu tworzenia formularzy w aplikacjach React i React Native, pozwala na łatwe i szybkie tworzenie formularzy, dzięki wykorzystaniu hooków w celu zarządzania stanem pól formularza i walidacji danych.

### **3.3.2 Dokumentacja użytkownika**

Aplikacja została przetestowana na prawdziwych urządzeniach.

#### **3.3.2.1 Ekran aplikacji - Ogólne**

Każdy ekran wyświetlający listę elementów może zostać odświeżony poprzez przeciągnięcie ekranu.

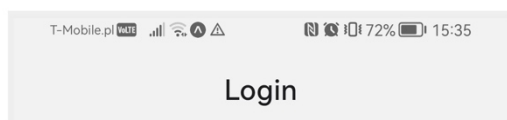


**Zdjęcie 2 Przykład odświeżania ekranu z elementami**

## **Logowanie**

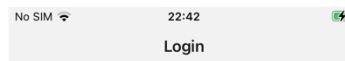
Logujemy się podając email oraz hasło, następnie klikamy przycisk login.

HUAWEI P30 LITE  
ANDROID



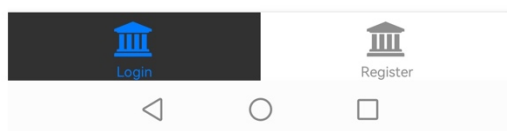
IPHONE 6S

IOS



Login

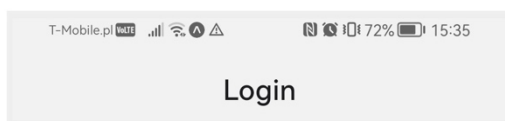
Login



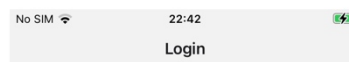
**Zdjęcie 3 Ekran logowania**

W przypadku błędu lub braku danych dostajemy odpowiednie powiadomienie, błąd znika automatycznie po wpisaniu poprawnego maila lub hasła.

HUAWEI P30 LITE  
ANDROID

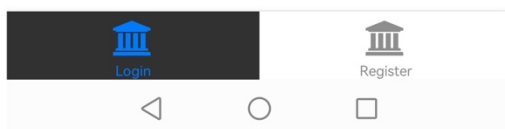


IPHONE 6S  
IOS



Login

Login

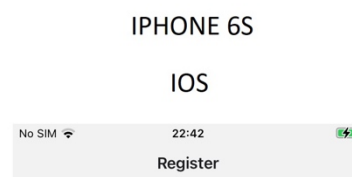
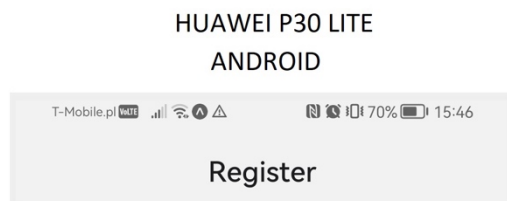


Zdjęcie 4 Ekran logowania - wymagane pola

## Rejestracja

Do przeprowadzenia procesu rejestracji potrzebujemy podać imię, nazwisko, email, hasło, powtórz hasło.





**Register**

First Name

Last Name

Email

Password

Confirm Password

**REGISTER**

Navigate to **login** page

**Register**

First Name

Last Name


Email


Password

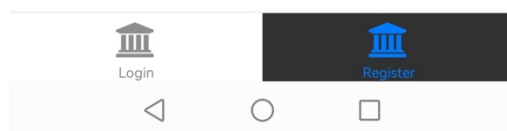
Confirm Password

[Register](#)

Navigate to **login** page

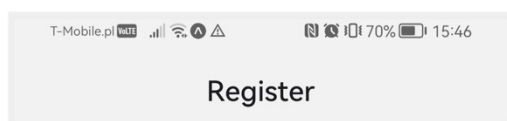
 Login

 Register



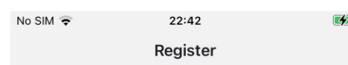
**Zdjęcie 5 Ekran Rejestracji**

HUAWEI P30 LITE  
ANDROID



IPHONE 6S

IOS



Register

First Name  
**First Name is required**

Last Name  
**Last Name is required**

Email  
**Email is required**

Password  
**Password is required**

Confirm Password  
**Password is required**

**REGISTER**

Navigate to **login** page

Register

First Name  
**First Name is required**

Last Name  
**Last Name is required**

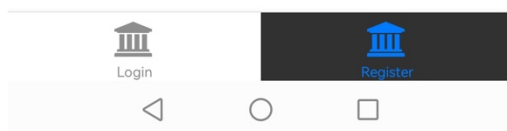
Email  
**Email is required**

Password  
**Password is required**

Confirm Password  
**Password is required**

[Register](#)

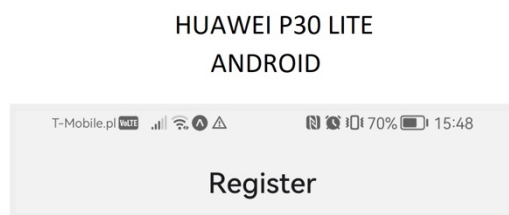
Navigate to **login** page



Zdjęcie 6 Ekran Rejestracji - wymagane pole

Hasło musi zawierać 8 znaków w tym 1 wielką literę oraz 1 cyfrę.

