



WYDZIAŁ
ELEKTROTECHNIKI
I INFORMATYKI
POLITECHNIKI RZESZOWSKIEJ



Spółeczeństwo informacyjne 2

Aplikacja do zarządzania firmą

Wykonał:

Rafał Piszko 156320

12.06.2021r.

L01, 1 EF-DU

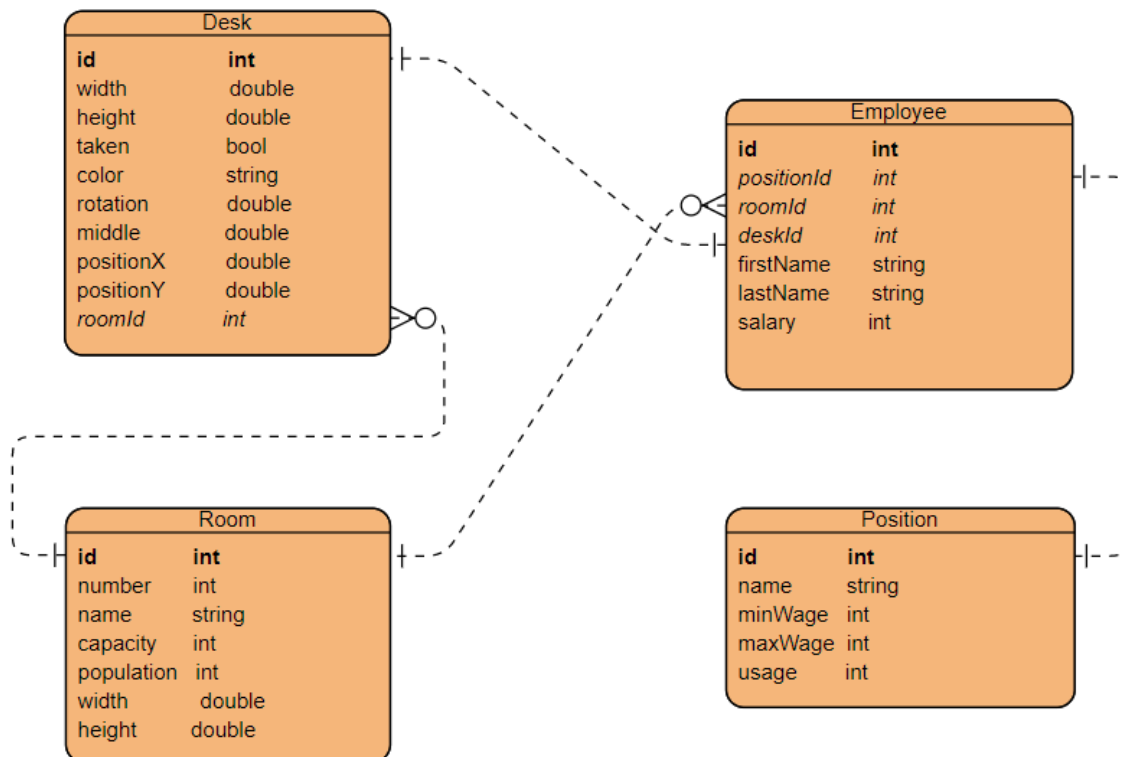
1. Technologie

Do stworzenia aplikacji wykorzystano następujące technologie:

- **Angular**, czyli Framework front-end'owy skupiony na dynamicznym aktualizowaniu zawartości strony. Wprowadza on wiele użytecznych funkcji jak wiązanie danych, co sprawia że widok jest automatycznie aktualizowany razem ze zmianą wartości zmiennej. Dodatkowym udogodnieniem jest wstrzykiwanie zależności, dzięki czemu komponenty aplikacji są ze sobą połączone. Angular reprezentuje nie tylko narzędzie, ale również wzorce projektowe, które pozwalają na utrzymywanie spójnego i przejrzystego kodu. Aplikacje napisane z wykorzystaniem tego środowiska mogą być kodowane z wykorzystaniem wielu języków i składni: ECMAScript 5, Dart, ECMAScript 6, TypeScript lub ECMAScript 7. W projekcie został wykorzystany TypeScript, który jest nadzbiorem ECMAScript 6, gdzie ostatecznie kod jest kompilowany do zwykłego JavaScript;
- **Spring Boot**, to aktualnie najpopularniejsze narzędzie do tworzenia aplikacji webowych w języku Java. Wspiera takie koncepty jak wstrzykiwanie zależności, czy programowanie aspektowe (AOP, *aspect oriented programming*). W tym narzędziu obiekty nazywane są fasolkami (*beans*) i są zarządzane oraz konfigurowane przez *BeanFactory*. Z racji swojej popularności, oprogramowanie ma szeroką społeczność, a w Internecie można znaleźć bardzo dużo dokumentacji. Spring Boot jest tylko jednym z projektów Spring'a, wśród nich można wymienić Spring Framework, Spring Data, czy Spring Security, który również jest wykorzystywany w pracy;

