

**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>	<i>5</i>
3.1.2	<i>Architektura systemu.....</i>	<i>6</i>
3.1.3	<i>Endpointy.....</i>	<i>6</i>
3.1.4	<i>Diagram klas UML.....</i>	<i>12</i>
3.1.5	<i>Model bazy danych .....</i>	<i>13</i>
3.2	FRONTEND .....	14
3.2.1	<i>Wykorzystane technologie.....</i>	<i>14</i>
3.3	APLIKACJA MOBILNA.....	16
3.3.1	<i>Wykorzystane technologie.....</i>	<i>16</i>
3.3.2	<i>Dokumentacja użytkownika .....</i>	<i>17</i>
3.3.2.1	<i>Ekrany Aplikacji - Ogólne .....</i>	<i>17</i>
3.3.2.2	<i>Ekrany Aplikacji - Student.....</i>	<i>23</i>
3.3.2.3	<i>Ekrany Aplikacji – Nauczyciel .....</i>	<i>28</i>
3.3.3	<i>Dokumentacja techniczna.....</i>	<i>35</i>
3.3.3.1	<i>Nawigacja w projekcie.....</i>	<i>35</i>

---

# 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.

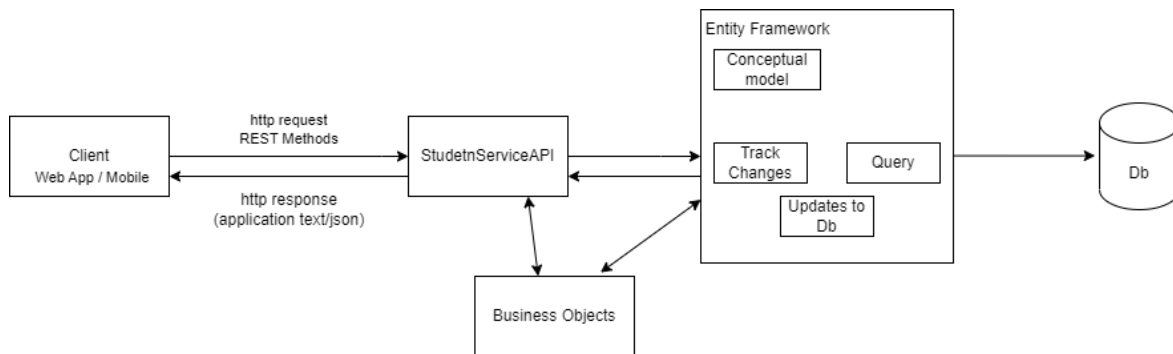
**Microsoft SQL Server (MS SQL)** - to system zarządzania bazami danych (DBMS) opracowany przez firmę Microsoft. Jest to bardzo popularne narzędzie do zarządzania dużymi zbiorami danych, które jest szeroko stosowane w różnych rodzajach organizacji, od małych firm po duże korporacje.

**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  
}
```

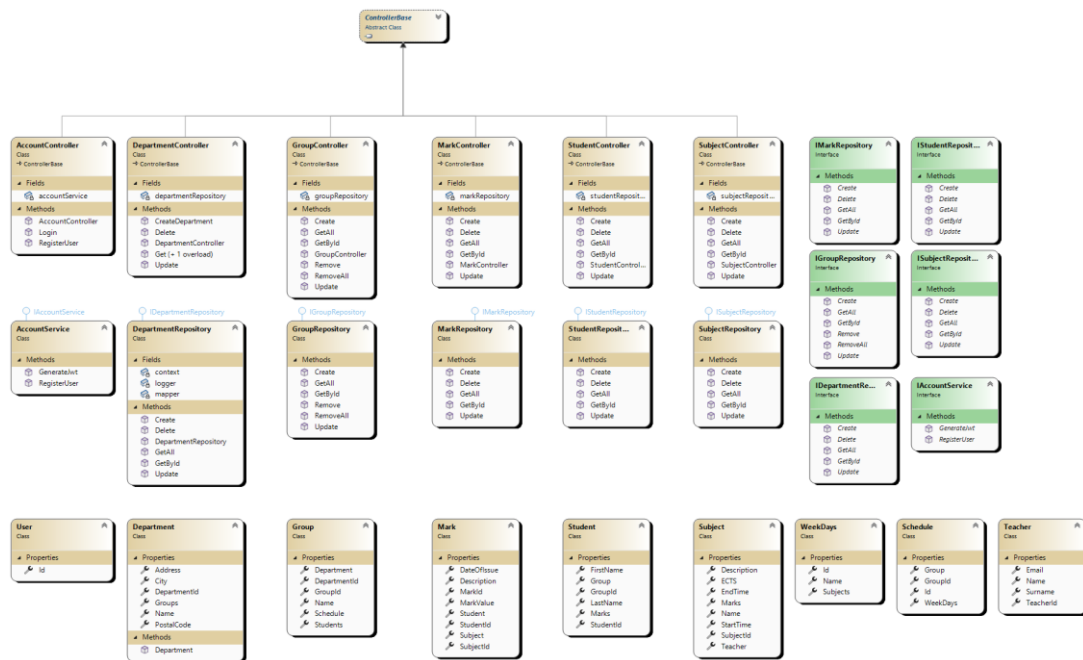
### Usuń obiekt typu Subject o podanym Id

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.