Email musi być wpisany w poprawnej formie.



**Register**

**Account have been created!**

**REGISTER**


Navigate to **login** page


**Register**

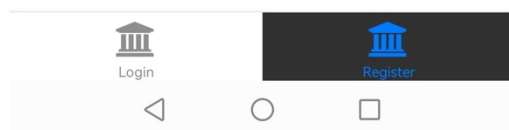
**Account have been created!**

[Register](#)

Navigate to **login** page

 Login

 Register

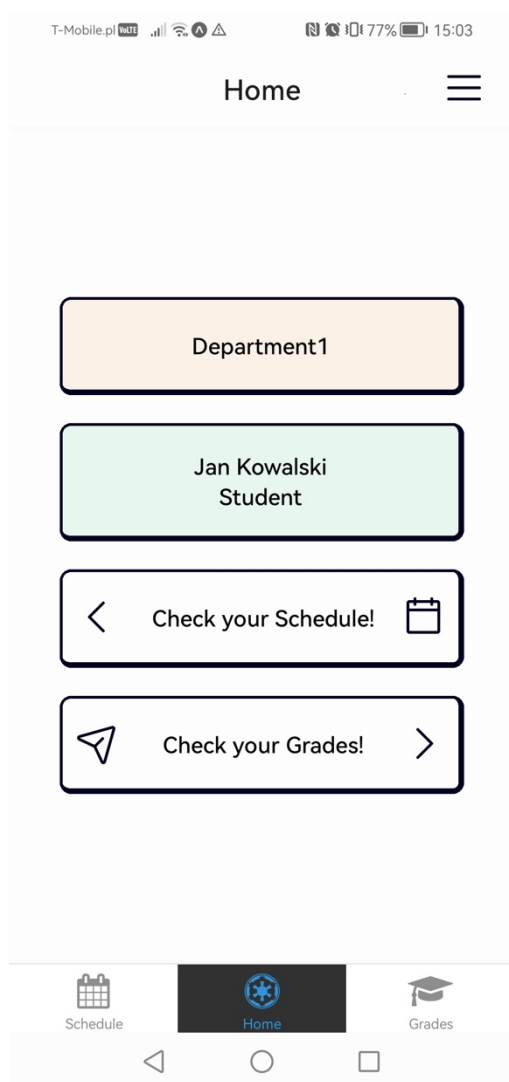


Zdjęcie 7 Poprawnie stworzone konto

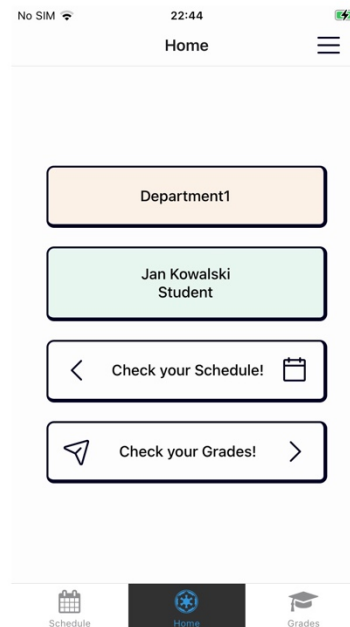