- **MySQL**, czyli darmowe oprogramowanie do zarządzania relacyjną bazą danych, które jest ciągle aktualizowane w celu dodania nowych funkcji i poprawy bezpieczeństwa. Istnieją płatne edycje dedykowane do użytku komercyjnego, ale przewagą wersji otwartoźródłowej jest skupienie się na szybkości i niezawodności działania. Ten program umożliwia wybór spośród wielu silników przechowywania danych, co pozwala na zmianę funkcjonalności narzędzia oraz obsługiwanie danych z tabel o różnych typach. Ważnym aspektem tego oprogramowania jest to, że oddziela przetwarzanie zapytań oraz inne zadania serwera od przechowywania i pobierania danych.

2. Diagram encji



Baza danych jest dosyć prosta. Przechowuje takie dane jak pracowników, pokoje, biurka w pokojach oraz stanowiska pracy. Do prawidłowego działania systemu baza zapisuje informacje o tym w których pokojach są jakie biurka, gdzie dokładnie są te biurka oraz kto siedzi przy tym biurku. Wszystko jest odpowiednio prezentowane na aplikacji klienta.

3. Punkty końcowe back-endu

Aplikacja napisana w Spring Boot'cie posiada następujące kontrolery:

desk-controller	Desk Controller
employee-controller	Employee Controller
position-controller	Position Controller
room-controller	Room Controller

- desk-controller, udostępnia endpointy obsługujące biurka m.in. do ich pobierania i usuwania:

desk-controller	Desk Controller
GET	/desk getDesks
PUT	/desk putDesk
GET	/desk/{id} getDeskById
PUT	/desk/Ddesks deleteDesks
PUT	/desk/desks putDesks
GET	/desk/room/{id} getDesksByRoomId

- employee-controller, udostępnia endpointy obsługujące pracowników, pozwala na ich pobieranie, usuwanie, aktualizowanie oraz dodawanie

employee-controller Employee Controller		
GET	/employee	getEmployees
POST	/employee	postEmployee
PUT	/employee	updateEmployee
GET	/employee/{id}	getEmployees
DELETE	/employee/{id}	deleteEmployeeById
GET	/employee/minMax	getMinMaxSalary
GET	/employee/room/{id}	getEmployeesByRoomId

- position-controller, udostępnia endpointy obsługujące stanowiska pracy. To podstawowy kontroler do dodawania i pobierania stanowisk:

position-controller Position Controller		
GET	/position	getPositions
POST	/position	postPosition
GET	/position/decrement/{id}	decrementRoom
GET	/position/increment/{id}	incrementRoom

- room-controller, udostępnia endpointy obsługujące pokoje. Pozwala na pobieranie i dodawanie pokoi. Podczas dodawania, dodaje się również biurka, żeby je od razu powiązać z tym pokojem.

room-controller Room Controller		
GET	/room	getRooms
POST	/room	postRoom
GET	/room/{id}	getRoom
GET	/room/decrement/{id}	decrementRoom
GET	/room/increment/{id}	incrementRoom

