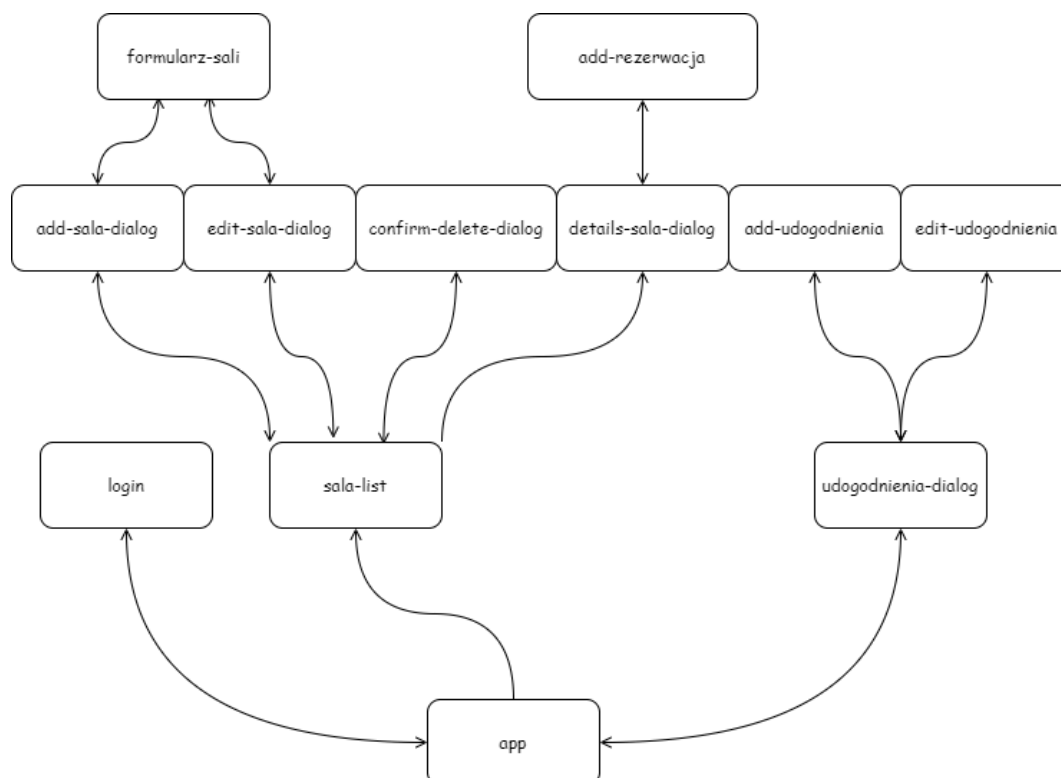


Wydział Informatyki Politechniki Białostockiej Aplikacje internetowe oparte o komponenty	Data oddania: 12.12.2024 r.
Pracownia specjalistyczna nr 4 Zespół: 1. Filip Białokozowicz 2. Kacper Przemielewski 3. Kacper Przeździecki	Prowadzący: dr inż. Urszula Kuźlewska

1. Opis projektu

Stworzenie prostej aplikacji webowej w Angularze umożliwiającej zarządzanie salami do wynajęcia. Aplikacja pozwala na dodawanie nowych sal, edycję istniejących, usuwanie sal z listy oraz przeglądanie dostępnych sal wraz z podstawowymi informacjami.

2. Wizualizacja architektury komponentów wraz z nazwami i typami przekazywanych elementów



Komponenty i przekazywane elementy

- AppComponent
 - Używa AuthService
 - Routuje do SalaListComponent, LoginComponent, AddRezerwacjaComponent
- SalaListComponent
 - Używa SalaService, AuthService
 - Otwiera dialogi: AddSalaDialogComponent, EditSalaDialogComponent, DetailsSalaDialogComponent, ConfirmDeleteDialogComponent, UdogodnieniaDialogComponent
- AddSalaDialogComponent
 - Zawiera FormularzSaliComponent
- EditSalaDialogComponent
 - Używa UdogodnieniaService
 - Zawiera FormularzSaliComponent
- DetailsSalaDialogComponent
 - Wejście: Sala
- ConfirmDeleteDialogComponent
 - Wejście: Sala
- UdogodnieniaDialogComponent
 - Używa UdogodnieniaService, AuthService
 - Otwiera dialogi: AddUdogodnienieDialogComponent, EditUdogodnienieDialogComponent
- AddUdogodnienieDialogComponent
 - Używa UdogodnieniaService
- EditUdogodnienieDialogComponent
 - Używa UdogodnieniaService
 - Wejście: Udogodnienie
- FormularzSaliComponent
 - Wejście: Sala, Udogodnienie[]
 - Wyjście: Sala
- AddRezerwacjaComponent
 - Wejście: Sala
 - Wyjście: Rezerwacja

Serwisy

- AuthService
- SalaService
- UdogodnieniaService

3. Struktura klas

Sala:

- **id: number** – unikalny identyfikator sali
- **nazwa: string** – nazwa sali
- **pojemnosc: number** – maksymalna liczba osób, jaka może przebywać w sali
- **udogodnienia: Udogodnienie[]** – tablica obiektów typu **Udogodnienie** określająca dostępne udogodnienia w sali
- **rezerwacje: Rezerwacja[]** – tablica obiektów typu **Rezerwacja** określająca aktualne i przyszłe rezerwacje sali

Rezerwacje

- **id: number** – unikalny identyfikator rezerwacji
- **imie: string** – imię osoby dokonującej rezerwację
- **nazwisko: string** – nazwisko osoby dokonującej rezerwację
- **email: string** – adres email osoby dokonującej rezerwację
- **startDateTime: Date** – data i godzina rozpoczęcia rezerwacji
- **endDateTime: Date** – data i godzina zakończenia rezerwacji