### 3.3.2.2 Ekrany Aplikacji - Student

Ekran główny po zalogowaniu

HUAWEI P30 LITE  
ANDROID



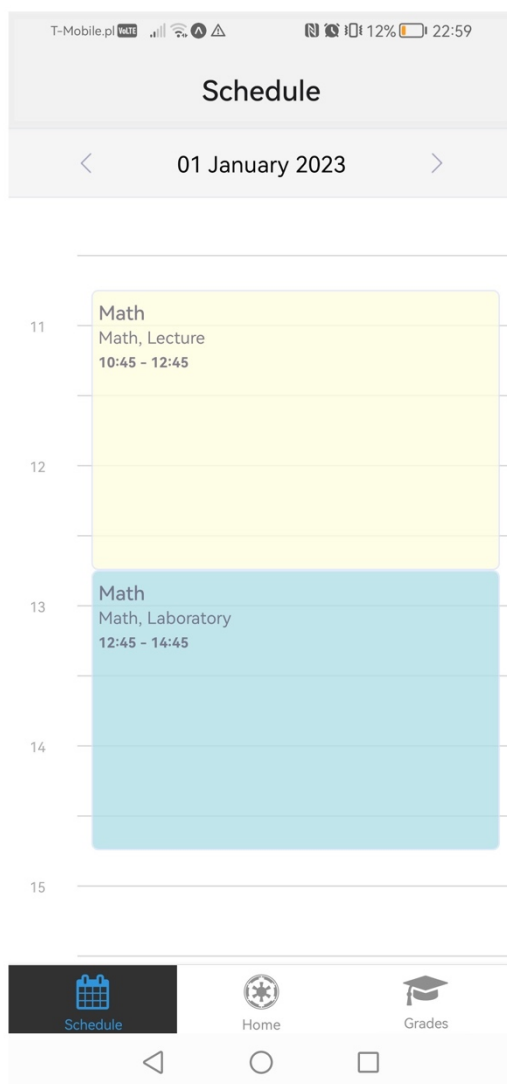
IPHONE 6S  
IOS



**Zdjęcie 8 Ekran główny studenta**

**Plan zajęć**

HUAWEI P30 LITE  
ANDROID



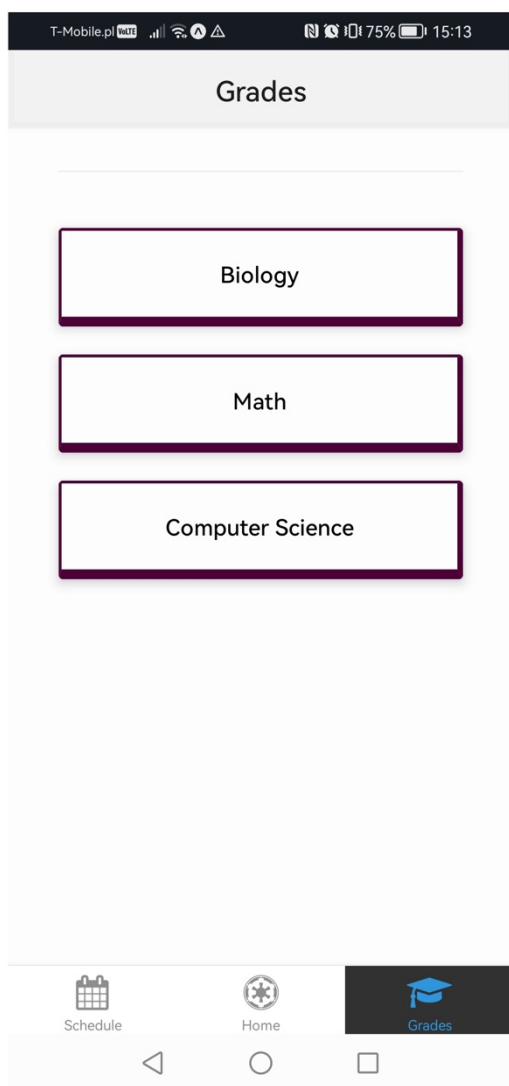
IPHONE 6S  
IOS



**Zdjęcie 9 Plan studenta**

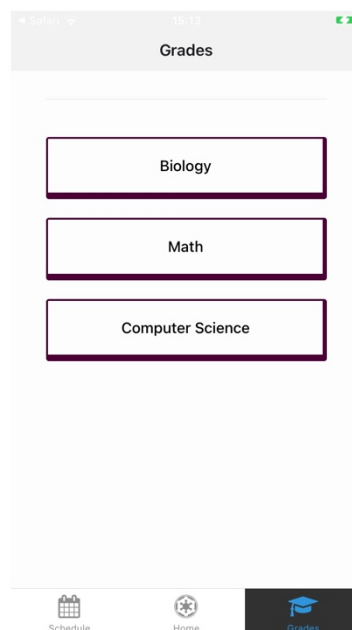
