



Akademia nauk stosowanych
w Elblągu
Instytut Informatyki Stosowanej
im. Krzysztofa Brzeskiego

**PRACA DYPLOMOWA
INŻYNIERSKA**

System ewidencji i rozliczania studenckich praktyk zawodowych

Autorzy:

Maciej Sierżęga 18737

Karol Mikołajewski 18582

Opiekun Pracy: dr inż. Jerzy Buriak, prof. uczelni

Elbląg, 2022

Spis treści.....

1. Cel pracy
 - 1.1. Zakres pracy wspólnej
 - 1.2. Zakres pracy Karola Mikołajewskiego
 - 1.3. Zakres pracy Macieja Sierżęgi
2. Technologie (biblioteki, narzędzia, moduły)
 - 2.1. Front-end - React.js
 - 2.1.1. React-router-dom
 - 2.1.2. Material ui
 - 2.1.3. React-toastify
 - 2.1.4. Axios
 - 2.1.5. React-paginate
 - 2.2. Back-end - Node.js
 - 2.2.1. Sequelize ORM
 - 2.2.2. Body-parser
 - 2.2.3. Multer
 - 2.2.4. Nodemon
 - 2.2.5. Express-session
3. Baza danych

1. Cel pracy

Naszym celem pracy, było utworzenie aplikacji internetowej, która między innymi:

- Ułatwia studentom rozliczanie się z ich praktykami zawodowymi, poprzez prowadzenie dzienniczka
- Polepsza rozliczanie praktyk oraz możliwość pomocy opiekunom pośród ich studentów
- Umożliwia pracownikom dziekanatu proste zarządzanie zakładami pracy oraz studentami, którzy w nich przebywają

1.1. Zakres pracy wspólnej

- Analiza funkcjonalna i założenia systemu,
- Projekt bazy danych, a także zaimplementowanie jego do aplikacji za pomocą modelu ORM,
- Implementacja projektu używając React.js do front-endu oraz Node.js do back-end'u,
- Dobór odpowiednich modułów do aplikacji,
- Wykonanie responsywnego interfejsu aplikacji,
- Testowanie działalności aplikacji oraz poprawa graficzna interfejsu

Zakres pracy Karola Mikołajewskiego:

- Utworzenie funkcjonalności dla profilu:
 - Studenta,
 - Dziekanatu,
 - Dyrektora,

Zakres pracy Macieja Sierżęgi:

- Utworzenie systemu logowania oraz rejestracji,
- Utworzenie sesji dla użytkowników aplikacji oraz zabezpieczenie dostępu,
- Utworzenie możliwości edycji swojego konta (hasło),
- Utworzenie funkcjonalności dla profilu:
 - Administratora
 - Opiekuna Zakładowego

- Opiekuna Uczelnianego,

1.2.

1.3.

1.4.

2. Wykorzystane technologie - biblioteki, narzędzia, moduły

2.1. Front-end - React.js

Jako bibliotekę budowy interfejsu użytkownika wykorzystano React.js. Jest to biblioteka javascript, wykorzystywana do szybkiego tworzenia interfejsów graficznych dla użytkowników danej aplikacji internetowej.

Podstawowymi komponentami React.js są:

- **React-router-dom**
- **Material ui**
- **React-toastify**
- **React-paginate**
- **Axios**

React-router-dom pozwala na stworzenie nawigacji pomiędzy stronami aplikacji, a co najważniejsze, daje możliwość wykonania zabezpieczeń przed niepowołanymi użytkownikami. [1]

Material ui znacznie ułatwia budowanie interfejsów graficznych, gdyż pozwala na korzystanie z wielkiej liczby gotowych komponentów, które są czytelne, posiadają animacje, a także łatwą edycję. Praktycznie każdy komponent przydaje się do czegoś innego, więc mamy gigantyczną możliwość uzyskania, tego czego potrzebujemy akurat w naszej aplikacji.

React-toastify pozwala bardzo szybko i łatwo utworzyć powiadomienie dla użytkownika interfejsu.

Axios pozwala nam na wykonywanie żądań HTTP z przeglądarki oraz obsługę danych żądania i odpowiedzi płynących z back-end'u

React-paginate to komponent pozwalający na łatwe wprowadzenie stronicowania danych do aplikacji, niestety trzeba napisać własny css do poprawienia wyglądu komponentu.

2.2. Back-end - Node.js

Asynchroniczne środowisko wykonawcze JavaScript sterowane zdarzeniami, jest przeznaczony do tworzenia skalowalnych aplikacji internetowych gdzie wiele połączeń może być obsługiwanych jednocześnie. Przy każdym połączeniu wywołanie zwrotne jest uruchamiane, ale jeśli nie ma żadnej pracy do wykonania, Node.js będzie uśpiony.