Udogodnienia

- **id: number** – unikalny identyfikator udogodnienia
- **nazwa: string** – nazwa udogodnienia (np. projektor, klimatyzacja)
- **opis: string** – dodatkowy opis udogodnienia

4. Spis funkcjonalności i sposób wykonania

1. Wyświetlanie listy sal

SalaList:

Możliwość przeglądania listy dostępnych sal z podstawowymi informacjami.

```
<table mat-table [dataSource]="sale" class="mat-elevation-z8" style="width: 100%;">

  <!-- ID Column -->
  <ng-container matColumnDef="id" @if(authService.isAdminLoggedIn())>
    <th mat-header-cell *matHeaderCellDef > ID </th>
    <td mat-cell *matCellDef="let sala"> {{ sala.id }} </td>
  </ng-container>

  <!-- Nazwa Column -->
  <ng-container matColumnDef="nazwa">
    <th mat-header-cell *matHeaderCellDef> Nazwa </th>
    <td mat-cell *matCellDef="let sala"> {{ sala.nazwa }} </td>
  </ng-container>

  <!-- Pojemność Column -->
  <ng-container matColumnDef="pojemnosc">
    <th mat-header-cell *matHeaderCellDef> Pojemność </th>
    <td mat-cell *matCellDef="let sala"> {{ sala.pojemnosc }} </td>
  </ng-container>

  <!-- Udogodnienia Column -->
  <ng-container matColumnDef="udogodnienia">
    <th mat-header-cell *matHeaderCellDef> Udogodnienia </th>
    <td mat-cell *matCellDef="let sala">
      {{ getUdogodnieniaNames(sala.udogodnienia) }}
    </td>
  </ng-container>
</table>
```

2. Dodawanie nowej sali

SalaList + AddSalaDialog + FormularzSali:

Możliwość dodania sali do systemu przez administ

```
openCreateDialog(): void {
  const dialogRef = this.dialog.open(AddSalaDialogComponent, {
    width: '600px',
  });

  dialogRef.afterClosed().subscribe((result: Sala | undefined) => {
    if (result) {
      this.salaService.addSala(result);
    }
  });
}
```

```
<h2 mat-dialog-title>{{ 'Dodaj Nową Salę' }}</h2>
<mat-dialog-content>
  <app-formularz-sali
    [udogodnienia]="udogodnienia"
    (submitSala)="onSalaSubmit($event)">
  </app-formularz-sali>
</mat-dialog-content>
<mat-dialog-actions align="end">
  <button mat-button (click)="onCancel()">Anuluj</button>
</mat-dialog-actions>
```

```
ngOnInit(): void {
  this.form = this.fb.group({
    id: [this.sala?.id || null],
    nazwa: [this.sala?.nazwa || '', [Validators.required, Validators.minLength(3)]],
    pojemnosc: [this.sala?.pojemnosc || null, [Validators.required, Validators.min(1)]],
    udogodnienia: [this.sala?.udogodnienia.map((u) => u.id) || []],
  });
}

onSubmit(): void {
  const formValue = this.form.value;
  const sala: Sala = {
    id: formValue.id,
    nazwa: formValue.nazwa,
    pojemnosc: formValue.pojemnosc,
    udogodnienia: this.udogodnienia.filter((u) =>
      formValue.udogodnienia.includes(u.id)
    ),
  };
  this.submitSala.emit(sala);
}
```

3. Edycja istniejącej sali

EditSalaDialog + FormularzSali:

Możliwość edycji istniejącej sali przez administratora

```
Go to component
<h2 mat-dialog-title>{{'Edytuj Salę'}}</h2>
<mat-dialog-content>
  <app-formularz-sali
    [sala]="data.sala"
    [udogodnienia]="udogodnienia"
    (submitSala)="onSalaSubmit($event)"></app-formularz-sali>
</mat-dialog-content>
<mat-dialog-actions align="end">
  <button mat-button (click)="onCancel()">Anuluj</button>
</mat-dialog-actions>
```

4. Usuwanie sali

SalaList:

Możliwość usunięcia sali przez administratora

```
openDeleteDialog(sala: Sala): void {
  const dialogRef = this.dialog.open(ConfirmDeleteDialogComponent, {
    width: '400px',
    data: {sala}
  });

  dialogRef.afterClosed().subscribe(result => {
    if (result) {
      this.salaService.deleteSala(sala.id);
    }
  });
}
```

5. Szczegóły sali

DetailsSala:

Możliwość wyświetlenia szczegółów sali wraz z dokonanymi rezerwacjami.

```
<h2 mat-dialog-title>{{sala.nazwa}}</h2>
<mat-dialog-content>
  <ng-container>
    <p><strong>ID:</strong> {{ sala.id }}</p>
  </ng-container>
  <p><strong>Nazwa:</strong> {{ sala.nazwa }}</p>
  <p><strong>Pojemność:</strong> {{ sala.pojemnosc }}</p>
  <p><strong>Udogodnienia:</strong> {{ getUdogodnieniaNames(sala.udogodnienia) }}</p>
  <p><strong>Rezerwacje:</strong></p>
  <table mat-table [dataSource]="rezerwacje" class="rezerwacje-table">
    <!-- ID Column -->
    @if (authService.isAdminLoggedIn()) {
      <ng-container matColumnDef="id">
        <th mat-header-cell *matHeaderCellDef>ID</th>
        <td mat-cell *matCellDef="let rezerwacja">{{ rezerwacja.id }}</td>
      </ng-container>
    }
    @if (authService.isAdminLoggedIn()){
      <!-- Imię Column -->
      <ng-container matColumnDef="imie">
        <th mat-header-cell *matHeaderCellDef>Imię</th>
        <td mat-cell *matCellDef="let rezerwacja">{{ rezerwacja.imie }}</td>
      </ng-container>
    }
    @if (authService.isAdminLoggedIn()){
      <!-- Nazwisko Column -->
      <ng-container matColumnDef="nazwisko">
        <th mat-header-cell *matHeaderCellDef>Nazwisko</th>
        <td mat-cell *matCellDef="let rezerwacja">{{ rezerwacja.nazwisko }}</td>
      </ng-container>
    }
    @if (authService.isAdminLoggedIn()){
      <!-- Email Column -->
      <ng-container matColumnDef="email">
        <th mat-header-cell *matHeaderCellDef>Email</th>
        <td mat-cell *matCellDef="let rezerwacja">{{ rezerwacja.email }}</td>
      </ng-container>
    }
    <!-- Start Date Column -->
    <ng-container matColumnDef="startDateTime">
      <th mat-header-cell *matHeaderCellDef>Data rozpoczęcia</th>
      <td mat-cell *matCellDef="let rezerwacja">{{ rezerwacja.startDateTime | date: format: 'short' }}</td>
    </ng-container>
    <!-- End Date Column -->
    <ng-container matColumnDef="endDateTime">
      <th mat-header-cell *matHeaderCellDef>Data zakończenia</th>
      <td mat-cell *matCellDef="let rezerwacja">{{ rezerwacja.endDateTime | date: format: 'short' }}</td>
    </ng-container>
  </table>
  @if (authService.isAdminLoggedIn()){
```

6. Dodanie rezerwacji

Możliwość dodania rezerwacji sali.

```
addRezerwacja(rezerwacja: Rezerwacja): Rezerwacja { Show usages  qer24 *
  const currentRezerwacje :Rezerwacja[] = this.getRezerwacje();
  const newId :number = currentRezerwacje.length > 0
    ? Math.max(...currentRezerwacje.map((r :Rezerwacja ) :number => r.id)) + 1
    : 1;

  const newRezerwacja = new Rezerwacja(
    newId,
    rezerwacja.imie,
    rezerwacja.nazwisko,
    rezerwacja.email,
    rezerwacja.startDateTime,
    rezerwacja.endDateTime
  );

  const updatedRezerwacje :Rezerwacja[] = [...currentRezerwacje, newRezerwacja];
  this.rezerwacjeSubject.next(updatedRezerwacje);
  this.saveRezerwacjeToStorage(updatedRezerwacje);

  return newRezerwacja;
}
```

7. Usunięcie rezerwacji

Możliwość usunięcia sali przez administratora.

```
deleteRezerwacja(id: number): void { Show usages  qer24
  const currentRezerwacje :Rezerwacja[] = this.getRezerwacje();
  const updatedRezerwacje :Rezerwacja[] = currentRezerwacje.filter(rezerwacja :Rezerwacja => rezerwacja.id !== id);
  this.rezerwacjeSubject.next(updatedRezerwacje);
  this.saveRezerwacjeToStorage(updatedRezerwacje);
}
```

8. Sortowanie listy sal i rezerwacji

Możliwość sortowania rosnąco lub malejąco po id, nazwie, pojemności. Rezerwacje są filtrowane po dacie startu rezerwacji.

```
private filterRezerwacjeBySala(rezerwacje: Rezerwacja[]): Rezerwacja[] { Show usages
    if (!this.sala || !this.sala.rezerwacje) {
        return [];
    }

    const salaRezerwacje :Rezerwacja[] = rezerwacje.filter(rezerwacja :Rezerwacja =>
        this.sala.rezerwacje.some(r :Rezerwacja => r.id === rezerwacja.id)
    );

    // Sortujemy rezerwacje według daty rozpoczęcia od najwcześniejszej
    salaRezerwacje.sort((a :Rezerwacja , b :Rezerwacja ) :number => {
        return new Date(a.startDate.getTime() - new Date(b.startDate.getTime());
    });

    return salaRezerwacje;
}
```

9. Wyszukanie sali

Możliwość wyszukania sali wprowadzając tekst (jedno okno wyszukuje po nazwie i pojemności).

5. Walidacja wprowadzanych danych

Sprawdzanie poprawności danych w formularzach dodawania i edycji. Sprawdzanie poprawności wpisanych dat rezerwacji (początek rezerwacji nie może być później niż koniec) i czy wpisana data nie koliduje z już istniejącą rezerwacją.

Walidacja logowania:

```
this.form = this.fb.group({
    username: ['', [Validators.required, Validators.minLength(3)]],
    password: ['', [Validators.required, Validators.pattern(/^(?=.*[A-Za-z])(?=.*\d)[A-Za-z\d]{8,}$/)]
});
```


Walidacja tworzenia rezerwacji:

```
constructor( Show usages  qer24 +1
  private fb: FormBuilder,
  private rezerwacjeService: RezerwacjeService,
  public dialogRef: MatDialogRef<AddRezerwacjaDialogComponent>,
  @Inject(MAT_DIALOG_DATA) public data: { sala: Sala }
) {
  this.rezerwacjaForm = this.fb.group(
    {
      imie: ['', Validators.required],
      nazwisko: ['', Validators.required],
      email: ['', [Validators.required, Validators.email]],
      startDate: ['', Validators.required],
      startTime: ['', Validators.required],
      endDate: ['', Validators.required],
      endTime: ['', Validators.required],
    },
    {
      validators: [this.dateRangeValidator, this.conflictValidator]
    }
  );
}
```