**Przedmioty studenta**

HUAWEI P30 LITE  
ANDROID



IPHONE 6S

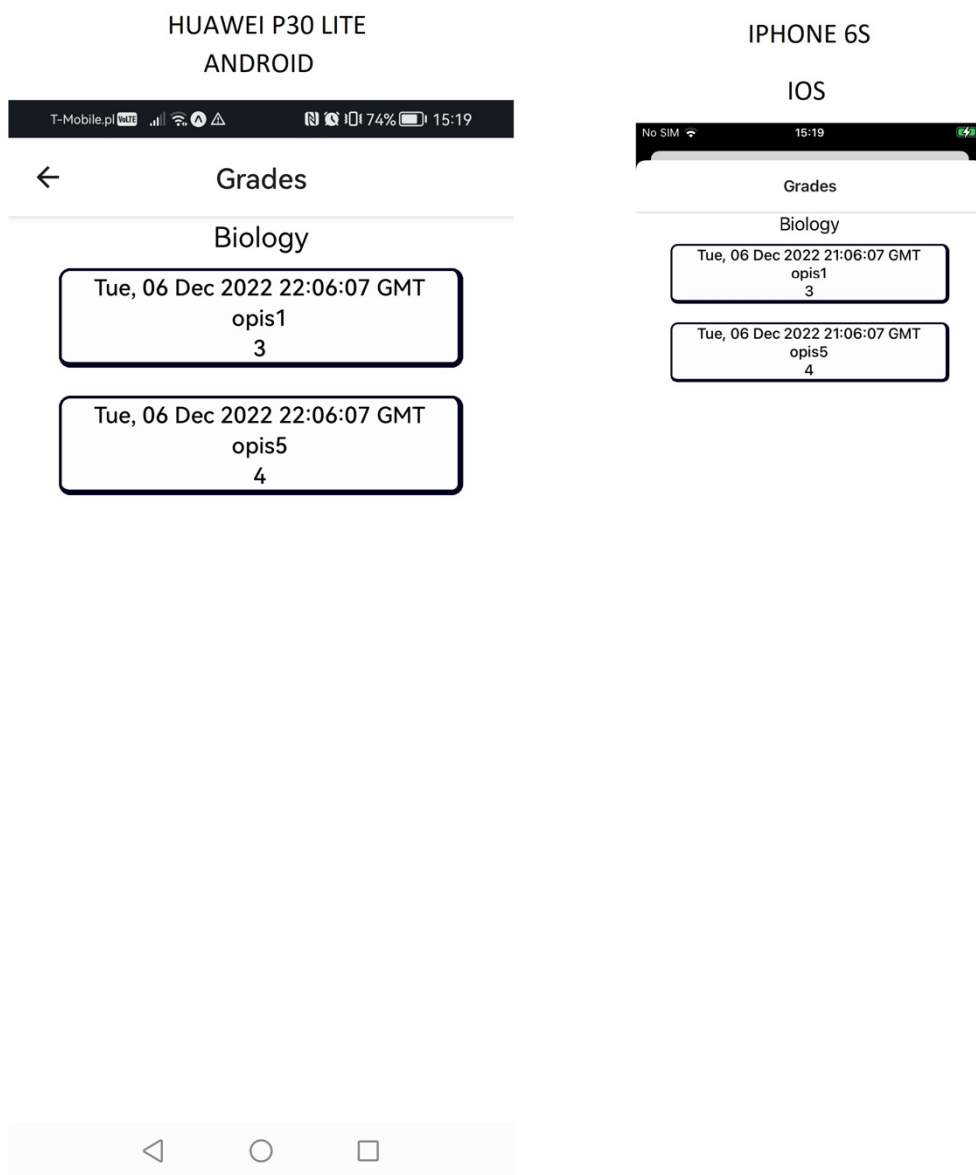
IOS



**Zdjęcie 10 Przedmioty studenta**

## Oceny studenta

Klikając na interesujący nas przedmiot możemy przejść do listy jego ocen.

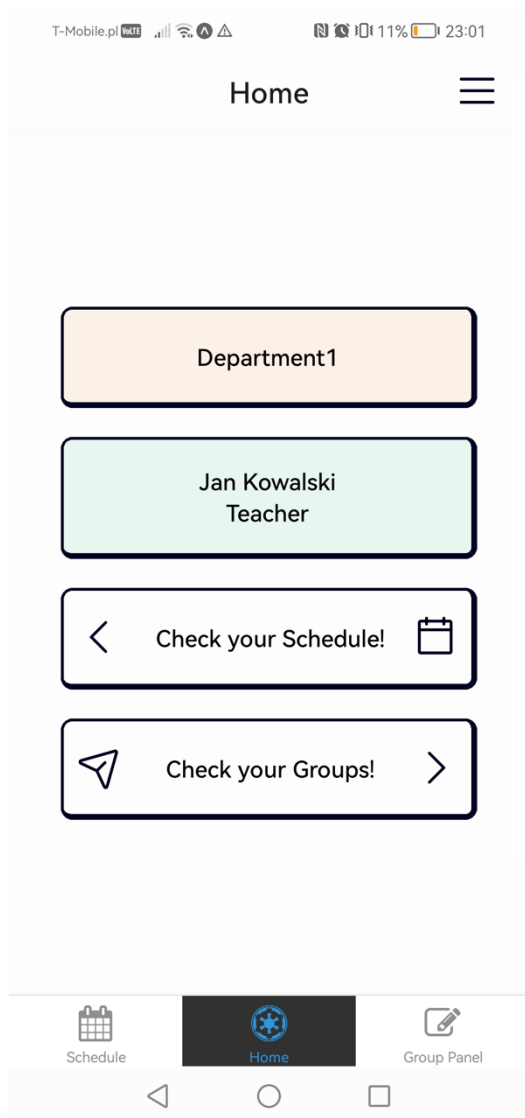


**Zdjęcie 11 oceny studenta**

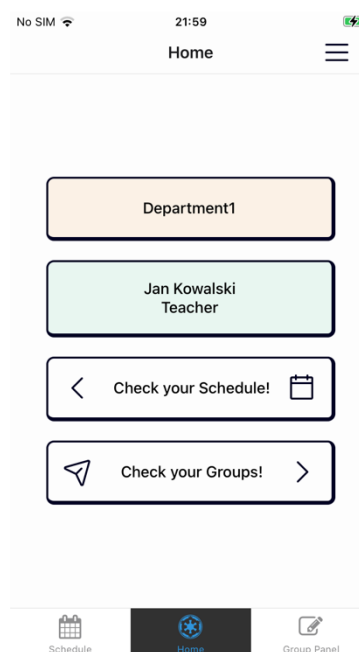
### 3.3.2.3 Ekrany Aplikacji – Nauczyciel