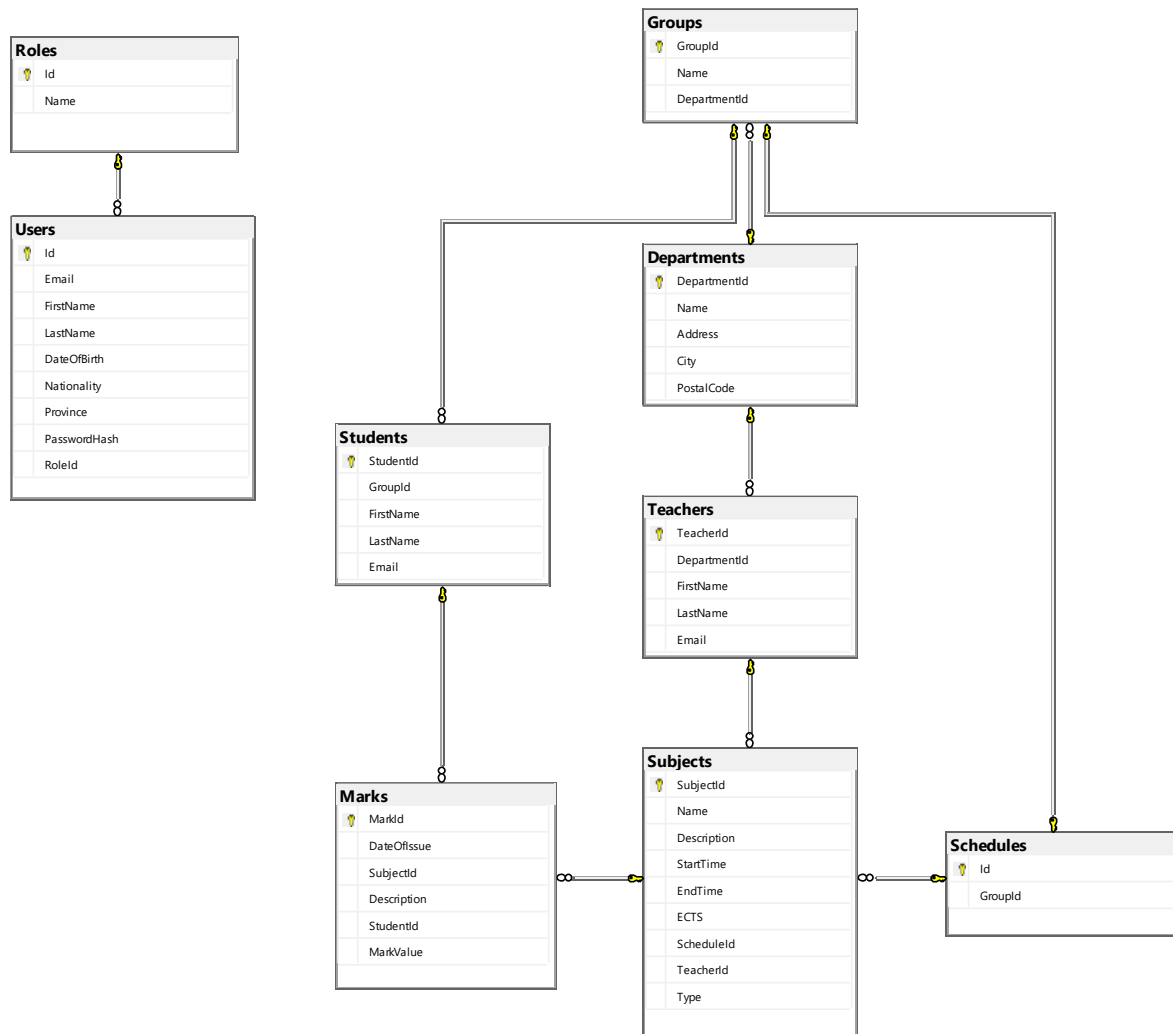


[Diagram klas API]

### 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.

## 3.3 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.



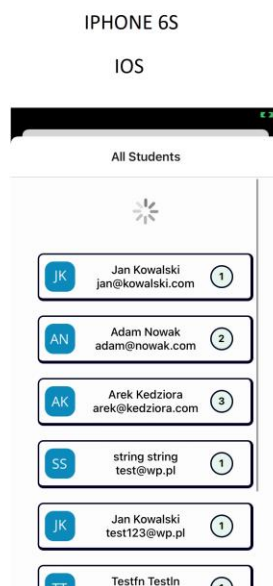
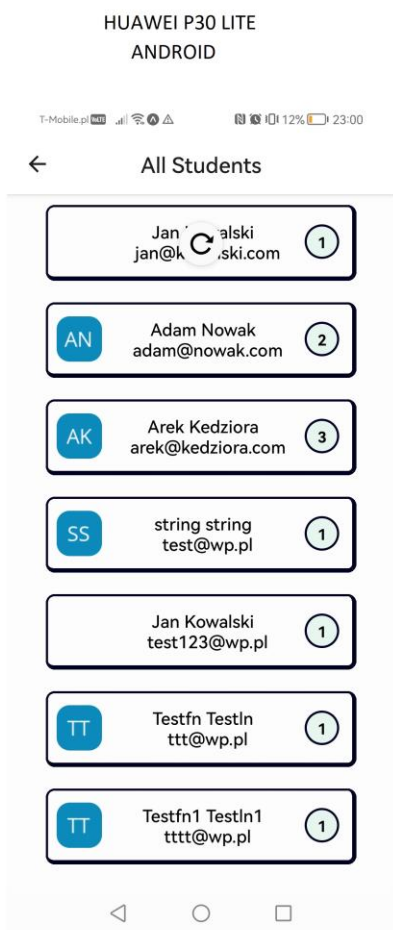
**React Hook Form** - React-hook-form to biblioteka stworzona z myślą o ułatwieniu tworzenia formularzy w aplikacjach React i React Native, pozwala na łatwe i szybkie tworzenie formularzy, dzięki wykorzystaniu hooków w celu zarządzania stanem pól formularza i walidacji danych.