metoda dateRangeValidator (walidacja dat):

```
private dateRangeValidator(group: FormGroup): { [key: string]: any } | null {
  const startDate :any = group.get('startDate')?.value;
  const startTime :any = group.get('startTime')?.value;
  const endDate :any = group.get('endDate')?.value;
  const endTime :any = group.get('endTime')?.value;

  if (!startDate || !startTime || !endDate || !endTime) {
    return null;
  }

  const startDateTime = new Date(startDate);
  const [startHours, startMinutes] = startTime.split(':');
  startDateTime.setHours(startHours, startMinutes);

  const endDateTime = new Date(endDate);
  const [endHours, endMinutes] = endTime.split(':');
  endDateTime.setHours(endHours, endMinutes);

  return endDateTime >= startDateTime ? null : { invalidDateRange: true };
}
```

metoda conflictValidator (sprawdzająca wybrany termin):

```
private conflictValidator : (group: FormGroup<any>) => {[key: string]: any } | null => {
  const startDate :any = group.get('startDate')?.value;
  const startTime :any = group.get('startTime')?.value;
  const endDate :any = group.get('endDate')?.value;
  const endTime :any = group.get('endTime')?.value;

  if (!startDate || !startTime || !endDate || !endTime) {
    return null;
  }

  const startDateTime = new Date(startDate);
  const [startHours, startMinutes] = startTime.split(':');
  startDateTime.setHours(+startHours, +startMinutes);

  const endDateTime = new Date(endDate);
  const [endHours, endMinutes] = endTime.split(':');
  endDateTime.setHours(+endHours, +endMinutes);

  const salaRezerwacje :Rezerwacja[] = this.data.sala.rezerwacje || [];

  const conflictingRezerwacje :Rezerwacja[] = salaRezerwacje.filter((r :Rezerwacja ) :boolean => {
    const rStart = new Date(r.startDateTime);
    const rEnd = new Date(r.endDateTime);
    return (
      (startDateTime >= rStart && startDateTime < rEnd) ||
      (endDateTime > rStart && endDateTime <= rEnd) ||
      (startDateTime <= rStart && endDateTime >= rEnd)
    );
  });

  return conflictingRezerwacje.length === 0 ? null : { conflict: true };
};
```

Walidacja tworzenia i edytowania sali:

```
this.form = this.fb.group({
  id: [this.sala?.id || null],
  nazwa: [this.sala?.nazwa || '', [Validators.required, Validators.minLength(3)]],
  pojemnosc: [this.sala?.pojemnosc || null, [Validators.required, Validators.min(1)]],
  udogodnienia: [this.sala?.udogodnienia.map((u) => u.id) || []],
});
```

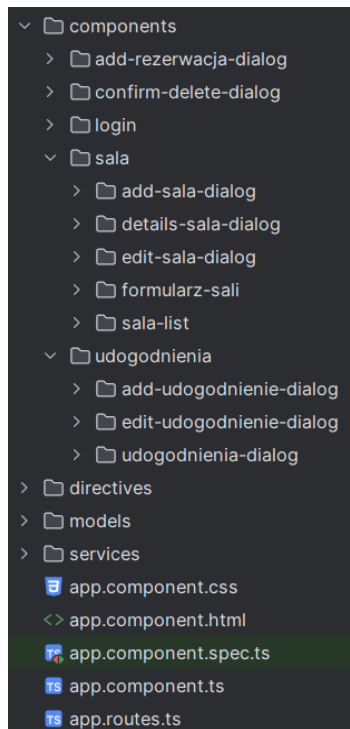
Walidacja tworzenia i edytowania udogodnienia:

```
this.udogodnienieForm = this.fb.group({
  nazwa: ['', Validators.required],
  opis: ['', Validators.required]
});
```

6. Elementy frameworka

1. Własne komponenty

Stworzono wiele komponentów do wielokrotnego użytku:



2. Dyrektywy z frameworka

W wielu miejscach używamy dyrektyw z frameworka (np. if, for, multiple, matInput, FormControlName).

```
<form [formGroup]="form" (ngSubmit)="onSubmit()">
  <!-- Nazwa Sali -->
  <mat-form-field appearance="fill" style="...">
    <mat-label>Nazwa sali</mat-label>
    <input matInput type="text" FormControlName="nazwa" />
    @if(form.get('nazwa')?.hasError( errorCode: 'required')) {
      <mat-error>Nazwa sali jest wymagana.</mat-error>
    }
    @if(form.get('nazwa')?.hasError( errorCode: 'minLength')) {
      <mat-error>Nazwa sali musi mieć przynajmniej 3 znaki.</mat-error>
    }
  </mat-form-field>

  <!-- Pojemność -->
  <mat-form-field appearance="fill" style="...">
    <mat-label>Pojemność</mat-label>
    <input matInput type="number" FormControlName="pojemnosc" />
    @if(form.get('pojemnosc')?.hasError( errorCode: 'required')) {
      <mat-error>Pojemność jest wymagana.</mat-error>
    }
    @if(form.get('pojemnosc')?.hasError( errorCode: 'min')) {
      <mat-error>Pojemność musi być przynajmniej 1.</mat-error>
    }
  </mat-form-field>

  <!-- Udogodnienia -->
  <mat-form-field appearance="fill">
    <mat-label>Udogodnienia</mat-label>
    <mat-select multiple FormControlName="udogodnienia">
      @for (udogodnienie of udogodnienia; track udogodnienie.id) {
        <mat-option [value]="udogodnienie.id">
          {{ udogodnienie.nazwa }}
        </mat-option>
      }
    </mat-select>
  </mat-form-field>

  <!-- Submit Button -->
  <button type="submit">Submit</button>
</form>
```

3. Walidacja wprowadzonych danych ([wyżej](#))

4. Komunikacja pomiędzy komponentami

Przekazywanie danych i zdarzeń między komponentami.