4. Kod

Najważniejszym elementem całej aplikacji jest kod odpowiedzialny za obsługę tworzenia nowego pokoju.

```
<svg width="500" height="500" [attr.viewBox]="viewBoxTxt" (mousemove)="mouseMoveEvent($event)"
(mouseup)="mouseUpEvent($event)" (mousedown)="mouseDownEvent($event)"
(mouseleave)="mouseUpEvent($event)">

  <g *ngFor="let desk of desks">

    <rect [attr.x]="desk.positionX" [attr.y]="desk.positionY"
      [attr.width]="desk.width[0] > desk.width[1]? desk.width[0]:desk.width[1]"
      [attr.height]="desk.width[0] > desk.width[1]? desk.width[0]:desk.width[1]" opacity="0.2"
      fill="blue"></rect>

    <g *ngIf="desk.rotation == 0">

      <ellipse
        [attr.cx]="desk.positionX + (desk.width[0] > desk.width[1]? desk.width[0]/2:desk.width[1]/2)"
        [attr.cy]="desk.positionY + (desk.positionX + desk.width[0] > desk.width[1]? desk.width[0]/2:desk.width[1]/2)"
        rx="15" ry="10" fill="white"></ellipse>

      <rect class="desk" rx="15" [attr.id]="desk.id" [attr.x]="desk.positionX"
        [attr.y]="desk.positionY" [attr.width]="desk.width[desk.rotation % 2]"
        [attr.height]="desk.height[desk.rotation % 2]" [attr.fill]="desk.color">
      </rect>
    </g>
  </g>
```

Do wyrysowania pokoju razem z biurkami zastosowano znacznik „svg”, który tworzy pole do rysowania o rozmiarach 500x500 pikseli. Jeżeli w obrębie tego elementu zostanie naciśnięty przycisk myszy to wykona się sprawdzenie czy użytkownik kliknął na myszkę.

```
170 mouseDownEvent(event) {
171   const CTM = event.target.getScreenCTM();
172   if (!this.dragging) {
173     this.findDesk(event.target.id);
174     console.log(this.selectedDesk);
175     if (this.selectedDesk > -1) {
176       this.dragging = true;
177       // tslint:disable-next-line: max-line-length
178       this.dividerX = this.desks[this.selectedDesk].width[this.desks[this.selectedDesk].rotation % 2]
179       / ((event.clientX - CTM.e) / CTM.d - this.desks[this.selectedDesk].positionX);
180       // tslint:disable-next-line: max-line-length
181       this.dividerY = this.desks[this.selectedDesk].height[this.desks[this.selectedDesk].rotation % 2]
182       / ((event.clientY - CTM.f) / CTM.a - this.desks[this.selectedDesk].positionY);
183     }
184   }
185 }
```

Polega to na sprawdzeniu czy istnieje biurko o id równym temu, które zostało kliknięte. Następnie podczas poruszania biurkiem wykonywana jest metoda „mouseMoveEvent”, której zadaniem jest przemieszczanie biurka i pilnowanie czy nie wychodzi poza krawędź pokoju.


```

98 mouseMoveEvent(event) {
99     if (this.dragging) {
100         const CTM = event.target.getScreenCTM();
101         const mouseX = (event.clientX - CTM.e) / CTM.d;
102         const fixX = this.desks[this.selectedDesk].width[this.desks[this.selectedDesk].rotation % 2] / this.dividerX;
103         const moveToX = mouseX - fixX;
104         const mouseY = (event.clientY - CTM.f) / CTM.a;
105         const fixY = this.desks[this.selectedDesk].height[this.desks[this.selectedDesk].rotation % 2] / this.dividerY;
106         const moveToY = mouseY - fixY;
107
108         let width = this.desks[this.selectedDesk].width[0];
109         if (width < this.desks[this.selectedDesk].width[1]) {
110             width = this.desks[this.selectedDesk].width[1];
111         }
112         const height = width;
113         // poruszanie biurkiem po osi X oraz sprawdzenie czy nie wychodzi poza obszar
114         if ((this.desks[this.selectedDesk].positionX - 5 > 0 || this.desks[this.selectedDesk].positionX < moveToX)
115             && (this.desks[this.selectedDesk].positionX + width + 5 < this.room.width
116                 * this.meterToPixel || this.desks[this.selectedDesk].positionX > moveToX)
117         ) {
118             this.desks[this.selectedDesk].positionX = moveToX;
119         }
120         // poruszanie biurkiem po osi Y oraz sprawdzenie czy nie wychodzi poza obszar
121         if ((this.desks[this.selectedDesk].positionY - 5 > 0 || this.desks[this.selectedDesk].positionY < moveToY)
122             && (this.desks[this.selectedDesk].positionY + height + 5 < this.room.height *
123                 this.meterToPixel || this.desks[this.selectedDesk].positionY > moveToY)
124         ) {
125             this.desks[this.selectedDesk].positionY = moveToY;
126         }
127         this.changeColor(event);
128     }
129 }