- 2.2.1. **Sequelize ORM** (ang. Object-Relational Mapping – ORM) - odwzorowuje składnię naszej bazy danych w postaci obiektu. Używając ORM optymalizuje to nasze zapytania SQL, dzięki czemu łatwiej operuje się na danych oraz tworzy strukturę tabel.
- 2.2.2. **Body-parser** - analizuje przychodzące treści żądania pomiędzy klientem a serwerem
- 2.2.3. **Multer** - służy do obsługi danych wieloczęściowych bądź formularzy, które jest używane głównie do przesyłania plików.
- 2.2.4. **Bcrypt** - biblioteka pozwalająca na szyfrowanie danych oraz ich odszyfrowywanie.
- 2.2.5. **Nodemon** - to narzędzie, które pozwala na automatyczne ponowne uruchamianie aplikacji po wykryciu zmian w plikach.
- 2.2.6. **Express-session** - daje możliwość na zarządzanie sesjami użytkowników.

3. Baza danych

Diagram

tabela tabeli

nazwa typ przeznaczenie

4. System logowania



AKADEMIA
NAUK STOSOWANYCH
w ELBLĄGU

E-mail: *

Hasło: *

ZALOGUJ SIĘ

Rys. 1 Wygląd komponentu logowania

Po kliknięciu w przycisk odpowiadający za funkcję logowania, axios wysłał zapytanie 'post' do backendu (Rys. 2), który sprawdza czy podane dane są poprawne, czyli czy znajduje się taki użytkownik w bazie danych o takim loginie i hasle (Rys 3).

```
const loginToAccount = async () => {  
  await Axios.post(`${url}loginToAccount`, {  
    login: login,  
    password: password,  
  }).then((res) => {  
    if (res.data.message) setLoginStatus(res.data.message);  
    if (res.data.logged) {  
      props.setStatus(res.data);  
    } else {  
      setOpen(true);  
    }  
  });  
};
```

Rys. 2 Wysłanie do backendu zapytanie o logowanie

```

exports.loginToAccount = async (req, res) => {
  const { login, password } = req.body;

  const checkLogin = await user.findOne({
    where: {
      login: login,
    },
  });
  if (!checkLogin) {
    req.session.logged = false;
    res.send({
      message: "Błędny login lub hasło",
    });
  } else if (checkLogin) {
    if (await bcrypt.compare(password, checkLogin.hasło)) {
      req.session.user = checkLogin;
      req.session.logged = true;
      res.send({
        logged: true,
        user: checkLogin,
      });
    } else {
      res.send({
        message: "Błędny login lub hasło",
      });
      req.session.logged = false;
    }
  }
};

```

Rys. 3 Backend sprawdza podane dane

Na początku wyszukiwany jest 'user' poprzez funkcję 'findOne' zawartą w module 'sequelize', jeżeli jest, to za pomocą modułu 'bcrypt' i jej funkcji 'compare' jesteśmy w stanie w łatwy sposób odszyfrować hasło, które znajduje się zaszyfrowane w bazie danych. Jeśli informacje o loginie i hasle się zgadzają, to do frontendu wysyłana jest informacja o użytkowniku i tworzona jest sesja użytkownika. Jeśli natomiast zdarzy się tak, że użytkownik podał błędne hasło lub login, informacja ta również wysyłana jest do frontendu i użytkownik otrzymuje taką wiadomość jak na rycinie nr 4.

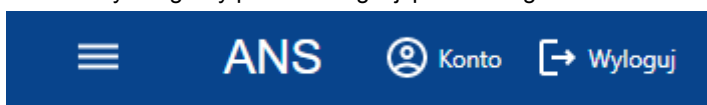
A screenshot of a login form. At the top, a red banner contains a white exclamation mark icon and the text 'Błędny login lub hasło', with a white 'X' icon on the right. Below this are two input fields. The first is labeled 'E-mail: *' and contains the text 'maciej@ans.student.pl'. The second is labeled 'Hasło: *' and contains four dots. At the bottom is a large blue button with the white text 'ZALOGUJ SIĘ'.