1. Parent to Child Communication using `@Input()`

From FormularzSaliComponent:

```
@Component({/*...*/})
export class FormularzSaliComponent {
  @Input() sala?: Sala; // Receives sala data from parent
  @Input() udogodnienia: Udogodnienie[] = []; // Receives udogodnienia list
}
```

2. Child to Parent Communication using `@Output()`

From FormularzSaliComponent:

```
export class FormularzSaliComponent {
  @Output() submitSala: EventEmitter<Sala> = new EventEmitter<Sala>();

  onSubmit(): void {
    const sala: Sala = {/*...*/};
    this.submitSala.emit(sala);
  }
}
```

3. Service-based Communication

Using `UdogodnieniaService` to share data between components:

```
@Injectable({
  providedIn: 'root'
})
export class UdogodnieniaService {
  private udogodnieniaSubject: BehaviorSubject<Udogodnienie[]>;
  public udogodnienia$ = this.udogodnieniaSubject.asObservable();

  updateUdogodnienie(updatedUdogodnienie: Udogodnienie): void {
    const currentUdogodnienia = this.getUdogodnienia();
    const updatedUdogodnienia = currentUdogodnienia.map(u =>
      u.id === updatedUdogodnienie.id ? updatedUdogodnienie : u
    );
    this.udogodnieniaSubject.next(updatedUdogodnienia);
  }
}
```

Subscribing to updates in UdogodnieniaDialogComponent:

```
export class UdogodnieniaDialogComponent {
  constructor(private udogodnieniaService: UdogodnieniaService) {
    this.udogodnieniaService.udogodnienia$.subscribe(data => {
      this.udogodnienia = data;
    });
  }
}
```

4. Dialog Communication

Using MatDialog for component interaction in SalaListComponent:

```
export class SalaListComponent {
  openEditDialog(sala: Sala): void {
    const dialogRef = this.dialog.open(EditSalaDialogComponent, {
      width: '600px',
      data: {sala}
    });

    dialogRef.afterClosed().subscribe((result: Sala | undefined) => {
      if (result) {
        this.salaService.updateSala(result);
      }
    });
  }
}
```

5. Własna dyrektywa, walidator ([wyżej](#)), filtr ([wyżej](#))

Stworzenie dyrektywy zmieniającej wygląd elementu.

```
@Directive({
  selector: '[zmienKolor]',
  standalone: true,
})
export class ZmienKolorDyrektywa {
  constructor(
    private el: ElementRef,
    private renderer: Renderer2
  ) {}

  @HostListener('mouseenter') onMouseEnter() {
    this.renderer.setStyle(
      this.el.nativeElement,
      'backgroundColor',
      '#e0e0e0' // Light gray color when hovered
    );
  }

  @HostListener('mouseleave') onMouseLeave() {
    this.renderer.setStyle(
      this.el.nativeElement,
      'backgroundColor',
      'transparent'
    );
  }
}
```

6. Formularze reaktywne z różnymi elementami

Użycie różnych typów pól w formularzach.

2. Multiple Select (from FormularzSalComponent):

```
<mat-form-field>
  <mat-label>Udogodnienia</mat-label>
  <mat-select multiple formControlName="udogodnienia">
    <mat-option *ngFor="let udogodnienie of udogodnienia" [value]="udogodnienie.id">
      {{ udogodnienie.nazwa }}
    </mat-option>
  </mat-select>
</mat-form-field>
```

```
this.form = this.fb.group({
  udogodnienia: [this.sala?.udogodnienia.map((u) => u.id) || []]
});
```

3. Email Input with Validation (from FormularzRezerwacjiComponent):

```
<mat-form-field>
  <mat-label>Email</mat-label>
  <input matInput type="email" formControlName="email" />
  @if(form.get('email')?.hasError('required')) {
    <mat-error>Email jest wymagany.</mat-error>
  }
  @if(form.get('email')?.hasError('email')) {
    <mat-error>Wprowadź poprawny adres email.</mat-error>
  }
</mat-form-field>
```

```
this.form = this.fb.group({
  email: ['', [Validators.required, Validators.email]]
});
```

4. Date-Time Picker (from FormularzRezerwacjiComponent):

```
<mat-form-field>
  <mat-label>Data i godzina rozpoczęcia</mat-label>
  <input matInput [ngxMatDatetimePicker]="startPicker" formControlName="startDateTime" />
  <ngx-mat-datetime-picker #startPicker></ngx-mat-datetime-picker>
</mat-form-field>
```

```
this.form = this.fb.group({
  startDateTime: [null, Validators.required],
  endDateTime: [null, Validators.required]
}, {
  validators: [this.endDateAfterStartDateValidator]
});
```

5. Password Input with Pattern Validation (from LoginComponent):

```
<input id="password" type="password" formControlName="password" />
```

```
this.form = this.fb.group({  
  password: ['', [Validators.required, Validators.pattern(/^(?=.*[A-Za-z])(?=.*\d)[A-Za-z\d]{8,}$/)]]  
});
```

5. Instrukcja użytkownika

Użytkownik bez zalogowania widzi listę sal

[Lista Sal](#) [Zaloguj](#)

Lista Sal

[Przeglądaj Udogodnienia](#)

Szukaj

Nazwa	Pojemność	Udogodnienia	Akcje
Sala Konferencyjna	201	WiFi, Kamera	
Sala 1	5	Projektor, WiFi, Klimatyzacja, Tablica, Głośniki, Mikrofon, Laptop, Telewizor, Flipchart, Kamera, Biała tablica	
sdaad	3244321412	WiFi	
dasdsa	231	Flipchart	
dasdsa	1234	Brak udogodnień	

Items per page: 5 1 - 5 of 6 |< < > >|

Może wyszukać interesującą go salę po nazwie lub pojemności wpisując odpowiedni tekst w okno wyszukiwania

Szukaj
Sala

Nazwa	Pojemność	Udogodnienia	Akcje
Sala Konferencyjna	201	WiFi, Kamera	
Sala 1	5	Projektor, WiFi, Klimatyzacja, Tablica, Głośniki, Mikrofon, Laptop, Telewizor, Flipchart, Kamera, Biała tablica	

Items per page: 5 1 - 2 of 2 |< < > >|

Szukaj
201

Nazwa	Pojemność	Udogodnienia	Akcje
Sala Konferencyjna	201	WiFi, Kamera	

Items per page: 5 1 - 1 of 1 |< < > >|

Może zmienić ilość wyświetlanych elementów na liście zmieniając wartość u dołu listy i przechodzić między stronami.