```

Poruszanie wymaga sprawdzenia czy biurko nie nachodzi na inne. Dlatego pod koniec funkcji wywołuje się funkcję „changeColor”, której implementacja wygląda następująco.

```

131 ✓ changeColor(event) {
132     let width = this.desks[this.selectedDesk].width[0];
133     if (width < this.desks[this.selectedDesk].width[1]) {
134         width = this.desks[this.selectedDesk].width[1];
135     }
136     tmp:
137     for (let j = 0; j < this.desks.length; j++) {
138         for (let i = 0; i < this.desks.length; i++) {
139             if (i !== j && Math.abs(this.desks[j].positionX - this.desks[i].positionX)
140                 <= width
141                 && Math.abs(this.desks[j].positionY - this.desks[i].positionY)
142                 <= width) {
143                 this.desks[j].color = 'red';
144                 continue tmp;
145             }
146         }
147         this.desks[j].color = '#ad7d1c';
148     }

```

Jeśli biurko najeżdża na inne to zmienia się jego kolor na czerwony, jeśli nie to na standardowy.

Kolejną metodą jest „mouseUpEvent”, która jest wykonywana przy zwolnieniu przycisku myszy, lub gdy opuści ona obszar „svg”. Funkcja sprowadza się do upuszczenia biurka, czyli zmiany wartości zmiennych, która za to odpowiada.

```
181  ✓ mouseUpEvent(event) {  
182  ✓    if (this.dragging) {  
183      this.dragging = false;  
184      this.changeColor(event);  
185      this.dividerX = -1;  
186      this.dividerY = -1;  
187      const width = this.desks[this.selectedDesk].  
188          width[this.desks[this.selectedDesk].rotation % 2];  
189      const height = this.desks[this.selectedDesk].  
190          height[this.desks[this.selectedDesk].rotation % 2];
```

Następnie sprawdza się czy biurko nie wyszło poza „svg”.

```
191      if (this.desks[this.selectedDesk].positionX - 5 < 0) {  
192          this.desks[this.selectedDesk].positionX = 5;  
193      }  
194      if (this.desks[this.selectedDesk].positionY - 5 < 0) {  
195          this.desks[this.selectedDesk].positionY = 5;  
196      }  
197      if (this.desks[this.selectedDesk].positionX + width + 5 >  
198          this.room.width * this.meterToPixel) {  
199          this.desks[this.selectedDesk].positionX =  
200              this.room.width * this.meterToPixel - width - 5;  
201      }  
202      if (this.desks[this.selectedDesk].positionY + height  
203          + 5 > this.room.height * this.meterToPixel) {  
204          this.desks[this.selectedDesk].positionY =  
205              this.room.height * this.meterToPixel - height - 5;  
206      }
```

Dodatkowo sprawdza się czy kolor biurka jest czerwony, jeśli tak to blokowany jest przycisk do przejścia dalej.

```
207     this.desks[this.selectedDesk].middle[0] = this.desks[this.selectedDesk].positionX
208     + this.desks[this.selectedDesk].width[this.desks[this.selectedDesk].rotation % 2] / 2;
209     this.desks[this.selectedDesk].middle[1] = this.desks[this.selectedDesk].positionY
210     + this.desks[this.selectedDesk].height[this.desks[this.selectedDesk].rotation % 2] / 2;
211   }
212   for (let i = 0; i < this.desks.length; i++) {
213     if (this.desks[i].color === 'red') {
214       this.valid = false;
215       return;
216     }
217   }
218   this.valid = true;
219 }
```

Bardzo ważne w całym projekcie są serwisy, których zadaniem jest komunikacja z aplikacją backend'ową, czyli pobieranie i przysyłanie danych.