Rys. 4 informacja o błędnych danych zalogowania

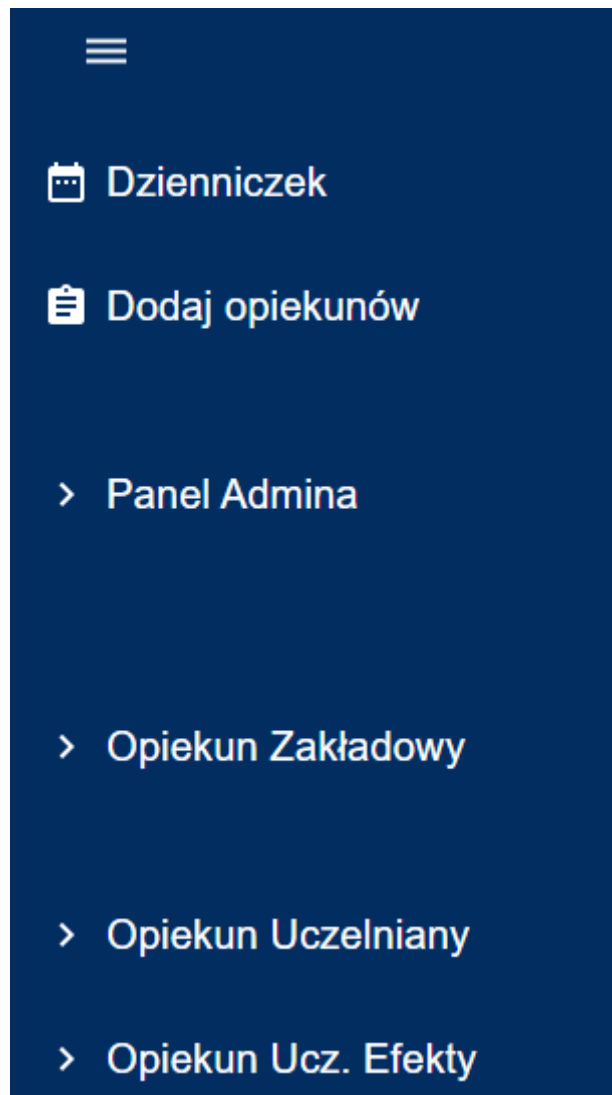
Po pomyślnym zalogowaniu zmienia nam się górny pasek nawigacyjny, a także pojawia się boczny pasek, który pozwala nam na nawigację zależną od ról zalogowanego użytkownika.



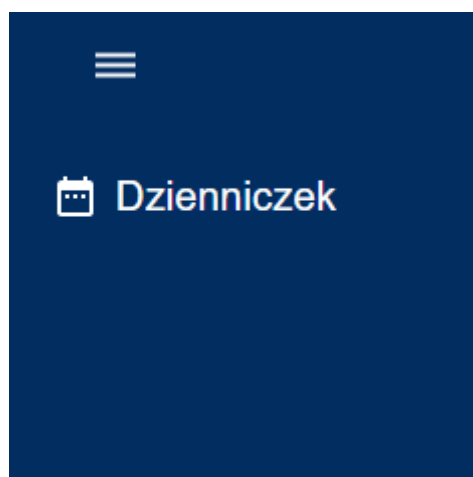
Rys. 3 górny pasek nawigacji przed zalogowaniem



Rys. 5 górny pasek nawigacji po zalogowaniu wraz z paskiem bocznym



Rys. 6 boczny pasek po uruchomieniu użytkownika ze wszystkimi rolami



Rys. 7 boczny pasek po uruchomieniu użytkownika z rolą studenta, czyli domyślną

5. System rejestracji



AKADEMIA
NAUK STOSOWANYCH
w ELBLĄGU

E-mail: *


Hasło: *

Powtórz hasło: *


UTWÓRZ KONTO

Rys. 7 wygląd komponentu logowania

Do utworzenia konta potrzebujemy podać e-mail, na którym będzie zarejestrowane konto a także hasło wraz z powtórzeniem w celach sprawdzenia, czy użytkownik nie pomylił się przy wpisywaniu.



