

System do zarządzania bursami i akademikami

Autorzy: Asenata Łuczak, Kamil Łukasik, Eryk Szulczyński

Promotor: dr inż. Michał Trziszka

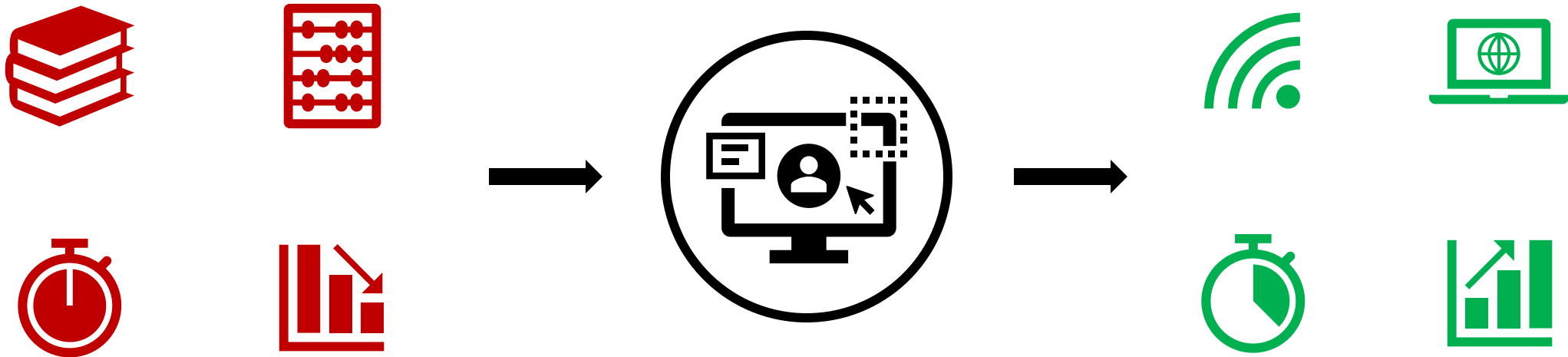
Poznań 2024

Spis treści

- Cel projektu
- Wymagania niefunkcjonalne oraz funkcjonalne
- Środowisko aplikacji
- Automatyzacja procesów
- Baza danych
- Aplikacja Frontend
- Aplikacja Backend
- Dalszy rozwój
- Podsumowanie

Cel projektu

**Cyfryzacja procesów związanych
z administracją w bursach i akademikach**



Wymagania niefunkcjonalne



Aplikacja internetowa



Responsywność



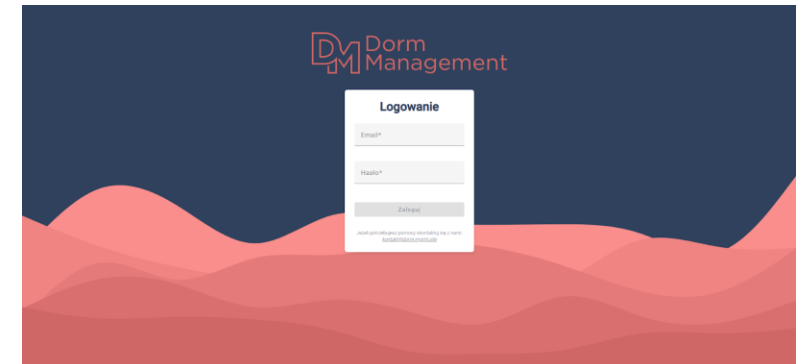
Wspierane przeglądarki



Branding



UX/UI



Bezpieczeństwo/uprawnienia

Wymagania funkcjonalne

The screenshot shows the 'Formularz aplikacji' (Application Form) in the Dorm Management system. It includes fields for 'Opłata za pobyt' (Accommodation fee) and 'Dane wychowanka' (Resident data). The form is titled 'Formularz aplikacji' and has a 'Zakończ formularz' (Finish form) button. The form is divided into sections: 'Opłata za pobyt' and 'Dane wychowanka'. The 'Opłata za pobyt' section has a 'Zapłacono' (Paid) status. The 'Dane wychowanka' section has a 'Nieopłacono' (Not paid) status. The form is titled 'Formularz aplikacji' and has a 'Zakończ formularz' (Finish form) button.