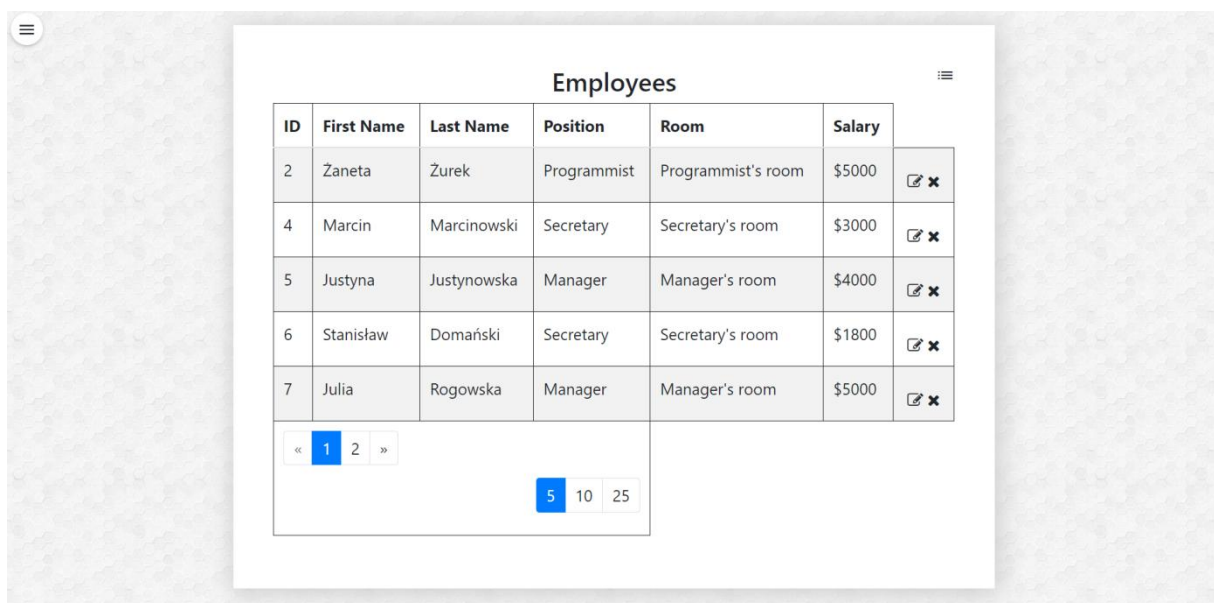
```
TS desk.service.spec.ts
TS desk.service.ts
TS employee.service.spec.ts
TS employee.service.ts
TS position.service.spec.ts
TS position.service.ts
TS room.service.spec.ts
TS room.service.ts
TS summary.service.spec.ts
TS summary.service.ts
TS table.service.spec.ts
TS table.service.ts
```

Serwisy posiadają stałą „API_URL”, której wartością jest adres backend'u. Metody korzystające z tej stałej zwyczajnie dopisują resztę wymaganego adresu, jeśli jest to wymagane.

```
7   const API_URL = "http://localhost:8080/desk";
8   ✓ const httpOptions = {
9     headers: new HttpHeaders({'Content-Type': 'application/json'})
10  };
11
12  ✓ @Injectable({
13    providedIn: 'root'
14  })
15  ✓ export class DeskService {
16
17  ✓   getSpecifiedDesks(roomId: number): Observable<any> {
18    |   return this.http.get(API_URL + '/room/' + roomId, { responseType: 'text' });
19    |   }
20
21    constructor(private http: HttpClient) { }
22
23  ✓   postDesks(desks: Desk[]): Observable<any>{
24    |   return this.http.post(API_URL, desks, httpOptions);
25    |   }
26
27  ✓   getDesks(): Observable<any> {
28    |   return this.http.get(API_URL, { responseType: 'text' });
29    |   }
30
31  ✓   deleteDesks(desks: Desk[]): Observable<any> {
32    |   return this.http.put(API_URL+"/Ddesks", desks, { responseType: 'text' });
```

5. Prezentacja aplikacji

Strona Główna aplikacji prezentuje tabelę z zatrudnionymi pracownikami. Tabela zawiera ich ogólne informacje (imię, nazwisko, stanowisko pracy, pokój, zarobki). W ostatniej kolumnie znajdują się przyciski do edycji i usunięcia pracownika. Tabela posiada ograniczoną liczbę pracowników na jedną stronę, gdzie standardowo wynosi to 5, a można zmienić na 10 lub 25.