#### Ekran główny po zalogowaniu.

HUAWEI P30 LITE  
ANDROID



IPHONE 6S  
IOS

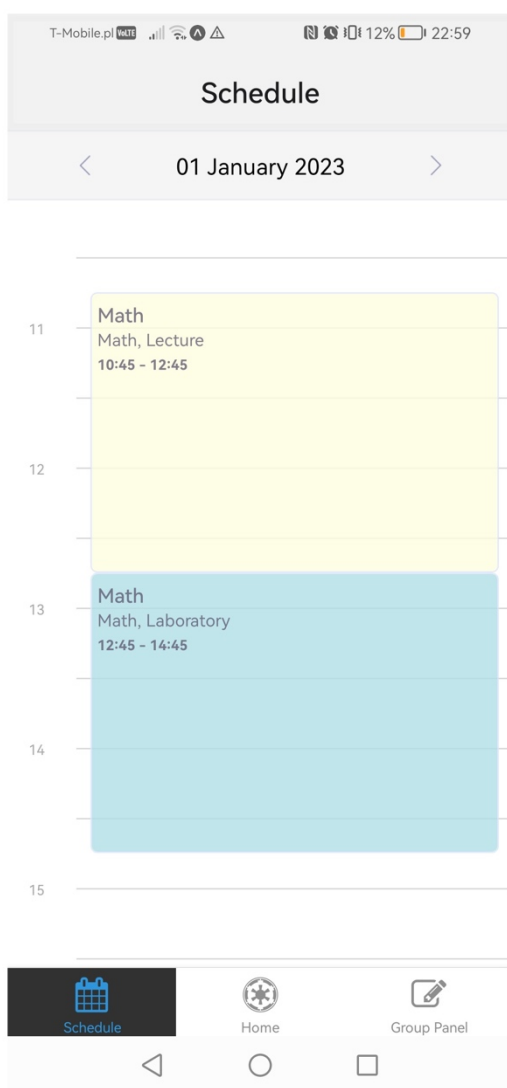


Zdjęcie 12 Ekran główny - Nauczyciel

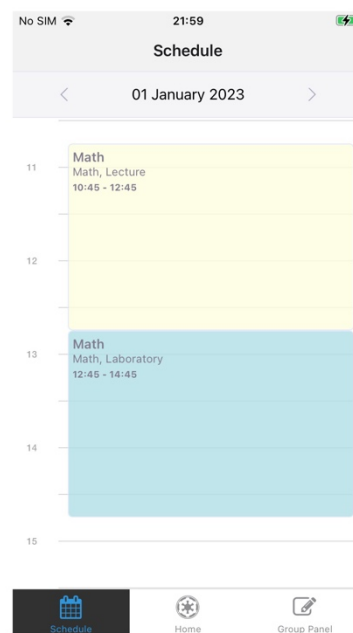


## Plan zajęć

HUAWEI P30 LITE  
ANDROID



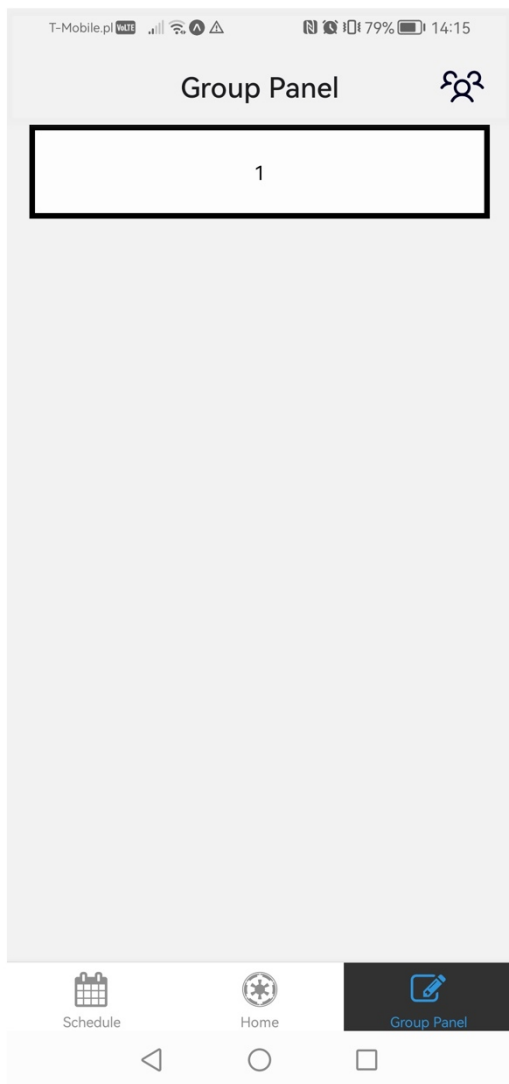
IPHONE 6S  
IOS



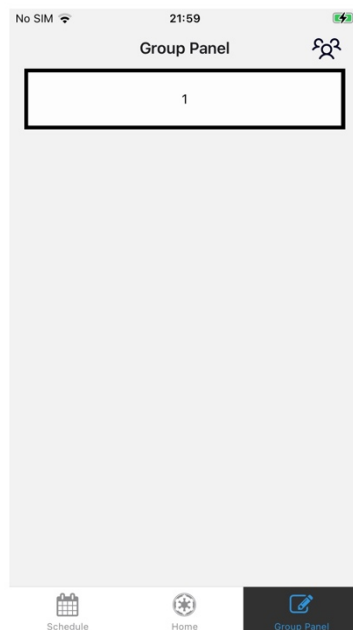
Zdjęcie 13 Plan Zajęć