### **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.

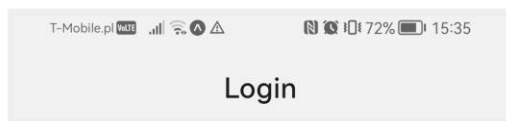


**Zdjęcie 1** Przykład odświeżania ekranu z elementami

## Logowanie

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

HUAWEI P30 LITE  
ANDROID



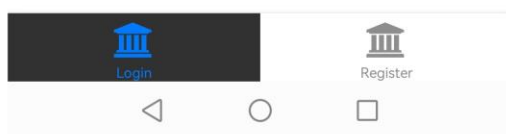
IPHONE 6S

IOS



Login

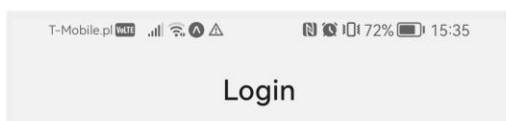
Login



**Zdjęcie 2 Ekran logowania**

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

HUAWEI P30 LITE  
ANDROID

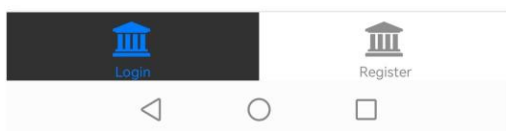


IPHONE 6S  
IOS



Login

Login

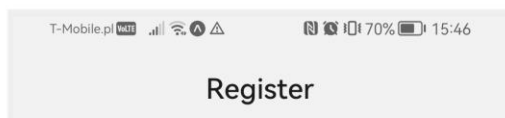


Zdjęcie 3 Ekran logowania - wymagane pola