Proces składania podania

The screenshot shows the 'Zgłoszenia' (Applications) page in the Dorm Management system. It displays a list of applications with columns for 'Status', 'Opłata', 'Dane osobiste' (Personal data), 'Dane wychowanka' (Resident data), and 'Dane kontaktowe' (Contact data). The 'Status' column shows 'Zakończony' (Completed) and 'Nieopłacony' (Not paid). The 'Opłata' column shows 'Zapłacono' (Paid) and 'Nieopłacono' (Not paid). The 'Dane osobiste' column shows 'Imię' (Name) and 'Nazwisko' (Surname). The 'Dane wychowanka' column shows 'Imię' (Name) and 'Nazwisko' (Surname). The 'Dane kontaktowe' column shows 'Telefon' (Phone) and 'Email' (Email).

Zarządzanie podaniami

The screenshot shows the 'Użytkownicy' (Users) page in the Dorm Management system. It displays a list of users with columns for 'ID', 'Imię' (Name), 'Nazwisko' (Surname), 'Email', 'Status', and 'Rola' (Role). The 'Status' column shows 'Aktywny' (Active) and 'Nieaktywny' (Inactive). The 'Rola' column shows 'Administrator' (Administrator), 'Pracownik' (Employee), 'Bibliotekarz' (Librarian), and 'Uczelnian' (Student). The 'Imię' column shows 'Imię' (Name) and 'Nazwisko' (Surname). The 'Nazwisko' column shows 'Imię' (Name) and 'Nazwisko' (Surname). The 'Email' column shows 'Email' (Email) and 'Telefon' (Phone). The 'Status' column shows 'Aktywny' (Active) and 'Nieaktywny' (Inactive). The 'Rola' column shows 'Administrator' (Administrator), 'Pracownik' (Employee), 'Bibliotekarz' (Librarian), and 'Uczelnian' (Student).

Zarządzanie użytkownikami

The screenshot shows the 'Pokoje' (Rooms) page in the Dorm Management system. It displays a list of rooms with columns for 'ID', 'Nazwa' (Name), 'Pole powierzchni' (Area), 'Przeznaczenie' (Purpose), and 'Zaopiekowanie' (Maintenance). The 'ID' column shows 'ID' (ID) and 'Nazwa' (Name). The 'Nazwa' column shows 'ID' (ID) and 'Nazwa' (Name). The 'Pole powierzchni' column shows 'Pole powierzchni' (Area) and 'Przeznaczenie' (Purpose). The 'Przeznaczenie' column shows 'ID' (ID) and 'Nazwa' (Name). The 'Zaopiekowanie' column shows 'ID' (ID) and 'Nazwa' (Name).

Zarządzanie pokojami

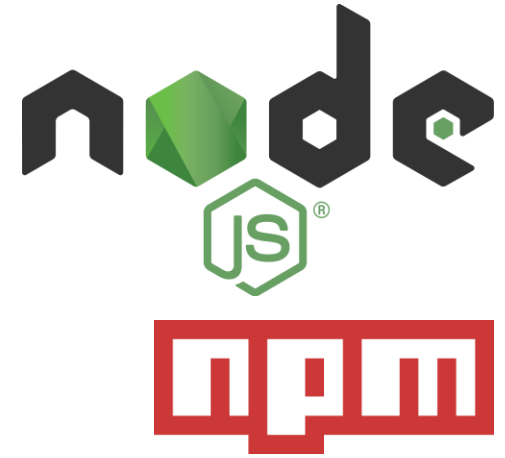
The screenshot shows the 'Kalendarz' (Calendar) page in the Dorm Management system. It displays a calendar for February 2024. The calendar shows days of the week and dates. The 'Dzień' (Day) column shows 'Dzień' (Day) and 'Miesiąc' (Month). The 'Miesiąc' column shows 'Dzień' (Day) and 'Miesiąc' (Month). The 'Dzień' column shows 'Dzień' (Day) and 'Miesiąc' (Month). The 'Miesiąc' column shows 'Dzień' (Day) and 'Miesiąc' (Month).

Kalendarz

The screenshot shows the 'Dokumenty' (Documents) page in the Dorm Management system. It displays a list of documents with columns for 'ID', 'Nazwa' (Name), 'Typ' (Type), and 'Uprawnienia' (Permissions). The 'ID' column shows 'ID' (ID) and 'Nazwa' (Name). The 'Nazwa' column shows 'ID' (ID) and 'Nazwa' (Name). The 'Typ' column shows 'ID' (ID) and 'Nazwa' (Name). The 'Uprawnienia' column shows 'ID' (ID) and 'Nazwa' (Name).