## Lista grup nauczyciela

HUAWEI P30 LITE  
ANDROID



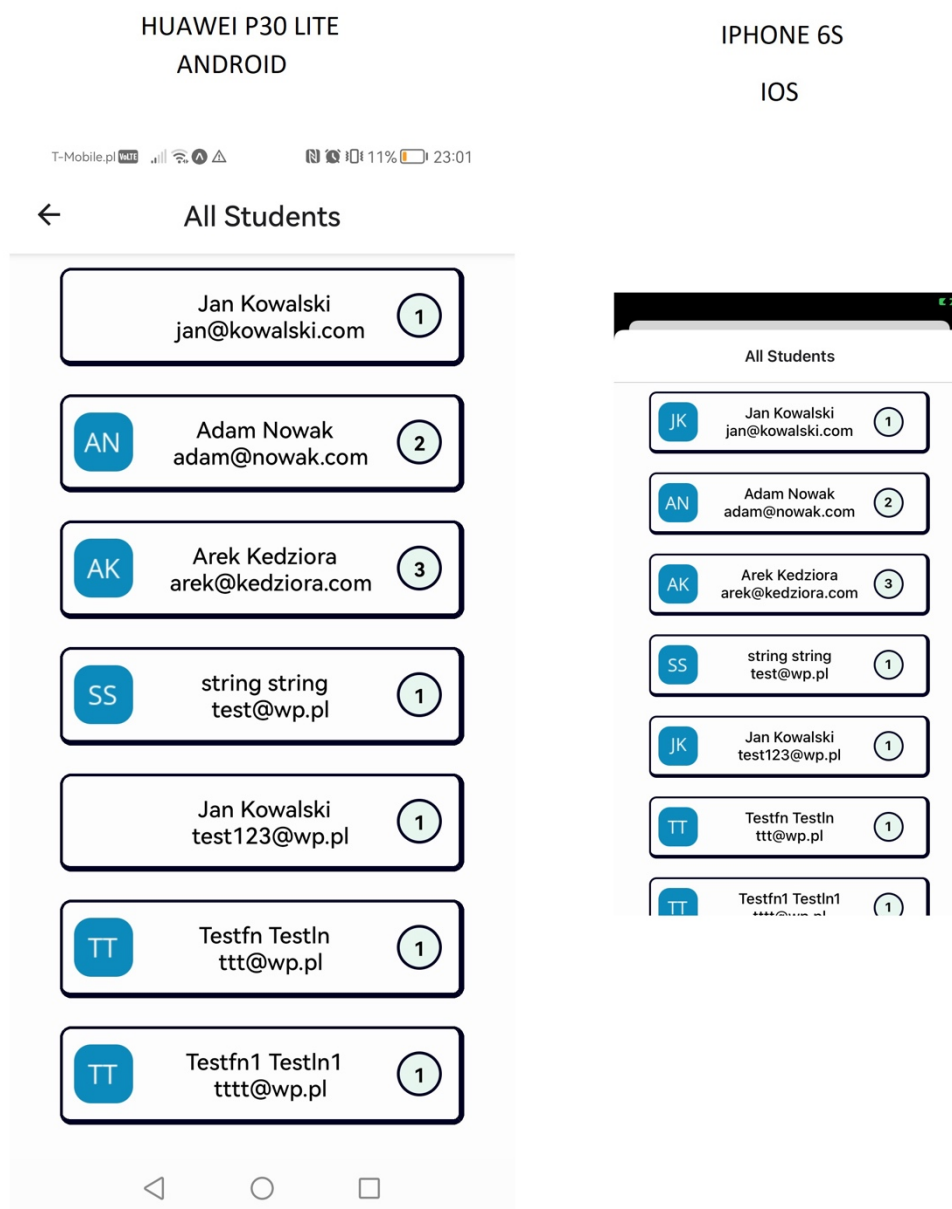
IPHONE 6S  
IOS



Zdjęcie 14 Lista grup nauczyciela

Klikając w prawym górnym rogu ekranu na ikonę możemy przejść do listy wszystkich studentów.

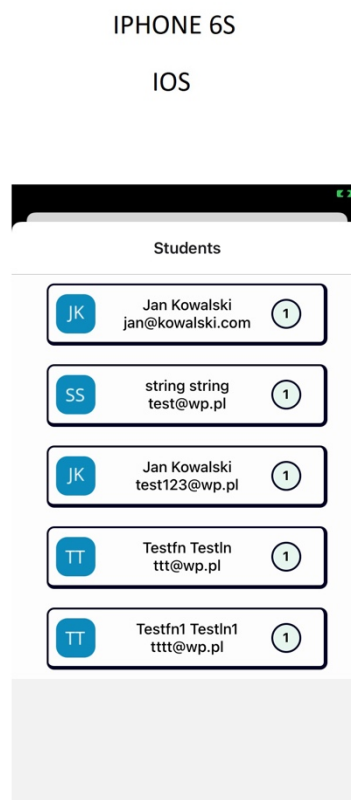
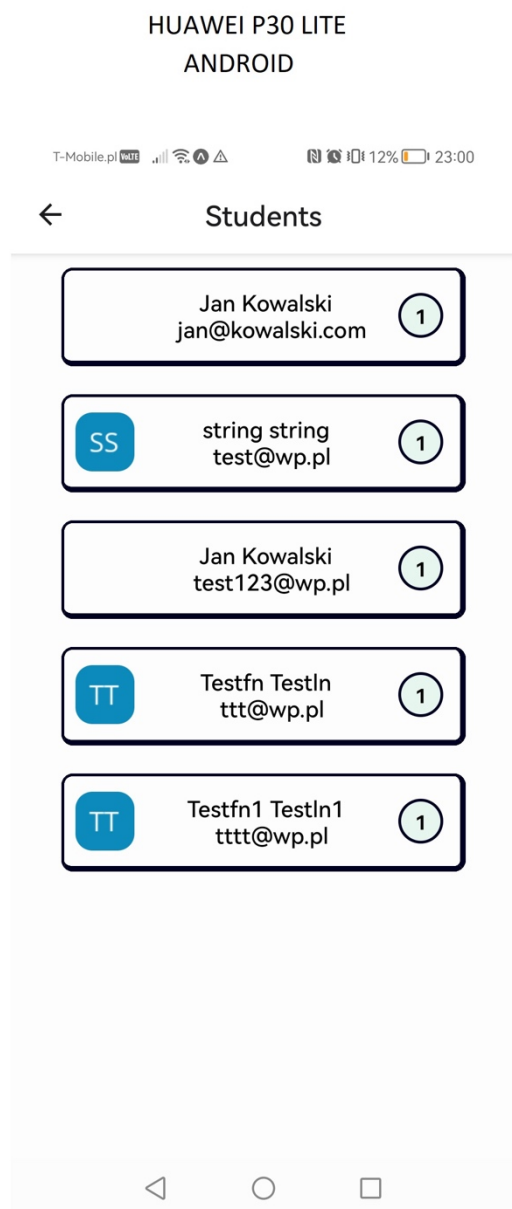
### Lista wszystkich studentów