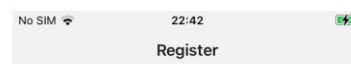
## Rejestracja

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

HUAWEI P30 LITE  
ANDROID



IPHONE 6S  
IOS



**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



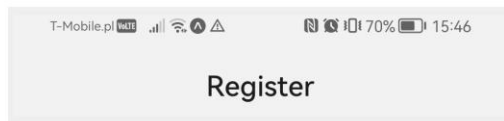
Login



Register

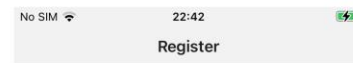
**Zdjęcie 4 Ekran Rejestracji**

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**

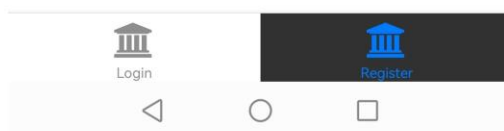
Email  
**Email is required**

Password  
**Password is required**

Confirm Password  
**Password is required**

[Register](#)

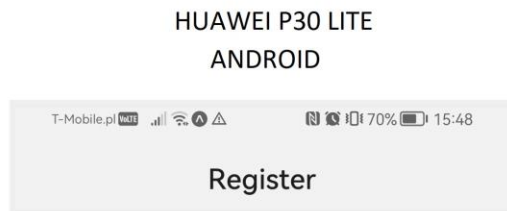