AKADEMIA
NAUK STOSOWANYCH
w ELBLĄGU



AKADEMIA
NAUK STOSOWANYCH
w ELBLĄGU

! Niestety taki e-mail jest już zajęty

E-mail: *

1

Hasło: *

Powtórz hasło: *

UTWÓRZ KONTO

! Hasła się nie zgadzają

E-mail: *

asd

Hasło: *

Powtórz hasło: *

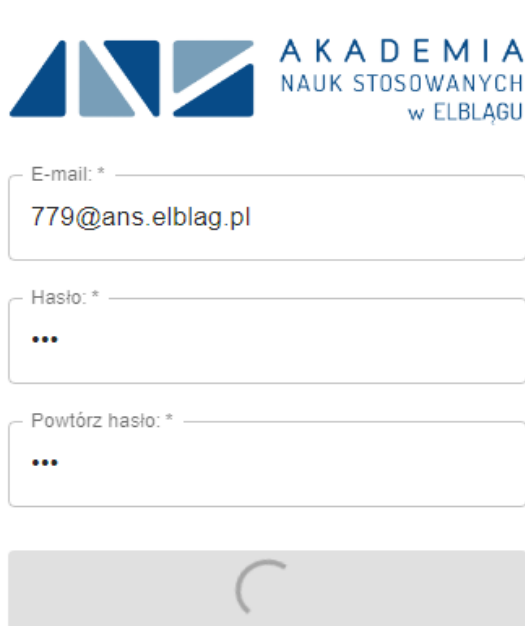
UTWÓRZ KONTO

Rys. 8 i 9 otrzymanie informacji o niepomyślnej rejestracji

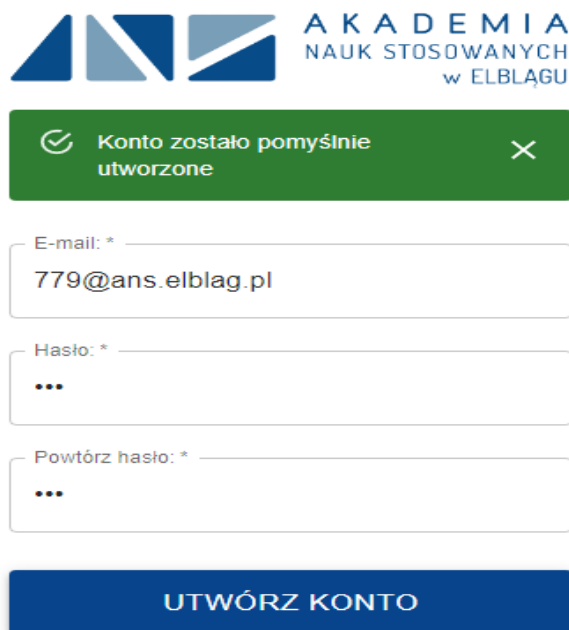
Jeśli hasło wraz z hasłem powtórzeniowym nie będzie się zgadzało dostaniemy informację, iż dane hasła się nie zgadzają.

Aby również uchronić się przed rejestracją kilku użytkowników na ten sam podany mail, zabezpieczenie wraz z wiadomością zwrotną, iż e-mail jest zajęty, również otrzyma użytkownik, który próbował utworzyć konto na mailu, który jest zajęty.

Po każdym razie, gdy użytkownik naciśnie przycisk należący do utworzenia konta, w trakcie sprawdzania danych przez serwer, przycisk zostanie wyłączony do momentu, aż serwer wyśle informację, gdy konto użytkownika przejdzie przez wszystkie zabezpieczenia, konto utworzy się, a użytkownik otrzyma o tym informację.



Rys. 10 oczekiwanie na odpowiedź od serwera



Rys. 11 otrzymanie pomyślnej informacji o utworzeniu konta

6. Panel administratora

Do panelu administratora ma dostęp użytkownik, który posiada rolę administratora, jest to bardzo odpowiedzialna rola, gdyż w tym panelu mamy możliwość dodania nowego użytkownika do bazy danych, przyznawania jak i odbierania ról innym kontom, a także zmiany e-mailu i hasła.

Dodawanie nowego użytkownika



Login:

Hasło:

Opiekun Z.



Student



!!Admin!!



Opiekun U.



Dyrektor



Dziekanat



DODAJ

Rys. 8 okno dialogowe pozwalające dodać użytkownika

Szukaj

Q das

DODAJ
UŻYTKOWNIKA

E-mail	Student	Admin	Opiekun Z.	Opiekun U.	Dyrektor	Dziekanat	Akcje
das	✓	✗	✓	✓	✓	✓	
sdasd	✗	✓	✗	✗	✓	✓	
dasdasd as	✓	✗	✗	✗	✓	✗	

Rows per page: 10 1-3 of 3 < >

Rys. 9 przykład wyszukiwania użytkowników

Szukaj

Q dassd

DODAJ
UŻYTKOWNIKA

E-mail	Student	Admin	Opiekun Z.	Opiekun U.	Dyrektor	Dziekanat	Akcje
--------	---------	-------	------------	------------	----------	-----------	-------

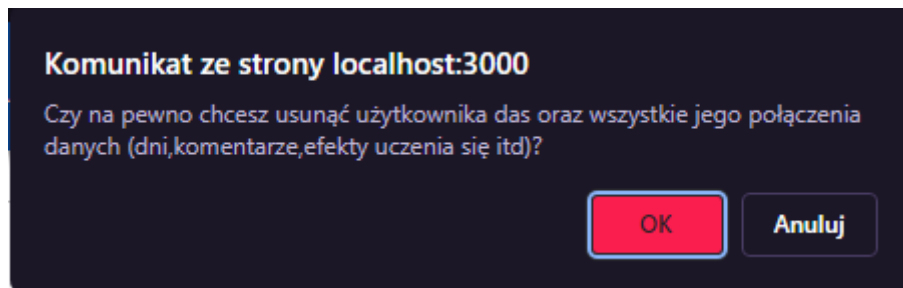
Brak danych...

Rows per page: 10 0-0 of 0 < >

Rys. 10 przykład wyszukiwania użytkownika, którego nie ma w bazie danych

Na samym końcu tabeli mamy kolumnę 'akcje', która odpowiedzialna jest za edycję lub usunięcie danego użytkownika z bazy danych.

Po kliknięciu na przycisk usunięcia użytkownika pojawi się okno potwierdzenia usunięcia danego użytkownika wraz z wiadomością o usunięciu wszystkich innych danych powiązanych z nim.