Szukaj

Nazwa	Pojemność	Udogodnienia	Akcje
Sala Konferencyjna	201	WiFi, Kamera	
Sala 1	5	Projektor, WiFi, Klimatyzacja, Tablica, Głośniki, Mikrofon, Laptop, Telewizor, Flipchart, Kamera, Biała tablica	
sdaad	3244321412	WiFi	
dasdsa	231	Flipchart	
dasdsa	1234	Brak udogodnień	
advsvxz	42	Brak udogodnień	

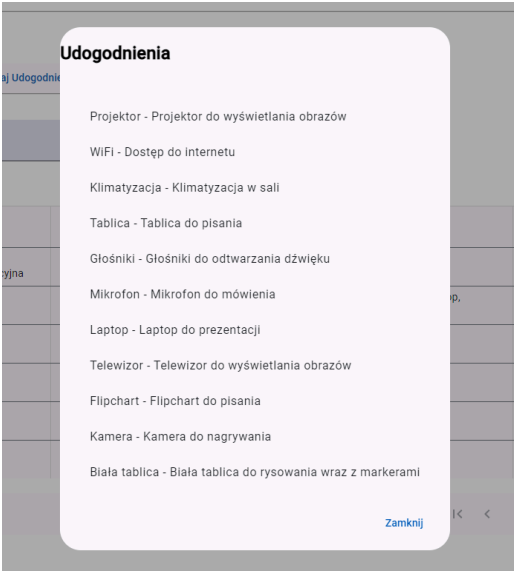
Items per page: 10

1 - 6 of 6

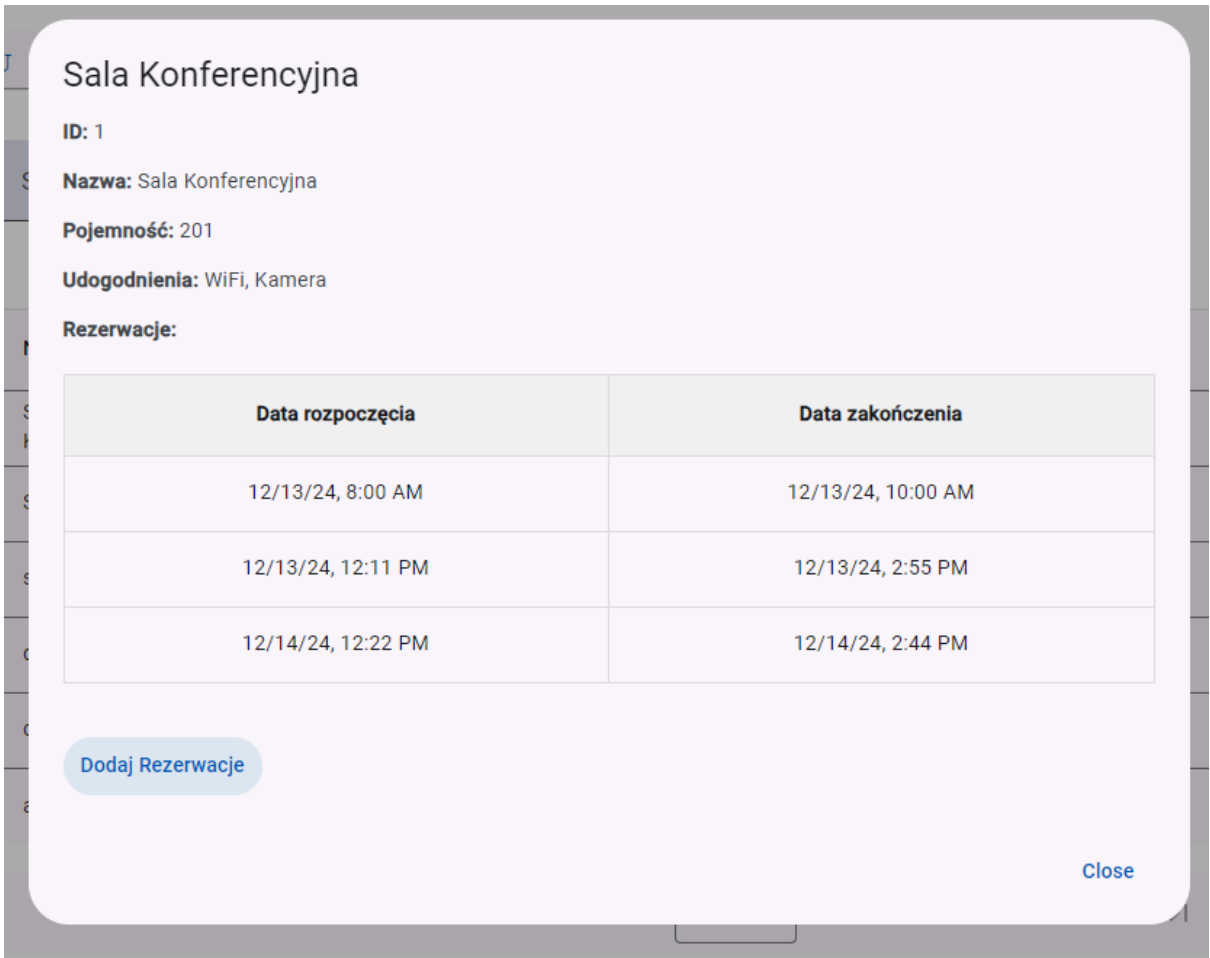
W celu sortowania sal (np. po pojemności) należy kliknąć nazwę kolumny (przycisk jest widoczny ze względu na zmianę koloru tła)

Nazwa	Pojemność ↑	Udogodnienia
Sala 1	5	Projektor, WiFi, Klimatyzacja,
advsvxz	42	Brak udogodnień
Sala Konferencyjna	201	WiFi, Kamera
dasdsa	231	Flipchart
dasdsa	1234	Brak udogodnień

Po wciśnięciu przycisku *Przeglądaj Udogodnienia* wyskakuje okno z możliwymi udogodnieniami.