Udostępnianie dokumentów

Środowisko aplikacji (1)

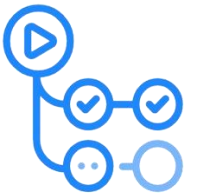


- System operacyjny: Ubuntu 22.04 LTS
 - Stabilny i bezpieczny system operacyjny.
 - Darmowe wsparcie dzięki wyborze wersji LTS do 2027 roku.
 - Aktywna społeczność użytkowników.
- Node.js i NPM (Node Package Manager)
 - Node.js to środowisko uruchomieniowe oparte na silniku V8 JavaScript a jego głównym celem jest umożliwienie uruchamiania kodu JavaScript poza przeglądarką internetową.
 - NPM to narzędzie do zarządzania bibliotekami Node.js, które pozwala na: instalowanie, aktualizowanie i usuwanie zależności.

Środowisko aplikacji (2)



- Nginx:
 - Wysoka wydajność, elastyczność i skalowalność.
 - Wykorzystywany jest do obsługi żądań HTTP/S.
 - Wsparcie dla szyfrowania SSL/TLS
 - Możliwość obsługi wielu domen przez jeden serwer.
- Cloudflare:
 - Dostawca usług DNS.
 - Zapewnia szybkie i niezawodne rozwiązywanie nazw domenowych.



GitHub Actions

Automatyzacja procesów

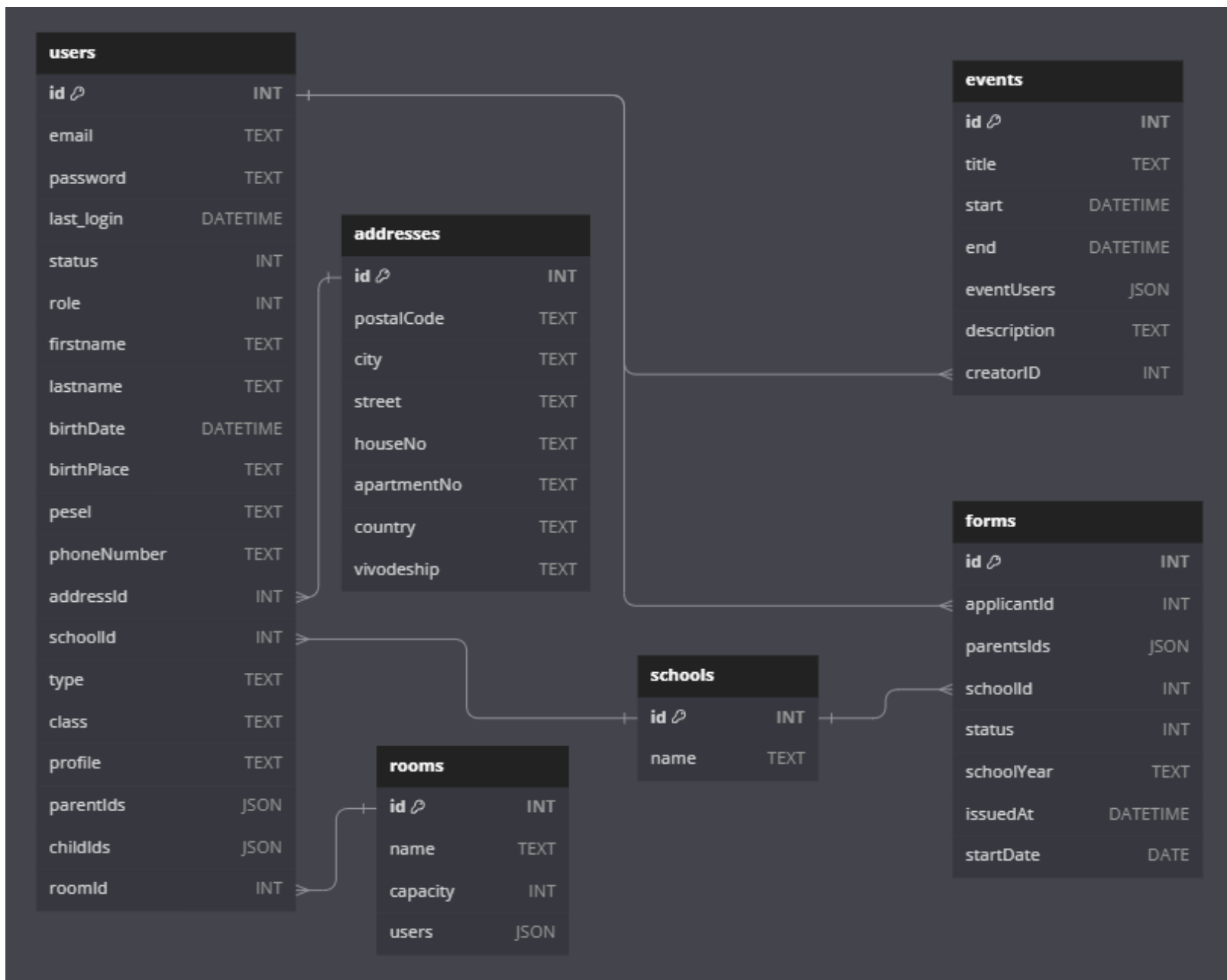
```
1 name: FRONT CI
2
3 on:
4   push:
5     branches: [ main ]
6
7 jobs:
8   build:
9     runs-on: self-hosted
10
11     strategy:
12       matrix:
13         node-version: [18.x]
14
15     steps:
16       - uses: actions/checkout@v2
17       - name: Use node.js ${ matrix.node-version }
18         uses: actions/setup-node@v1
19         with:
20           node-version: ${ matrix.node-version }
21
22       - name: Install Dependencies
23         run: npm i
24
25       - name: Build
26         run: npm run build
27
28       - name: Deploy
29         run: cp -r dist/client/* /opt/repo/front
```