Rys. 11 okno usunięcia użytkownika

7. Sesje oraz zabezpieczenie dostępu

Jak już wspomniałem w punkcie nr 4, czyli o systemie logowania, po pomyślnym zalogowaniu się, tworzona jest sesja użytkownika.

```
useEffect(async () => {  
  await Axios.get(`${url}loginToAccount`).then((res) => {  
    setAuth(res.data);  
    console.log(res.data);  
  });  
}, [status]);
```

Rys. 12 pobieranie informacji o sesji użytkownika od back-endu

Sesja ta pobierana jest od serwera za każdym razem gdy odświeżymy stronę, lub jeśli zmienimy 'status'.

```
exports.getLoginToAccount = async (req, res) => {  
  if (req.session.user) {  
    res.send({ logged: true, user: req.session.user });  
  } else {  
    res.send({ logged: false });  
  }  
};
```

Rys. 13 wysyłanie informacji o sesji użytkownika do front-endu

Serwer na początku sprawdza czy znajduje się sesja użytkownika, która tworzy się po zalogowaniu, jeśli takiej sesji nie ma, to serwer wysyła informację iż 'logged:false', czyli nie ma informacji o zalogowaniu się, natomiast jeśli sesja użytkownika znajduje się na serwerze, to wysyłana jest informacja o statusie zalogowania się, czyli 'logged:true', a także wysyłana jest informacja o użytkowniku, czyli 'user:req.session.user', w której zapisane są role użytkownika jak i jego dane.

```

<Link
  to="/"
  onClick={() => {
    props.setStatus();
    logout();
  }}
  className={classes.links}
>
  <div className={classes.register}>
    <LogoutImg style={{ marginRight: "0.2rem" }} />
    Wyloguj
  </div>
</Link>
</div>

```

Rys. 14 zmiana statusu podczas wylogowania się

Zmiana statusu odbywa się wtedy, gdy naciśniemy przycisk odpowiedzialny za wylogowanie się jak i za zalogowanie się.

Aby zabezpieczyć ścieżki, przed użytkownikami, którzy nie powinni mieć wstępu do konkretnych komponentów, należy utworzyć komponent, który jest odpowiedzialny za zabezpieczenie tego dostępu.

```

function RoleRoute(props) {
  if (props?.logged === undefined) return null;
  return props?.role ? (
    <Outlet />
  ) : props?.logged ? (
    <Navigate to="/noauth" replace />
  ) : (
    <Navigate to="/login" replace />
  );
}

```

Rys. 15 komponent RoleRoute

```

<Route
  element={
    <RoleRoute
      role={auth?.user?.isAdmin}
      logged={auth?.logged}
    />
  }
>
  <Route path="admin" element={<Admin />} />
</Route>

```

Rys. 16 zabezpieczenie ścieżki administratora

Komponent 'RoleRoute' jest odpowiedzialny za sprawdzanie tego otóż dostępu, rysunek nr. 16 ukazuje zabezpieczenie ścieżki administratora, do komponentu odpowiedzialnego za zabezpieczenie jest wysyłana informacja czy użytkownik posiada rolę 'isAdmin' oraz czy jest zalogowany na konto. Jeśli użytkownik nie posiada roli 'isAdmin' następuje przekierowanie go na ścieżkę '/noauth', czyli na ścieżkę, która informuje użytkownika, że nie ma do danej ścieżki dostępu, jeśli natomiast użytkownik nie jest zalogowany, zostaje przekierowany na ścieżkę '/login', odpowiadającą za zalogowanie się do konta.

8. Profil Opiekunów

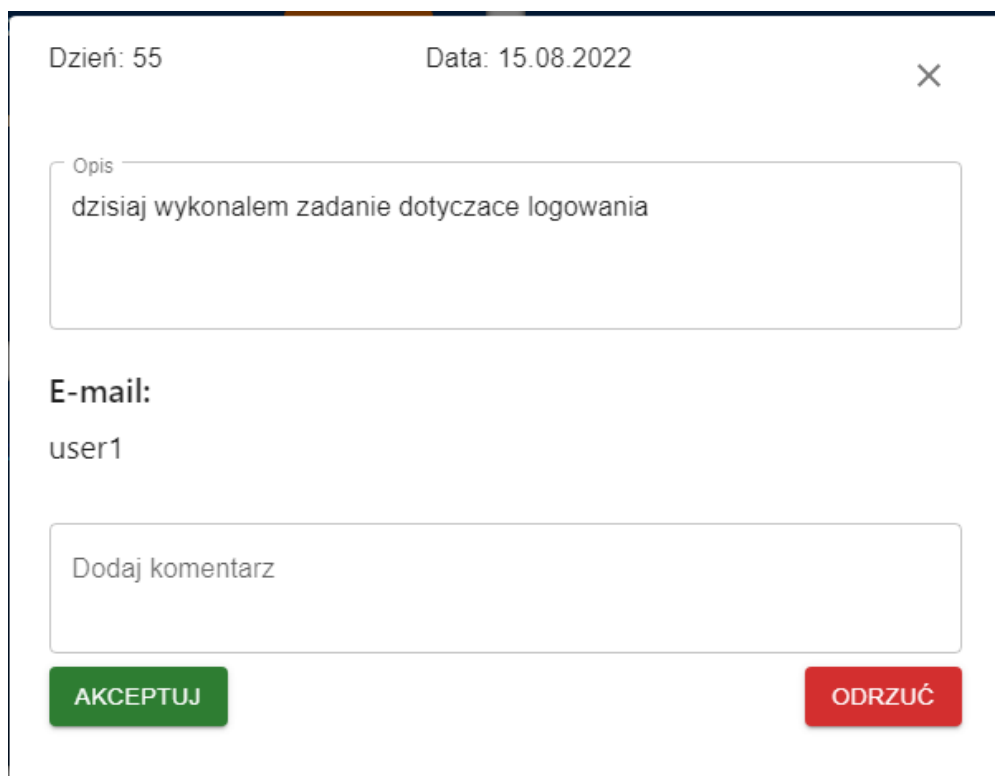
Profil opiekuna uczelnianego jak i opiekuna zakładowego, zostały wykonane w identyczny sposób.