Zdjęcie 15 Lista wszystkich studentów

## Lista wybranej grupy

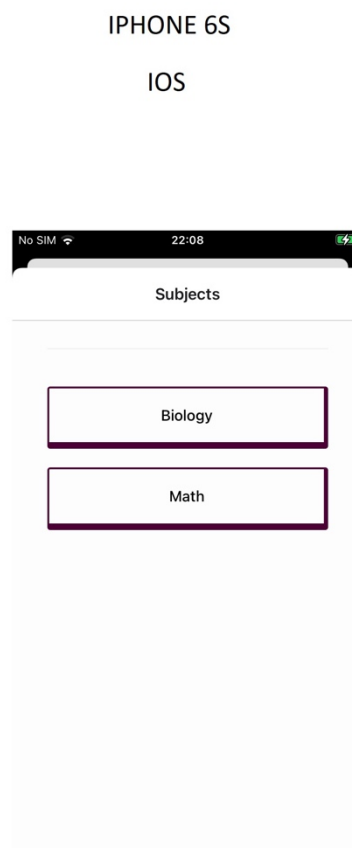
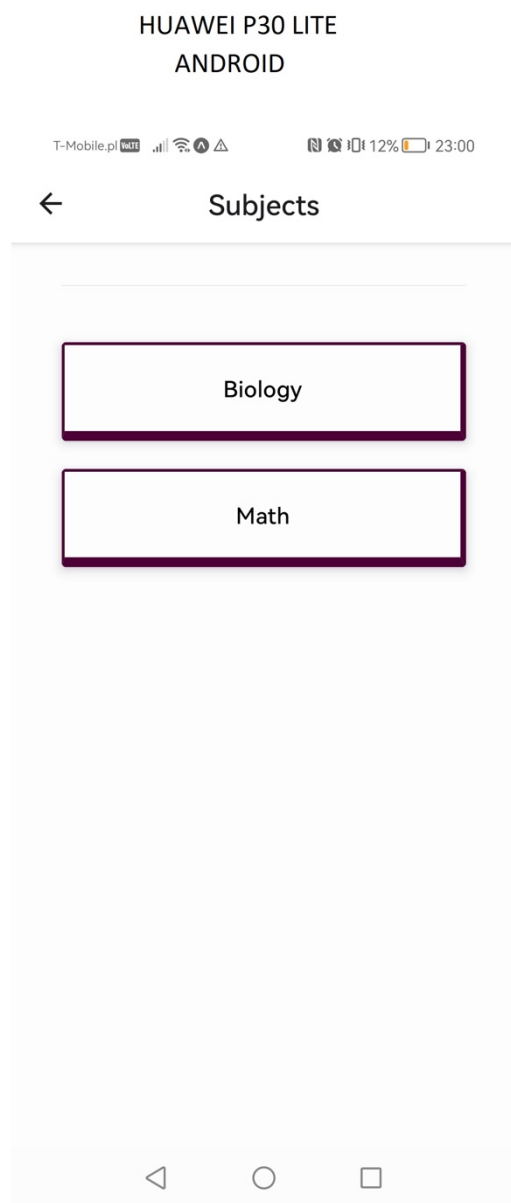
Klikając na daną grupę możemy przejść do listy studentów wybranej grupy.