ID	First Name	Last Name	Position	Room	Salary	
2	Zaneta	Żurek	Programmist	Programmist's room	\$5000	
4	Marcin	Marcinowski	Secretary	Secretary's room	\$3000	
5	Justyna	Justynowska	Manager	Manager's room	\$4000	
6	Stanisław	Domański	Secretary	Secretary's room	\$1800	
7	Julia	Rogowska	Manager	Manager's room	\$5000	

« 1 2 »5 10 25

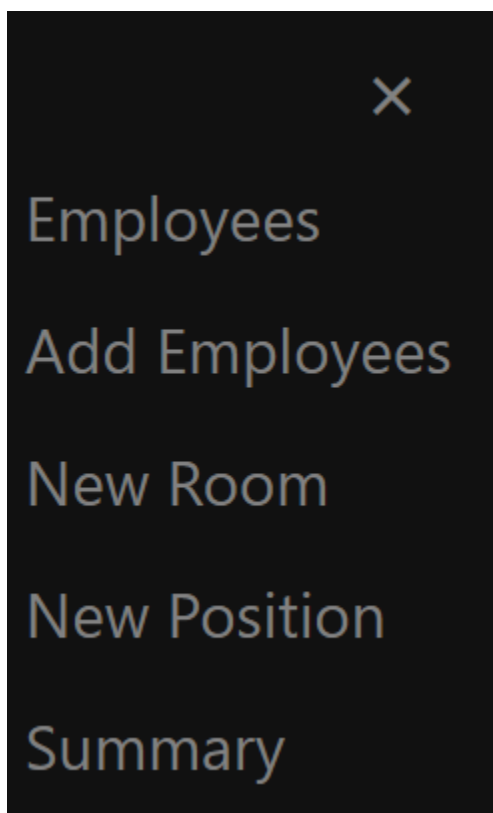
W prawym górnym rogu tabeli jest przycisk, który pozwala na rozwinięcie menu filtracji pracowników.

Employees

first name last name --select position-- ▾

Minimal salary: 1800 Maximal salary: 5000

W lewym górnym rogu strony znajduje się przycisk do rozwijania menu nawigującego po podstronach.



Istnieje możliwość wyboru pięciu podstron. Pierwsza z nich to omówiona już tabela pracowników. Druga pozwala na dodawanie nowego pracownika. Trzecia dodaje nowy pokój. Czwarta dodaje nowe stanowisko pracy, a ostatnia wyświetla podsumowanie całego systemu.

- **Add Employees**

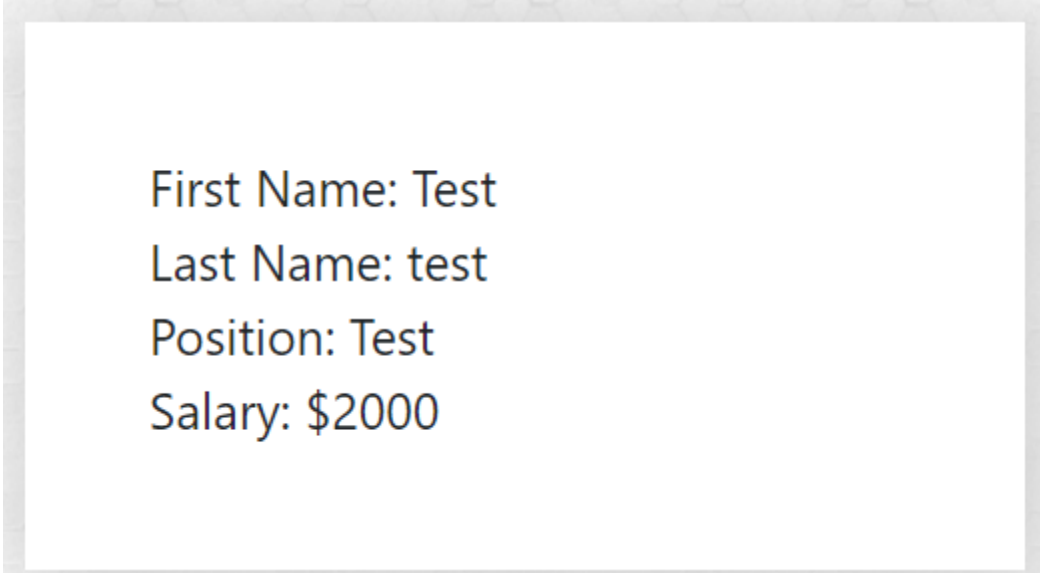
Jak już wcześniej wspomniano, ta podstrona pozwala na dodanie pracownika. Aby to zrobić trzeba przejść trzy kroki, gdzie w pierwszym podaje się jego ogólne dane:

The screenshot shows a three-step process for adding a new employee. Step 1 is active, indicated by a purple circle with a white checkmark. The form is titled 'Add a new employee' and contains five input fields: 'first name', 'last name', '--Select a position--', '--Select a room--', and 'salary'. The 'Next' button is located at the bottom of the form.