<p>Dzień: 55 Data: 15.08.2022 <input type="button" value="EDYCJA"/></p> <p>Opis: dzisiaj wykonałem zadanie dotyczące logowania</p> <p>Student: user1</p> <p><input type="button" value="AKCEPTUJ"/> <input type="button" value="ODRZUĆ"/></p>	<p>Dzień: 56 Data: 15.08.2022 <input type="button" value="EDYCJA"/></p> <p>Opis: wykonałem dziś rejestrację do konta</p> <p>Student: user1</p> <p><input type="button" value="AKCEPTUJ"/> <input type="button" value="ODRZUĆ"/></p>
<p>Dzień: 57 Data: 23.08.2022 <input type="button" value="EDYCJA"/></p> <p>Opis: coś tam wykonałem</p> <p>Student: user1</p> <p><input type="button" value="AKCEPTUJ"/> <input type="button" value="ODRZUĆ"/></p>	<p>Dzień: 71 Data: 21.09.2022 <input type="button" value="EDYCJA"/></p> <p>Opis:</p> <p>Student: user2</p> <p><input type="button" value="AKCEPTUJ"/> <input type="button" value="ODRZUĆ"/></p>

Rys. 17 wygląd profilu opiekunów

Profil opiekunów, jest odpowiedzialny za nadzorowanie studentów, którzy są przypisani do danych opiekunów.

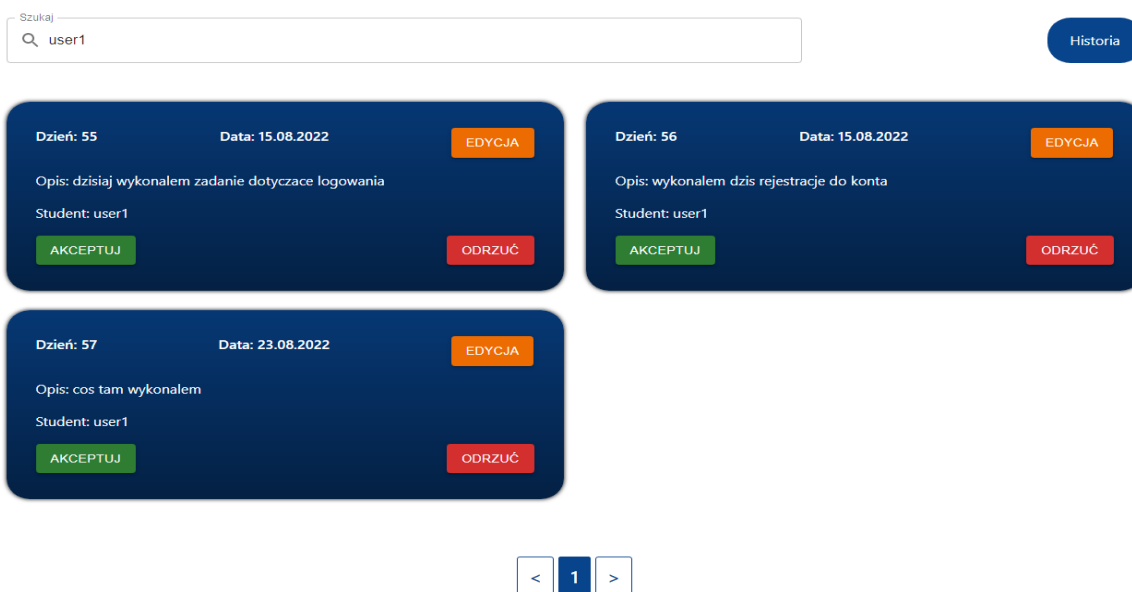
Na przykładzie powyżej, mamy dostęp do dzienniczka dwóch studentów 'user1' i 'user2', mamy możliwość akceptacji oraz odrzucenia dnia, możemy również wejść w dokładniejszą edycję dnia.