- GitHub Actions
narzędzie do automatyzacji procesów w projekcie, wbudowane bezpośrednio w platformę GitHub.
 - Ciągła Integracja (CI):
Automatyczne budowanie projektu po wprowadzeniu zmiany i testowanie aplikacji.
 - Ciągła Dostawa (CD):
Automatyczne wdrażanie aplikacji do środowiska produkcyjnego po pomyślnej integracji.
- W projekcie automatyzacja obejmowała:
aktualizacje pakietów node.js, budowanie kodu oraz implementacje na serwerze.

Baza danych

Tabele:

- Użytkownicy
- Adresy
- Wydarzenia
- Pokoje
- Szkoły
- Formularze



Aplikacja frontend (1)

Routing

Guards
Interceptors

Forms

Validators



Services

Components

Template
Class

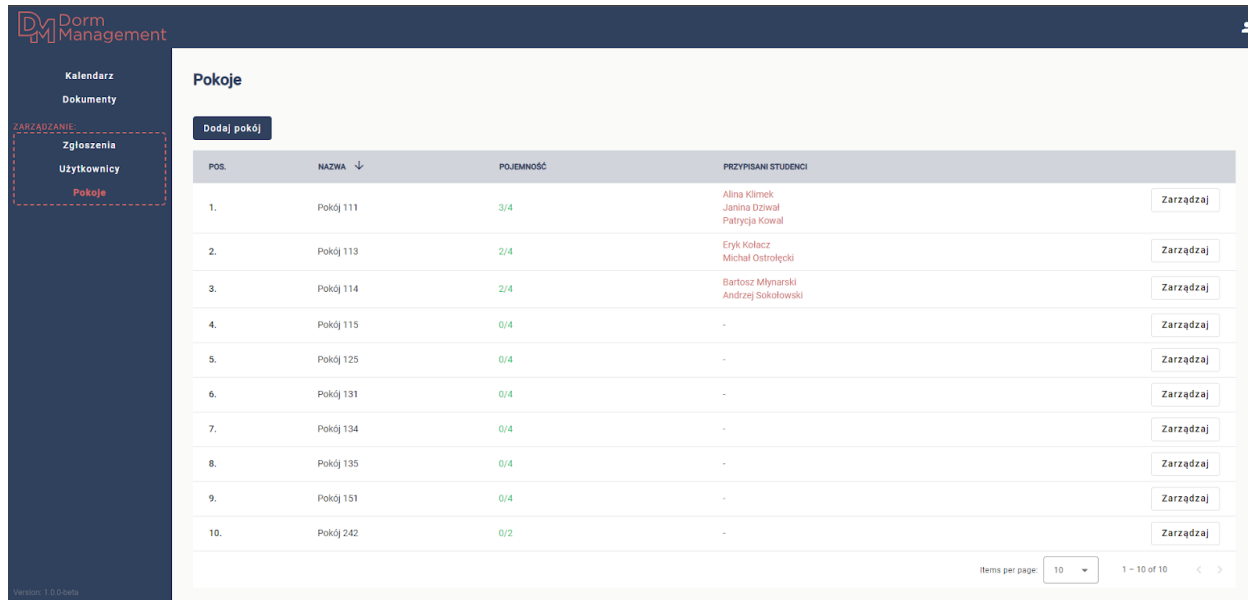


Angular Material



tailwindcss

Aplikacja frontend (2)