Po wciśnięciu “oka” w wierszu sali wyskakuje okno ze szczegółami sali i rezerwacjami



Użytkownik może dodać rezerwację poprzez wciśnięcie przycisku *Dodaj Rezerwację* i uzupełnienie wszystkich pól (datę można wybrać z kalendarza) oraz zatwierdzeniu przyciskiem *Dodaj*

The image shows two screenshots of a web application for adding a reservation. The left screenshot displays a form titled 'Dodaj Rezerwację' with the following fields: 'Imię*' (Kacper), 'Nazwisko*' (P), 'Email*' (przyklad@przyklad.pl), 'Data rozpoczęcia*' (12/19/2024), 'Czas rozpoczęcia*' (15:00), 'Data zakończenia*' (12/26/2024), and 'Czas zakończenia*' (10:00). The right screenshot shows a calendar overlay for December 2024, with the 26th selected, and the 'Dodaj' button highlighted.

(rezerwacja została dodana pomyślnie)

12/14/24, 12:22 PM	12/14/24, 2:44 PM
12/19/24, 3:00 PM	12/26/24, 10:00 AM

W celu zalogowania się administratora należy wcisnąć przycisk *Zaloguj* w lewym górnym rogu strony. Przejdziemy wtedy do strony logowania. Należy wprowadzić dane logowania (admin, admin123)

The image shows a login page with a blue 'Zaloguj' button in the top left. Below it are input fields for 'Username' (admin) and 'Password' (masked with dots), and a green 'Login' button at the bottom.

Powrócimy wtedy do listy sal z dodatkowymi opcjami

Lista SalWyloguj

Lista Sal

+ Dodaj Salę

Przeglądaj Udogodnienia

Szukaj

ID	Nazwa	Pojemność	Udogodnienia	Akcje
1	Sala Konferencyjna	201	WiFi, Kamera	  
2	Sala 1	5	Projektor, WiFi, Klimatyzacja, Tablica, Głośniki, Mikrofon, Laptop, Telewizor, Flipchart, Kamera, Biała tablica	  
3	sdaad	3244321412	WiFi	  
5	dasdsa	231	Flipchart	  
6	dasdsa	1234	Brak udogodnień	  

Items per page: 51 - 5 of 6<>>|

Administrator może dodać salę poprzez wciśnięcie przycisku *Dodaj Salę*. Wskoczy wtedy formularz dodania sali. Należy uzupełnić wymagane pola (oznaczone gwiazdką *), udogodnienia nie są wymagane, ale chcą je dodać wyświetla się lista z możliwością zaznaczenia odpowiednich udogodnień). By zatwierdzić należy wcisnąć przycisk *Submit*.

Dodaj Nową Salę

Nazwa sali*

Sala Przykładowa

Pojemność*

50

Udogodnienia

Projektor, WiFi, Klim...

☒ Projektor

☒ WiFi

☒ Klimatyzacja







☐ Tablica

☒ Głośniki

☐ Mikrofon

Submit

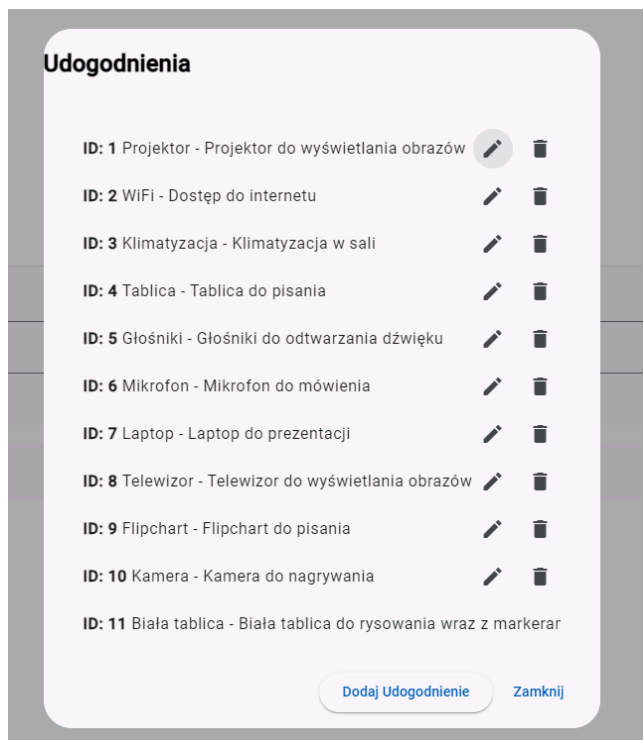
Anuluj

ID ↑	Nazwa	Pojemność	Udogodnienia	Akcje
7	advsvrxz	42	Brak udogodnień	  
8	Sala Przykładowa	50	Projektor, WiFi, Klimatyzacja, Głośniki, Biała tablica	  

Items per page: 56 - 7 of 7<>>|

(sala dodana pomyślnie)

Po wciśnięciu przycisku *Przeglądaj Udogodnienia* dla administratora wyświetlają się dodatkowe opcje.