Rys. 18 edycja dnia

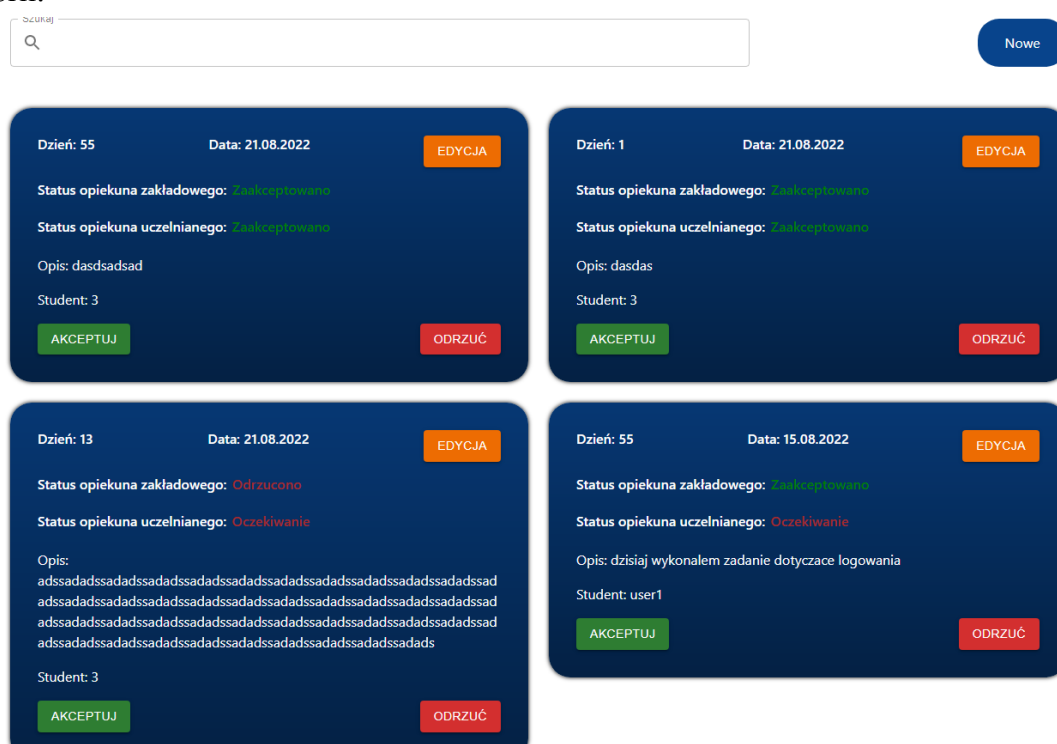
W okienku z dokładną edycją dnia, mamy możliwość dodania komentarza do dnia studenta, jak i zmianę opisu dnia.

Jeżeli dany opiekun, chce wyszukać konkretnie danego studenta po jego e-mailu, może to zrobić w wyszukiwarce.



Rys. 19 wyszukiwanie studenta po e-mailu

Jeżeli opiekun zaakceptuje dzień studenta lub odrzuci, wtedy ten dzień przeniesie się do jego historii.



Rys. 20 historia opiekunów

W historii opiekunów, mamy podgląd na aktualne statusy obu opiekunów, ponownie możemy odrzucić lub zaakceptować dzień, zmienić opis jak i dodać komentarz do dnia, a także przejść ponownie do nowych dni.

Do historii dni opiekunów, jak i do nowych dni dodane jest stronicowanie, pomagające w przejrzystniejszym korzystaniu z tych profili.

9. Dzienniczek

Jest to miejsce dla studenta w którym może prowadzić swój dziennik, który składa się z dwóch części: dzienniczka praktyk oraz efektów uczenia się

9.1. Dzienniczek praktyk

Jest to tabela dla studenta z informacjami takimi jak: dzień, data, status opiekuna zakładowego i uczelnianego oraz opis. Każdy z wpisów można edytować przyciskiem *edytuj*. Przyciskiem *dodaj nowy dzień* dodajemy nowy wpis do dzienniczka.

DODAJ NOWY DZIEŃ					
Dzień	Data	Status Opiekuna Uczelnianego	Status Opiekuna Zakładowego	Opis	
1	12.12.2021	Zaakceptowano	Odrzucono	Lorem Ipsum is simply dummy text of...	EDUTUJ
2	12.09.2022	Zaakceptowano	Zaakceptowano	Lorem Ipsum is simply dummy text of...	EDUTUJ
3	12.09.2022	Zaakceptowano	Zaakceptowano	Lorem Ipsum is simply dummy text of...	EDUTUJ
4	12.09.2022	Zaakceptowano	Odrzucono	Lorem Ipsum is simply dummy text of...	EDUTUJ
5	12.09.2022	Zaakceptowano	Oczekiwanie	Lorem Ipsum is simply dummy text of...	EDUTUJ
6	12.09.2022	Zaakceptowano	Oczekiwanie	Lorem Ipsum is simply dummy text of...	EDUTUJ
8	12.09.2022	Zaakceptowano	Oczekiwanie	Lorem Ipsum is simply dummy text of...	EDUTUJ

Rys. ## zdjęcie przedstawiające wygląd dzienniczka

Przycisk nowego dnia otwiera okno dialogowe w którym student może wprowadzić wszystkie potrzebne informacje oraz załączniki. Wpisuje nr. dnia jego date ilość przepracowanych godzin oraz opis co w danym dniu zrobił.