Navigate to **login** page



Zdjęcie 5 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

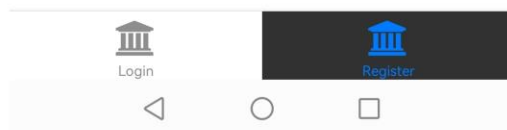
**Register**

**Account have been created!**

[Register](#)

Navigate to **login** page

 Login  Register

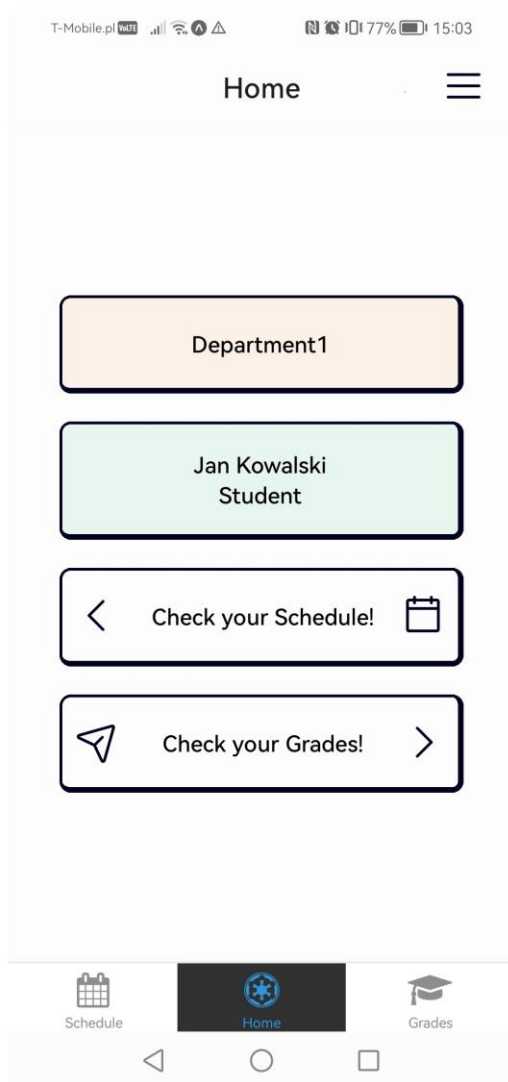


Zdjęcie 6 Poprawnie stworzone konto

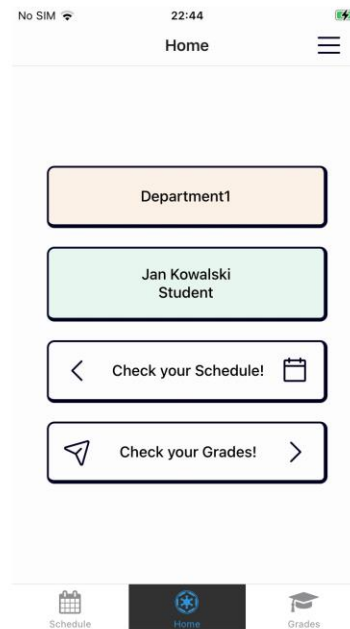
### 3.3.2.2 Ekrany Aplikacji - Student

Ekran główny po zalogowaniu

HUAWEI P30 LITE  
ANDROID



IPHONE 6S  
IOS



**Zdjęcie 7 Ekran główny studenta**

**Plan zajęć**



HUAWEI P30 LITE  
ANDROID



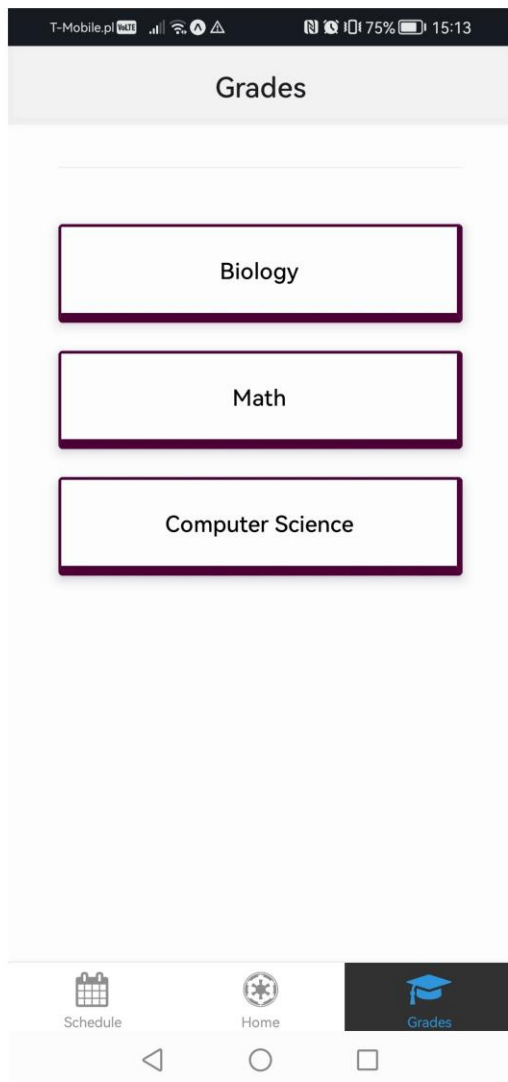
IPHONE 6S  
IOS



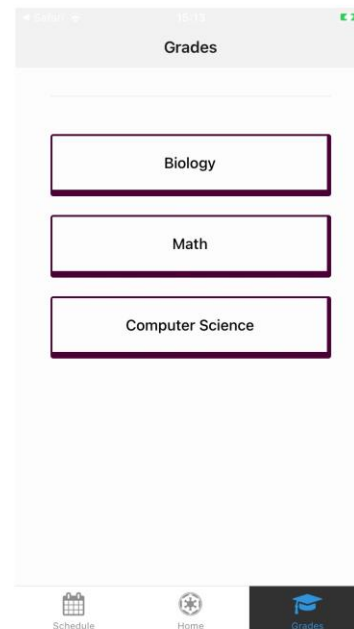