W drugim wybiera się miejsce w pokoju w którym będzie siedział:

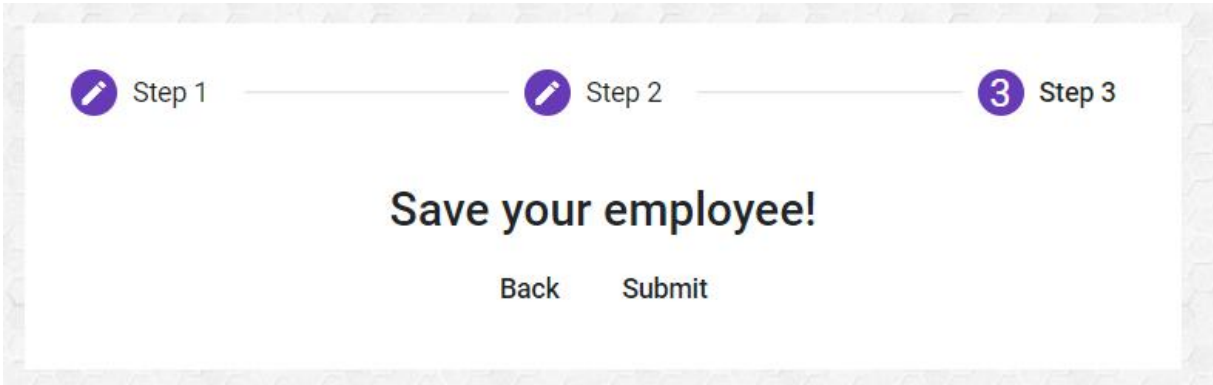
The screenshot shows the second step of the process, where the user selects a room. The interface displays a grid of four room icons, each consisting of a purple square with a white circle and a colored rectangle (yellow, blue, brown, and brown). The 'Back' and 'Next' buttons are located at the bottom of the screen.

Niebieskie biurko to wybrane. Żółte to zajęte a brązowe to miejsca wolne. Kliknięcie na zajęte biurko pozwala na wyświetlenie komunikatu kto to miejsce zajmuje:

A white rectangular box with a thin grey border. Inside, the following text is displayed in a sans-serif font:

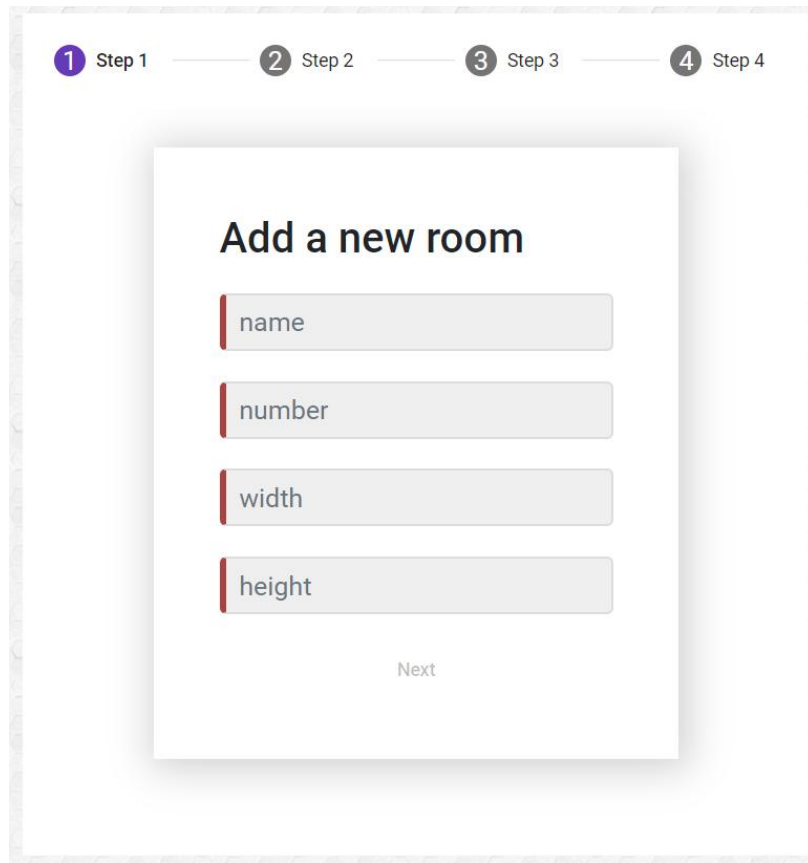
First Name: Test
Last Name: test
Position: Test
Salary: \$2000