The screenshot shows the 'Pokoje' (Rooms) section of the 'Dorm Management' application. The interface includes a sidebar with navigation links: 'Kalendarz', 'Dokumenty', 'Zgłoszenia', 'Użytkownicy', and 'Pokoje' (highlighted). The main content area displays a table of rooms with columns for position, name, capacity, and assigned students. Each row has a 'Zarządzaj' (Manage) button. The table lists 10 rooms, with the last one having a capacity of 2/2.

POS.	NAZWA ↓	POJEMNOŚĆ	PRZYPISANI STUDENCI	
1.	Pokój 111	3/4	Alina Klimek Janina Dziwał Patrycja Kowal	Zarządzaj
2.	Pokój 113	2/4	Eryk Kolacz Michał Ostrołęcki	Zarządzaj
3.	Pokój 114	2/4	Bartosz Młynarski Andrzej Sokółowski	Zarządzaj
4.	Pokój 115	0/4	-	Zarządzaj
5.	Pokój 125	0/4	-	Zarządzaj
6.	Pokój 131	0/4	-	Zarządzaj
7.	Pokój 134	0/4	-	Zarządzaj
8.	Pokój 135	0/4	-	Zarządzaj
9.	Pokój 151	0/4	-	Zarządzaj
10.	Pokój 242	0/2	-	Zarządzaj

Items per page: 10 1 - 10 of 10

widok z aplikacji

```
getRooms({
  pageIndex,
  orderBy,
  pageSize,
}: RoomsParams): Observable<PaginatedData<Room>> {
  let queryParams = new HttpParams();
  queryParams = queryParams.append('orderBy', orderBy);
  queryParams = queryParams.append('pageIndex', pageIndex);
  queryParams = queryParams.append('pageSize', pageSize);
  return this.httpClient.get<PaginatedData<Room>>(
    `${environment.backendUrl}/rooms`,
    {
      params: queryParams,
      withCredentials: true,
    },
  );
}
```

room.service.ts

Aplikacja frontend (3)

```
private getDisplayedRooms() {
  this.roomService
    .getRooms({
      pageSize: this.pageSize,
      pageIndex: this.pageIndex,
      orderBy: `${this.orderBy} ${this.orderAsc ? 'asc' : 'desc'}`,
    })
    .pipe(
      tap(() => {
        this.loading = true;
      }),
    )
    .subscribe({
      next: (rooms: PaginatedData<Room>) => {
        this.rooms = rooms.data || [];
        this.length = rooms.totalRecords;
        this.loading = false;
      },
      error: () => {
        this.rooms = [];
        this.loading = false;
      },
    });
}
```

rooms-table.component.ts

```
<tbody>
  <tr
    *ngFor="let room of rooms; let i = index"
    class="bg-white border-b whitespace-nowrap"
    [class.text-accent]="currentUser.id === room.id"
  >
    <th scope="row" class="px-6 py-2 font-medium">{{ pageIndex * pageSize + i + 1 }}</th>
    <td class="px-6 py-2">{{ room.name }}</td>
    <td
      class="px-6 py-2"
      [class.text-green-500]="room.users.length < room.capacity"
    >{{ room.users.length }}/{{ room.capacity }}</td>
    <td class="px-6 py-2">
      <div
        *ngFor="let user of room.users"
        class="text-accent hover:underline cursor-pointer"
        [routerLink]="'/users/' + user.id"
      >{{ user.firstname }} {{ user.lastname }}</div>
      <p *ngIf="!room.users.length" class="my-2 text-gray-500">-</p>
    </td>
    <td class="px-6 py-2 flex justify-end">
      <button mat-stroked-button [routerLink]="'/rooms/' + room.id">
        Zarządzaj
      </button>
    </td>
  </tr>
  <tr *ngIf="!rooms?.length && !loading" class="bg-white border-b whitespace-nowrap">
    <td class="px-6 py-2 text-center text-gray-500 italic" colspan="6">
      <p class="my-6">Nie znaleziono pokoi</p>
      
    </td>
  </tr>
  <tr *ngIf="loading" class="bg-white border-b whitespace-nowrap">
    <td class="px-6 py-2 text-center text-gray-500 italic" colspan="6">
      <mat-progress-spinner class="mx-auto my-6" mode="indeterminate"></mat-progress-spinner>
    </td>
  </tr>
</tbody>
```