Dodawanie nowego dnia

Dzień _____ Data _____

Ilość godzin _____

Opis wykonanej pracy _____

Załącz pliki Nie wybrano pliku

Rys. ## okno dodawania nowego dnia

Przykład budowy pola tekstowego, wartość wpisana w pole *opis* jest wykrywana za pomocą funkcji *onChange* i następnie jest przekazywana do zmiennej w tablicy *dayObject*.

Po kliknięciu w przycisk *Dodaj* wykonywana jest funkcja *createDay* za pomocą funkcji *onClick*, która następnie wykonuje żądanie do serwera za pomocą *axios* i przekazuje tablicę *dayObject*, która zawiera informację z pól tekstowych.

```
const createDay = () => {
  axios.post("http://localhost:5000/api/createDay", {
    dayObject: dayObject,
  })
};
const onChange = (e) => {
  const { value, id } = e.target;
  setDayObject({ ...dayObject, [id]: value });
};
```

```
<TextField className={classes.TextField}
  fullWidth
  label="Opis wykonanej pracy"
  id="opis"
  value={opis}
  onChange={(e) => onChange(e)}
  multiline
  margin="normal"/>
<Button
  variant="contained"
  style={{ marginTop: "4%" }}
  onClick={createDay}>
  Dodaj
</Button>
```

Student może w każdej chwili zmienić informacje dnia które wprowadził. W panelu tym wyświetlane są wszystkie informacje z dnia oraz status jego akceptacji. Student aby zmienić dane musi je wprowadzić i zatwierdzić przyciskiem *zmień* lub usunąć przyciskiem *usuń*.

Dzień:1

Dzień

1

Data

12.12.2021

Ilość godzin

8


Opis

Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book. It has su

Zatwierdzenie

Opiekun uczelniany: Zaakceptowano

Opiekun zakładowy: Odrzucono



1662640186016-IMG_7086.gif

USUŃ

Załącz pliki

Wybierz plik

Nie wybrano pliku

Submit

ZMIEN

USUŃ

Rys. ## okno edycji dnia

9.2. Efekty uczenia się

Każdy student aby ukończyć praktyki będzie musiał wypełnić efekty uczenia się, dodawane są one pod względem **kierunku/specjalności** studenta. W każdym efekcie student będzie musiał uzasadnić w jaki sposób go osiągnął.

Lista efektów do realizacji	
Nazwa efektu	Uzasadnienie
Praca w grupie	UZASADNIJ
Poznanie narzędzi, technologii, sprzętu	UZASADNIJ
Prawne skutki własnych działań	UZASADNIJ
BHP	UZASADNIJ
Sposób realizacji zadań	UZASADNIJ

Rys. ## przedstawia efekty uczenia się

Każdy efekt posiada informację o opisie efektu, student musi wprowadzić zmiany i zapisać przyciskiem *zapisz*. Opiekunowie mają dostęp do zmiany w uzasadnianiu efektu danego studenta.

Efekt:BHP
×

Opis: Zna podstawowe zasady bezpieczeństwa pracy i ergonomii w zawodzie informatyka

Opis

ZAPISZ

Rys. ## okno uzasadnienia efektu

10. Zarządzanie zakładami

DODAJ ZAKŁAD

Firma: Przykładowa firma ✎

Opiekuni: EDYCJA OPIEKUNÓW

(Opiekun zakładowy) +/-

Kamil Paciakiak

Studenci:

Roger House Indeks: 40000

Tytus Hawajska Indeks: 30000

(Opiekun zakładowy) +/-

Andrzej TutuotuTutuotuTutuotu

Studenci:

Jakub Wójcik Indeks: 19003

Leon Kowalczyk Indeks: 19004

(Opiekun uczelniany) +/-

Adam Urek

Studenci:

Roger House Indeks: 40000

Tytus Hawajska Indeks: 30000

Jakub Wójcik Indeks: 19003

Leon Kowalczyk Indeks: 19004

Firma: Przykładowa firma 2 ✎

Opiekuni: EDYCJA OPIEKUNÓW

(Opiekun zakładowy) +/-

Marcel Krawczyk

Studenci:

Nikodem Lewandowski Indeks: 19005

Stanisław Zieliński Indeks: 19006

Mikołaj Szymański Indeks: 19007

(Opiekun uczelniany) +/-

Adam Wojciechowski

Studenci:

Nikodem Lewandowski Indeks: 19005

Stanisław Zieliński Indeks: 19006

Mikołaj Szymański Indeks: 19007

Spis tabel

Spis kodu

Spis rysunków

- Rys. 1 Wygląd komponentu logowania
- Rys. 2 Wysłanie do backendu zapytanie o logowanie
- Rys. 3 Backend sprawdza podane dane
- Rys. 3 Backend sprawdza podane dane
- Rys. ### zdjęcie przedstawiające wygląd dzienniczka
- Rys. ### okno dodawania nowego dnia

Rys. ## okno edycji dnia

Rys. ## przedstawia efekty uczenia się

Bibliografia

[1] <https://www.npmjs.com/package/react-router-dom> (dostęp dnia 08.09.2022)