**Zdjęcie 8 Plan studenta**

**Przedmioty studenta**

HUAWEI P30 LITE  
ANDROID



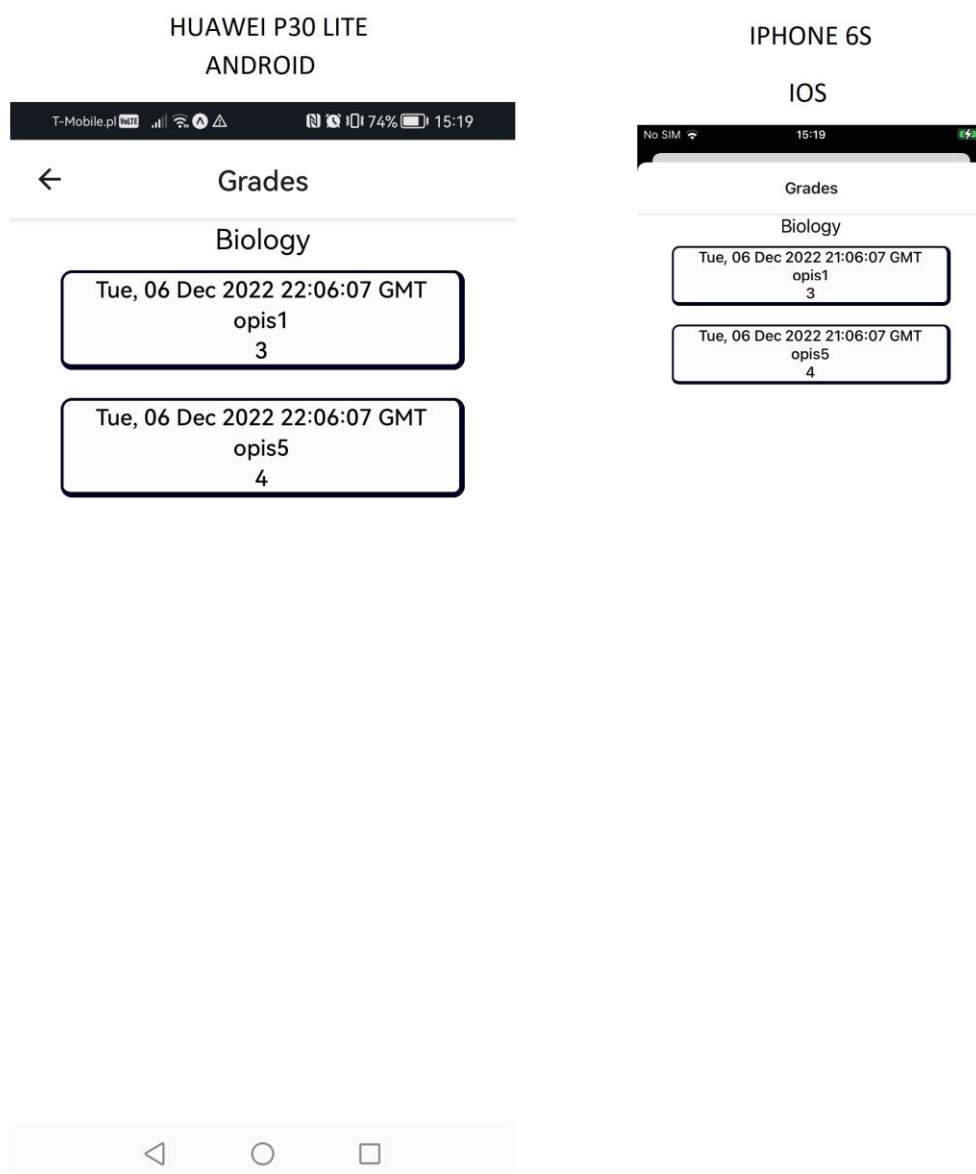
IPHONE 6S  
IOS



**Zdjęcie 9 Przedmioty studenta**

## Oceny studenta

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

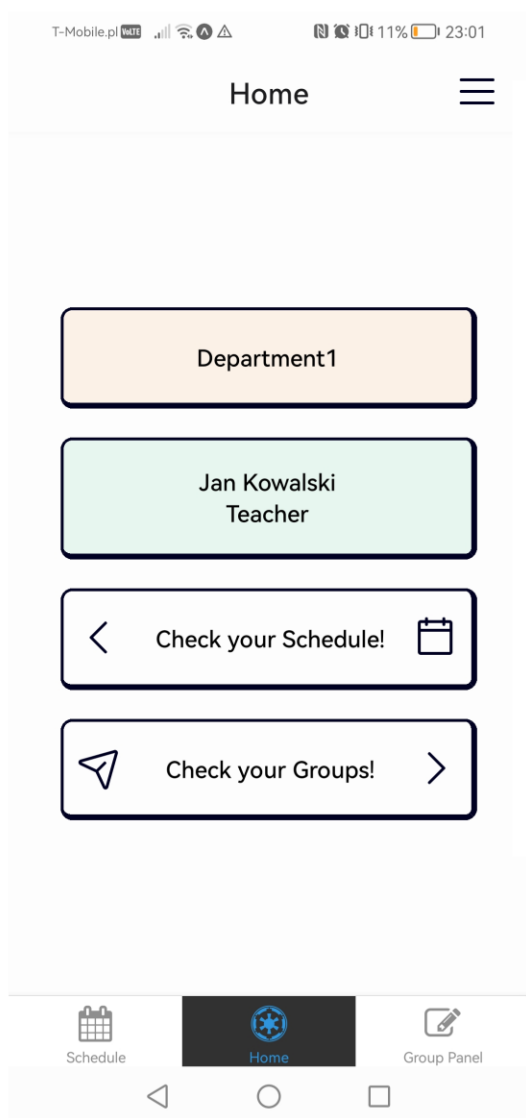


Zdjęcie 10 oceny studenta

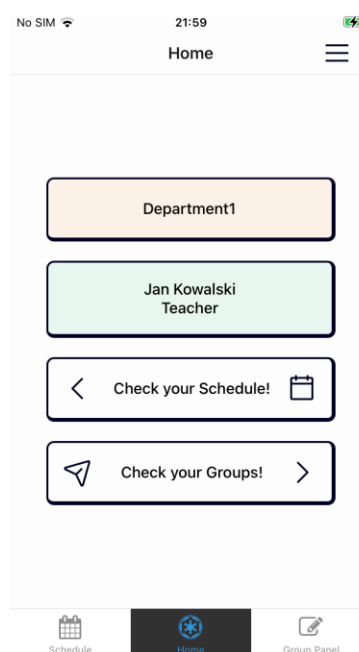
### 3.3.2.3 Ekran Aplikacji – Nauczyciel

#### Ekran główny po zalogowaniu.

HUAWEI P30 LITE  
ANDROID



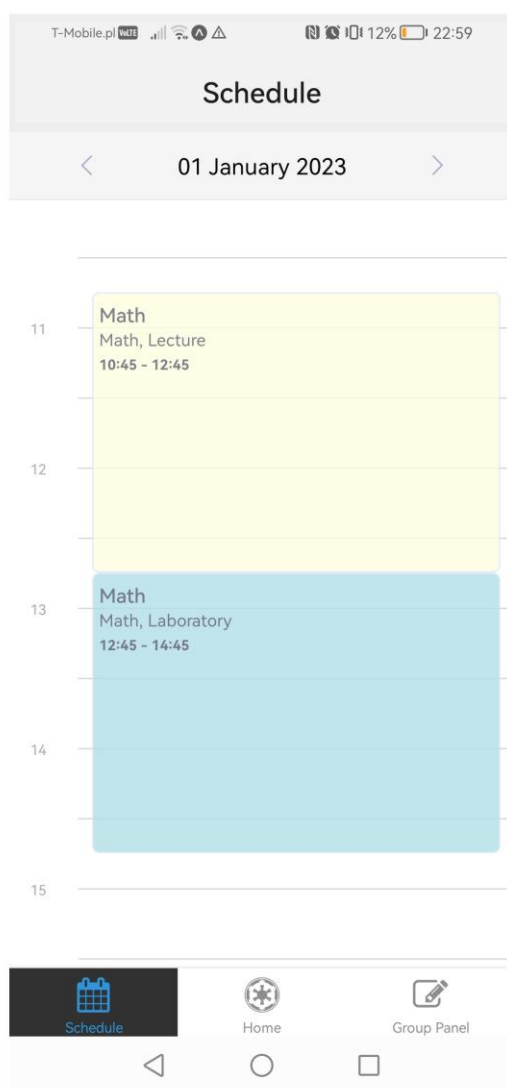
IPHONE 6S  
IOS



Zdjęcie 11 Ekran główny - Nauczyciel

## Plan zajęć

HUAWEI P30 LITE  
ANDROID



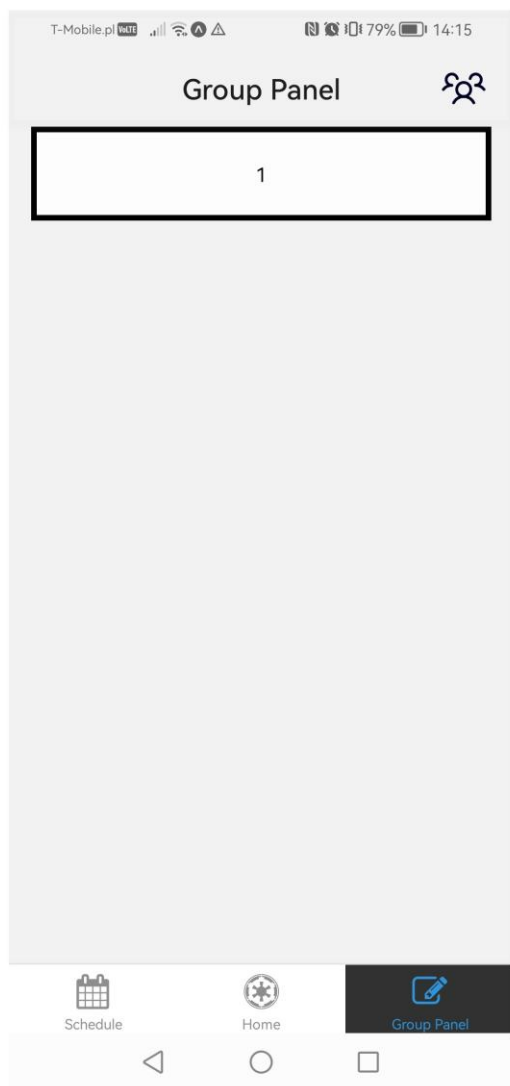