Administrator może dodać Udogodnienie do listy poprzez Wciśnięcie przycisku *Dodaj Udogodnienie*. Wskoczy wtedy formularz dodania udogodnienia. Należy wprowadzić wymagane pola i zatwierdzić przyciskiem *Dodaj*.

Dodaj Udogodnienie

Nazwa*


Nazwa jest wymagana

Opis*

Cofnij Dodaj

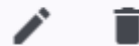
(wyskoczył błąd walidacyjny umożliwiający dodanie udogodnienia, takie błędy są wszędzie jeśli nie wprowadzimy wymaganych pól)

Możemy edytować wybrane udogodnienie wciskając ikonkę “ołówka” przy interesującym nas udogodnieniu. Wskoczy wtedy identyczne okno jak przy dodawaniu.

ID: 12 Stół Bilardowy - Stół do gry w bilarda + bile + kije 

W celu usunięcia należy wcisnąć ikonkę “kosza”. Udogodnienie zostanie wtedy usunięte z listy oraz we wszystkich salach w których było dane udogodnienie.

ID: 2 WiFi - Dostęp do internetu



Administrator może edytować sale wciskając ikonkę “ołówka” przy interesującej go sali. Wskoczy wtedy identyczne okno jak przy dodawaniu sali. Może tam zmienić nazwę, pojemność i udogodnienia w taki sam sposób jak przy tworzeniu sali.

Edytuj Salę

Nazwa sali*

Bardzo duża sala

Pojemność*

3244321412

Udogodnienia

Projektor, WiFi, Tabli...

☒ Projektor

☒ WiFi

☐ Klimatyzacja





☒ Tablica

☐ Głośniki

☐ Mikrofon

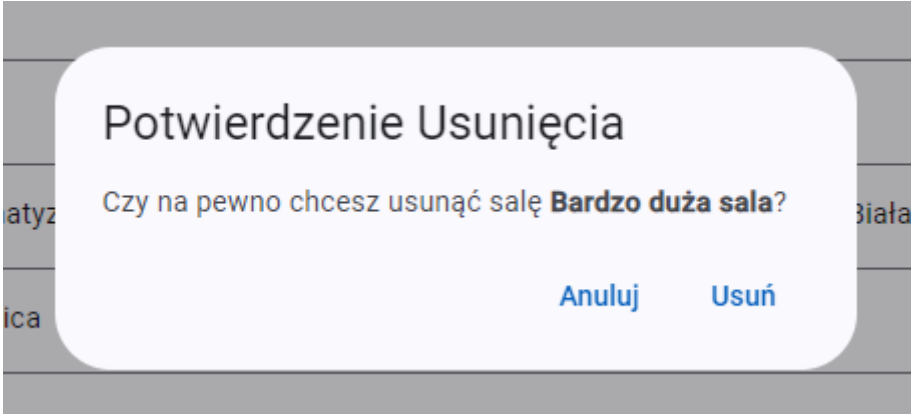
Submit

Anuluj

4	Salę 1	✓	Projektor, WiFi, Klimatyzacja, Tablica, Głośniki, Mikrofon, Karty, Internet, i inne usługi	 
3	Bardzo duża sala	3244321412	Projektor, WiFi, Tablica	 

(sala została zedytowana)

Aby usunąć salę należy wcisnąć ikonę “kosza” przy sali. Wskoczy wtedy okno potwierdzenia usunięcia, które należy potwierdzić przyciskiem *Usuń*.



ID	Nazwa	Pojemność	Udogodnienia	Akcje
1	Sala Konferencyjna	201	WiFi	  
2	Sala 1	5	Projektor, WiFi, Klimatyzacja, Tablica, Głośniki, Mikrofon, Laptop, Telewizor, Flipchart, Biała tablica	  
5	dasdsa	231	Flipchart	  
6	dasdsa	1234	Brak udogodnień	  
7	advsvxz	42	Brak udogodnień	  
8	Sala Przykładowa	50	Projektor, WiFi, Klimatyzacja, Głośniki, Biała tablica	  

(sala została usunięta)

Administrator po wciśnięciu ikony “oka” widzi prawie to samo co niezalogowany użytkownik z drobną różnicą. W tabeli rezerwacji wyświetlają mu się dodatkowe kolumny z danymi wynajmujących i kolumna akcji umożliwiająca usunięcie rezerwacji

Sala Konferencyjna





ID: 1

Nazwa: Sala Konferencyjna

Pojemność: 201

Udogodnienia: WiFi




Rezerwacje:

ID	Imię	Nazwisko	Email	Data rozpoczęcia	Data zakończenia	Akcje
4	akfjsdfjk	jscanjc	aknsld@vlsdkc	12/13/24, 8:00 AM	12/13/24, 10:00 AM	
2	avxczdsd	wedacxz	dasdv@dasf	12/13/24, 12:11 PM	12/13/24, 2:55 PM	
1	fasd	asddvcxz	asda@fsdfsd	12/14/24, 12:22 PM	12/14/24, 2:44 PM	
7	Kacper	P	przyklad@przyklad.pl	12/19/24, 3:00 PM	12/26/24, 10:00 AM	

[Dodaj Rezerwacje](#)

Close

aby usunąć rezerwację, tak jak w przypadku udogodnień i sal należy wcisnąć ikonę “kosza”

ID	Imię	Nazwisko	Email	Data rozpoczęcia	Data zakończenia	Akcje
4	akfjsdfjk	jscanjc	aknsld@vlsdkc	12/13/24, 8:00 AM	12/13/24, 10:00 AM	
1	fasd	asddvcxz	asda@fsdfsd	12/14/24, 12:22 PM	12/14/24, 2:44 PM	
7	Kacper	P	przyklad@przyklad.pl	12/19/24, 3:00 PM	12/26/24, 10:00 AM	

(rezerwacja została usunięta)