Zdjęcie 16 Lista wybranej losowo grupy

### Lista przedmiotów studenta

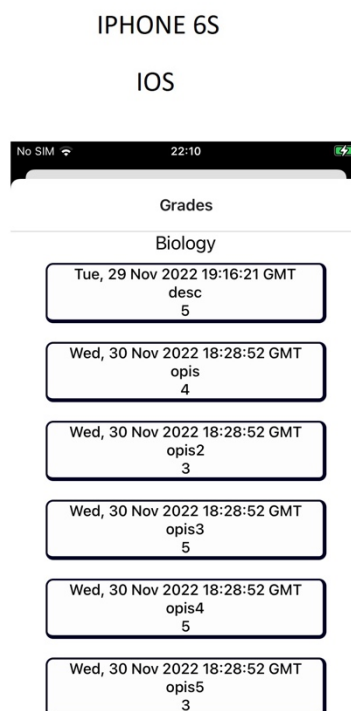
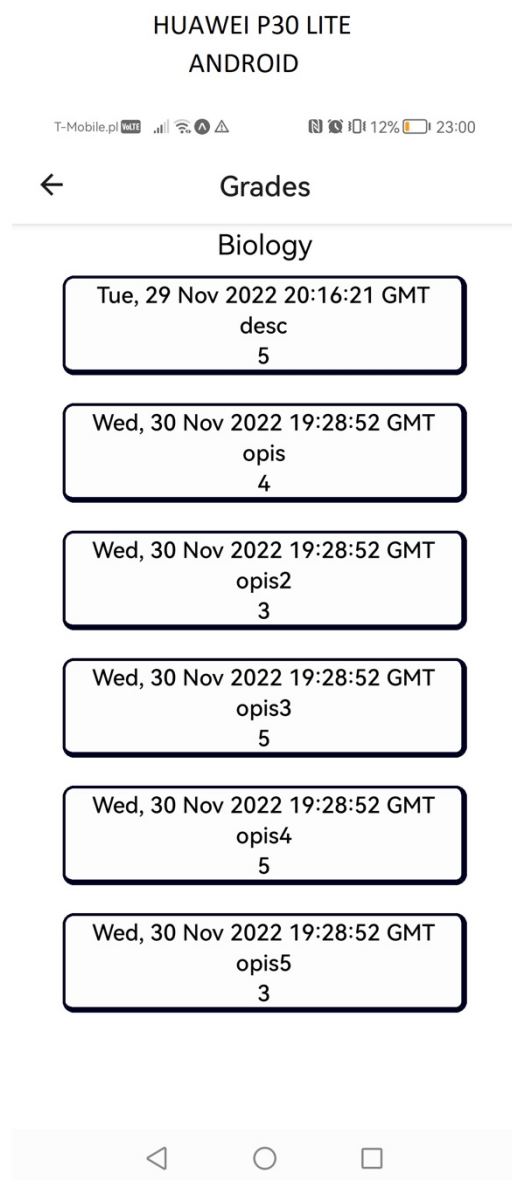
Klikając na profil studenta przechodzimy do listy jego przedmiotów.



**Zdjęcie 17** Lista przedmiotów wybranego studenta

## Oceny wybranego studenta

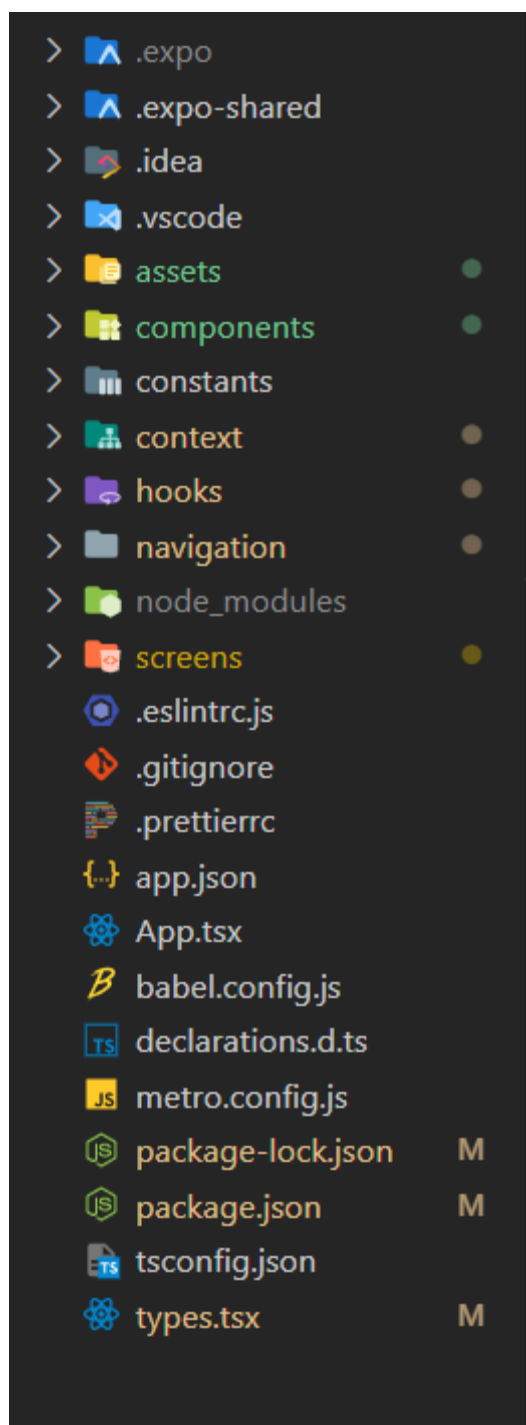
Klikając na nazwę przedmiotu możemy przejść do listy ocen wybranego przedmiotu.



### 3.3.3 Dokumentacja techniczna

Do procesu tworzenia aplikacji został wykorzystany edytor Visual Studio Code.

#### 3.3.3.1 Nawigacja w projekcie



Zdjęcie 19 Struktura projektu

Projekt korzysta z TypeScript'a, cała konfiguracja ustawień znajdziemy w pliku tsconfig.json. Folder assets zawiera wszystkie zdjęcia oraz czcionki używane w projekcie.

Folder components zawiera wszystkie komponenty z których zostały stworzone ekrany.

Wszystkie ekrany znajdują się w folderze screens. Ustawienia nawigatora ekranu znajdują się w folderze navigation. Folder context zawiera całą logikę logowania oraz przechowywania tokenu JWT. Folder hooks to folder w którym znajdziemy wszystkie unikalne hooki ułatwiające budowanie aplikacji. Folder constants zawiera wszystkie ustawienia kolorów oraz stylów. Standardowo cała aplikacja ma rozpoczęcie w pliku app.tsx.

Projekt uruchamiamy poleceniem „npx expo start”.