IPHONE 6S  
IOS



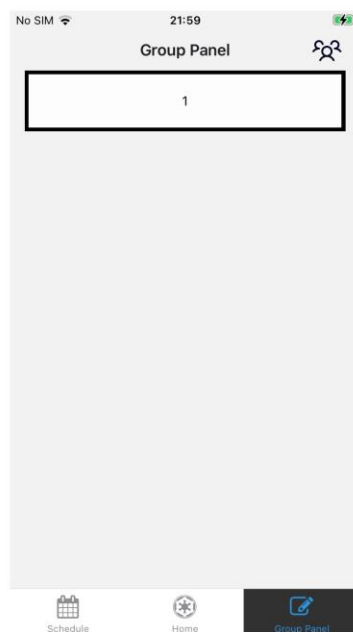
Zdjęcie 12 Plan Zajęć

## Lista grup nauczyciela

HUAWEI P30 LITE  
ANDROID



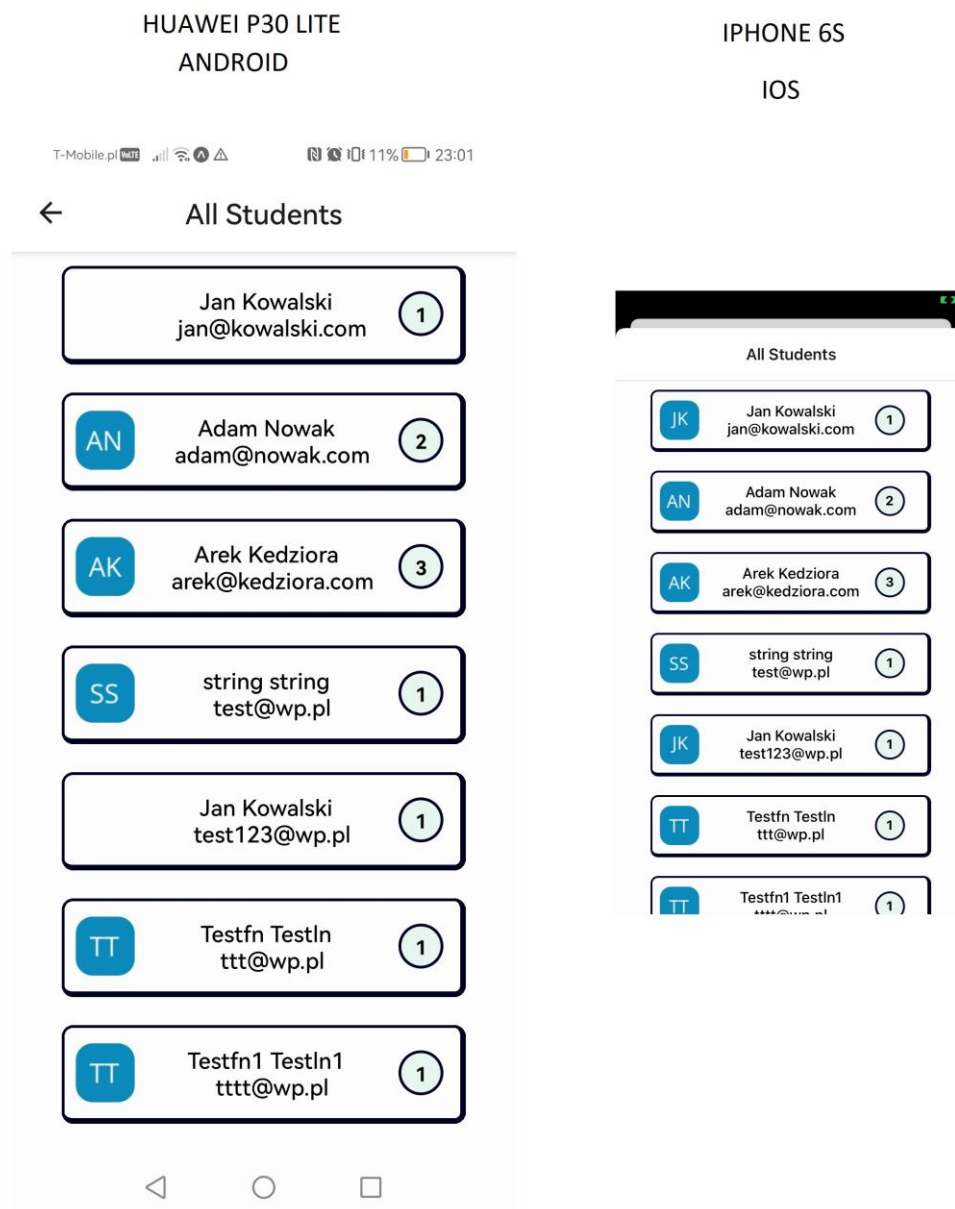
IPHONE 6S  
IOS



Zdjęcie 13 Lista grup nauczyciela

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

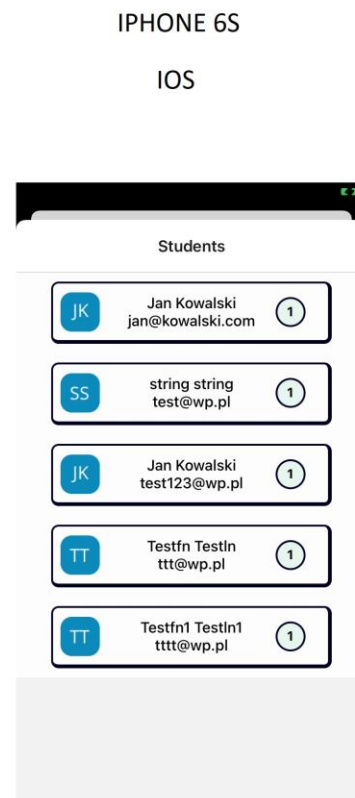
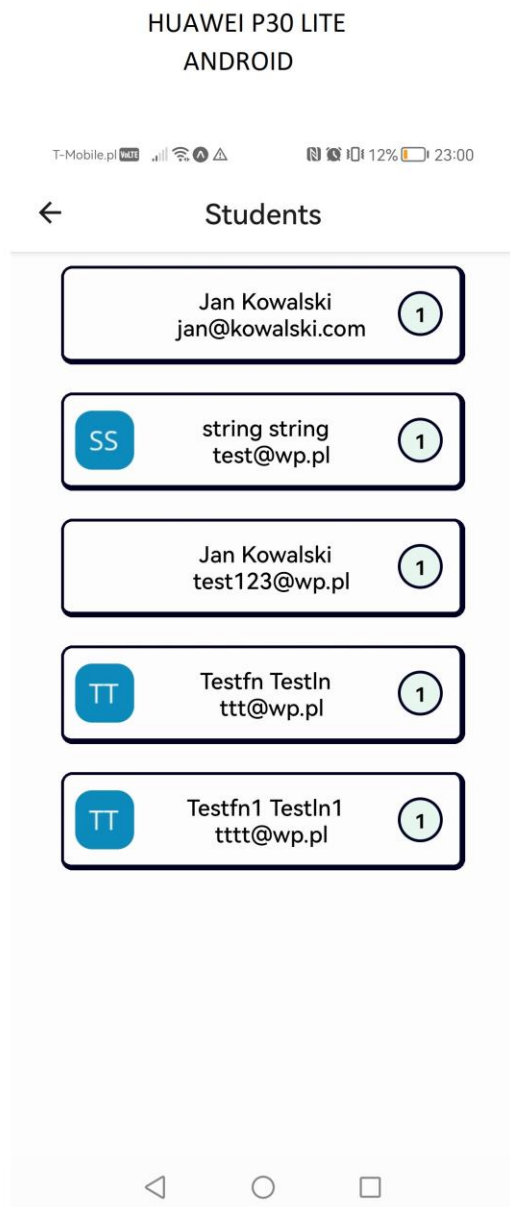
### Lista wszystkich studentów



Zdjęcie 14 Lista wszystkich studentów

## Lista wybranej grupy

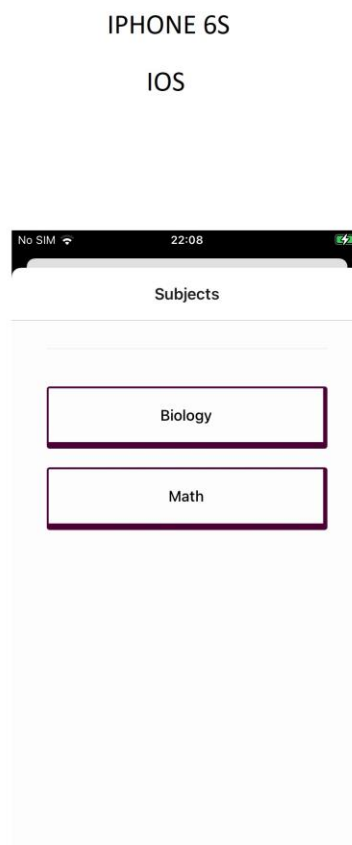
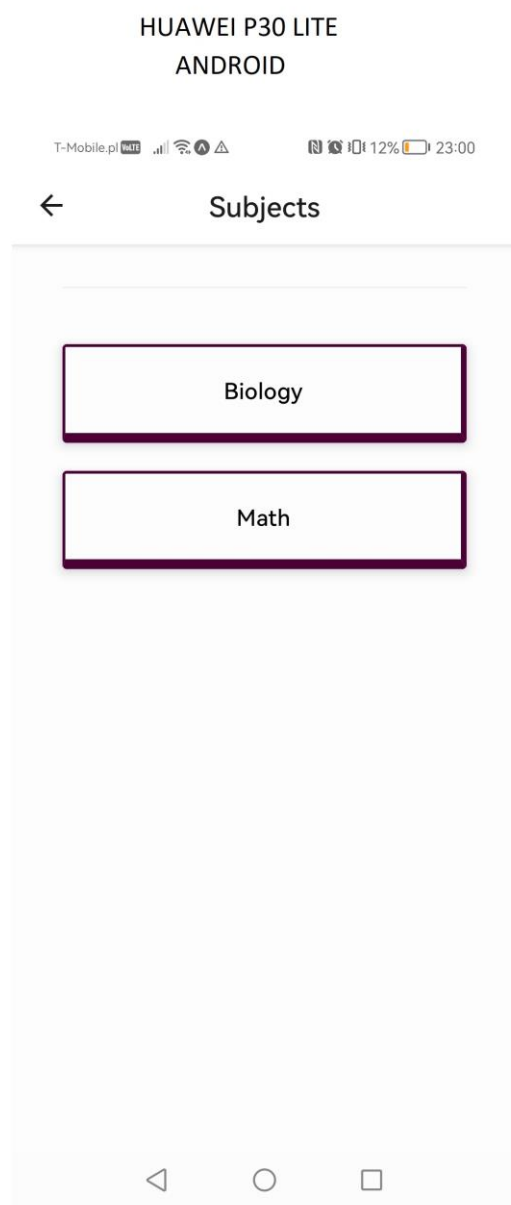
Klikając na daną grupę możemy przejść do listy studentów wybranej grupy.