rooms-table.component.html

Aplikacja backend



```

let dataQuery = `SELECT u.id, u.email, u.last_login, u.role,
u.firstname, u.lastname, u.birthDate, u.birthPlace, u.pesel, u.phoneNumber, u.addressId, u.schoolId,
u.type, u.status, u.class, u.profile, u.parentIds, u.childIds, u.roomId, a.postalCode, a.city,
a.street, a.houseNo, a.apartmentNo, a.county, a.voivodeship, s.name FROM users u
LEFT JOIN addresses a ON u.addressId = a.id LEFT JOIN schools s ON u.schoolId = s.id`

let searchTermWhereClause = ''

if (searchTerm) {
  let dbSearchTermArray = searchTerm.split(' ')

  searchTermWhereClause = dbSearchTermArray
    .map((item) => {
      return `u.firstname='${item}' OR u.lastname='${item}' OR u.email='${item}'`
    })
    .join(' OR ')
}

if (role && searchTerm) {
  dataQuery += ` WHERE u.role=${role} AND (${searchTermWhereClause})`
} else if (role) {
  dataQuery += ` WHERE u.role=${role}`
} else if (searchTerm) {
  dataQuery += ` WHERE ${searchTermWhereClause}`
}

dataQuery += ` ORDER BY ${orderBy} LIMIT ${pageSize} OFFSET ${offset};`

const results = await new Promise((resolve, reject) => {
  db.query(dataQuery, function (error, results) {
    if (error) {
      logger.error(
        `SQL Error! Fetching data failed: ${error.message}`,
      )
      reject(error)
    } else {
      resolve(results)
    }
  })
})

```

```

db.query(`INSERT INTO rooms SET ?`, body, function (error, results) {
  if (error) {
    logger.error(`SQL error! Resource creation failed: ${error.message}`);
    return res.status(500).send({ msg: "An error occurred while creating the resource" });
  }

  const resourceId = results.insertId;

  logger.info("Resource creation successful.");
  res.status(201).send({id: resourceId})
});

```

```

const updateQuery = `UPDATE rooms SET ? WHERE id = ?`;

db.query(updateQuery, [updatedSet, id], function (error, results) {
  if (error) {
    logger.error(`SQL Error! Updating failed: ${error.message}`);
    return res
      .status(500)
      .send({ msg: 'An error occurred while updating the row' });
  }

  if (results.affectedRows === 0) {
    logger.warn("No rows were updated. Row with id ${id} not found.");
    return res.status(404).send({ msg: 'Row not found' });
  }
}

```

```

db.query(`DELETE FROM rooms WHERE id = ${id}`, function (error, results) {
  if (error) {
    logger.error(`SQL error! Deleting operation failed: ${error.message}.`)
    return res.status(500).send({msg: "An error occurred while deleting the row"});
  }

  if (results.affectedRows === 0) {
    logger.warn("No rows were deleted. Row with id ${id} not found.");
    return res.status(404).send({ msg: "Row not found" });
  }

  logger.info("Data deleted successfully.")
  return res.status(200).send({ msg: "Row deleted successfully"});
});

```

Dalszy rozwój

- Powiadomienia dla użytkowników
- Przeglądy i kontrole obiektów
- Skargi i wnioski
- Rada pedagogiczna
- Wsparcie językowe
- Tryb ciemny

Podsumowanie

Projekt dyplomowy był szansą na wykorzystanie umiejętności nabytych nie tylko podczas okresu studiów pierwszego stopnia, ale również tych zdobytych pracując już w zawodzie. Jej owocem jest przedstawiony system.

Implementacja oraz wdrożenie aplikacji odbyły się w oparciu o najnowsze technologie. System w postaci responsywnej i bezpiecznej aplikacji internetowej oferuje funkcje:

- Cyfryzacji procesu składania oraz akceptowania podań do akademików.
- Bezpieczeństwa procesu logowania i zarządzanie kontami użytkowników.
- Zarządzania pokojami, w tym dodawanie, edycja i usuwanie.
- Zarządzania wydarzeniami w obiekcie i udostępnianie ich użytkownikom.
- Udostępniania, edycji i zarządzania dokumentami w formatach .pdf i .docx.

Każda z oferowanych funkcjonalności pozwala na zwiększenie efektywności oraz bezpieczeństwa procesów administracyjnych.