### **Lista przedmiotów studenta**

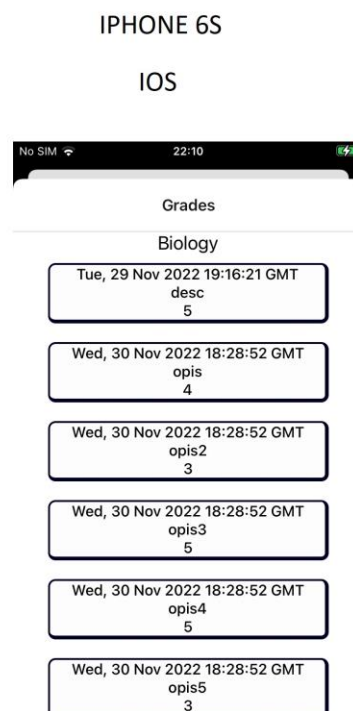
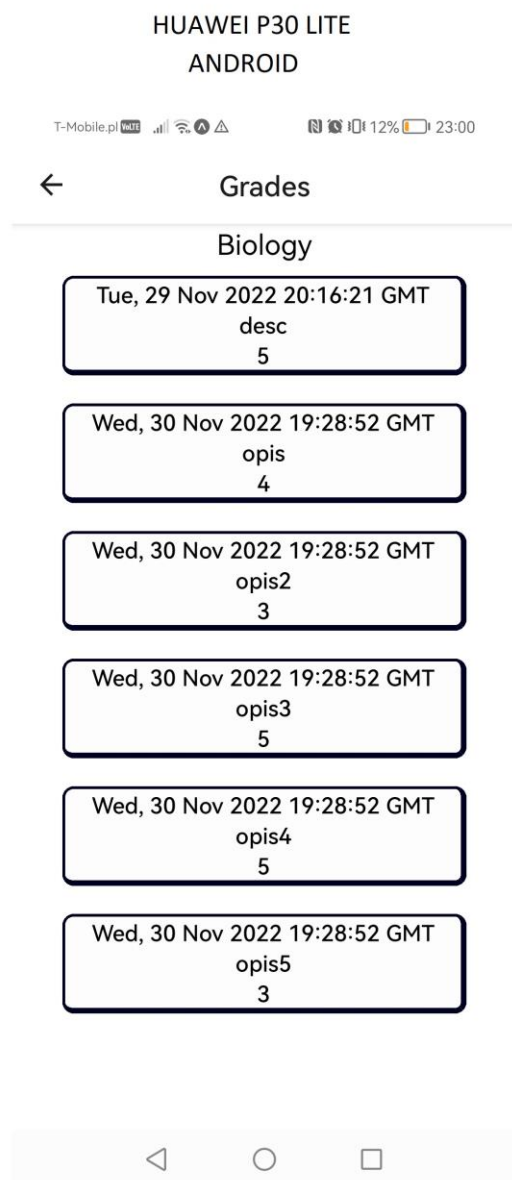
Klikając na profil studenta przechodzimy do listy jego przedmiotów.



**Zdjęcie 16** Lista przedmiotów wybranego studenta

## Oceny wybranego studenta

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

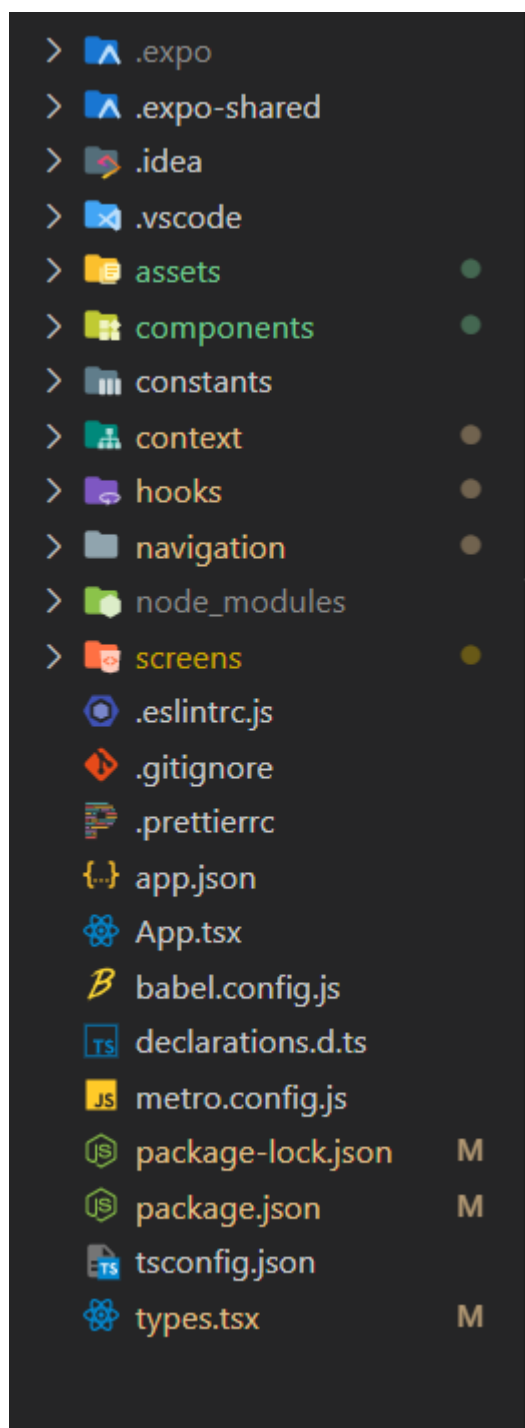


Zdjęcie 17 Oceny wybranego studenta

### 3.3.3 Dokumentacja techniczna

Do procesu tworzenia aplikacji został wykorzystany edytor Visual Studio Code.

#### 3.3.3.1 Nawigacja w projekcie



Zdjęcie 18 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”.