Po wybraniu miejsca ostatnim krokiem jest zapisanie pracownika:

A white rectangular box with a thin grey border. At the top, there is a progress bar with three steps: 'Step 1' (pencil icon), 'Step 2' (pencil icon), and 'Step 3' (number 3 icon). The text 'Save your employee!' is centered below the progress bar. At the bottom, there are two buttons: 'Back' and 'Submit'.

- **New Room**

Ta podstrona pozwala stworzenie nowego pokoju po przejściu czterech kroków. Pierwszym krokiem jest nadaniu pokoju nazwy, numeru oraz rozmiarów w metrach:



The screenshot shows a web interface for adding a new room. At the top, there is a progress bar with four steps: '1 Step 1' (active), '2 Step 2', '3 Step 3', and '4 Step 4'. Below the progress bar is a white card with the title 'Add a new room'. Inside the card, there are four text input fields, each with a red vertical bar on the left and a placeholder label: 'name', 'number', 'width', and 'height'. At the bottom of the card is a 'Next' button.

Po wypełnieniu tych danych można przejść do następnego kroku. Aplikacja oblicza ile w danym pokoju o danych rozmiarach może się znaleźć użytkowników. Dla prezentacji stworzono pokój o rozmiarach 6m x 8m.

Aplikacja obliczyła, że w tym pokoju zmieści się od 0 do 6 biurek z zachowaniem wymaganych odległości dla pracowników. Dla celów prezentacji wybrano 6 biurek.

Step 1 — Step 2 — Step 3 — Step 4

Fill capacity in the range of 0 - 6

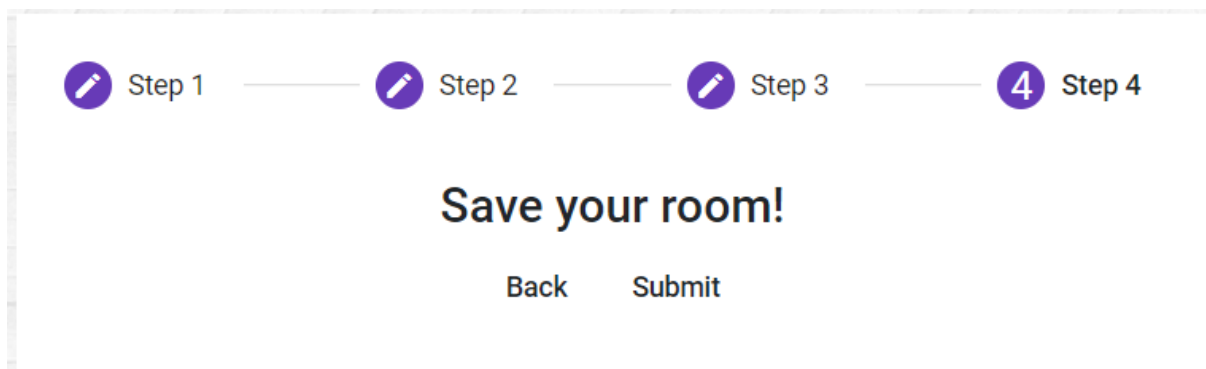
Room's capacity is required.

Back Next

Po przejściu do następnego kroku następuje wyrysowanie pokoju i wygenerowanie biurek. Biurka należy rozłożyć w dowolny sposób tak, aby na siebie nie nachodziły. Poniżej przedstawiono niepoprawne oraz poprawne rozłożenie:



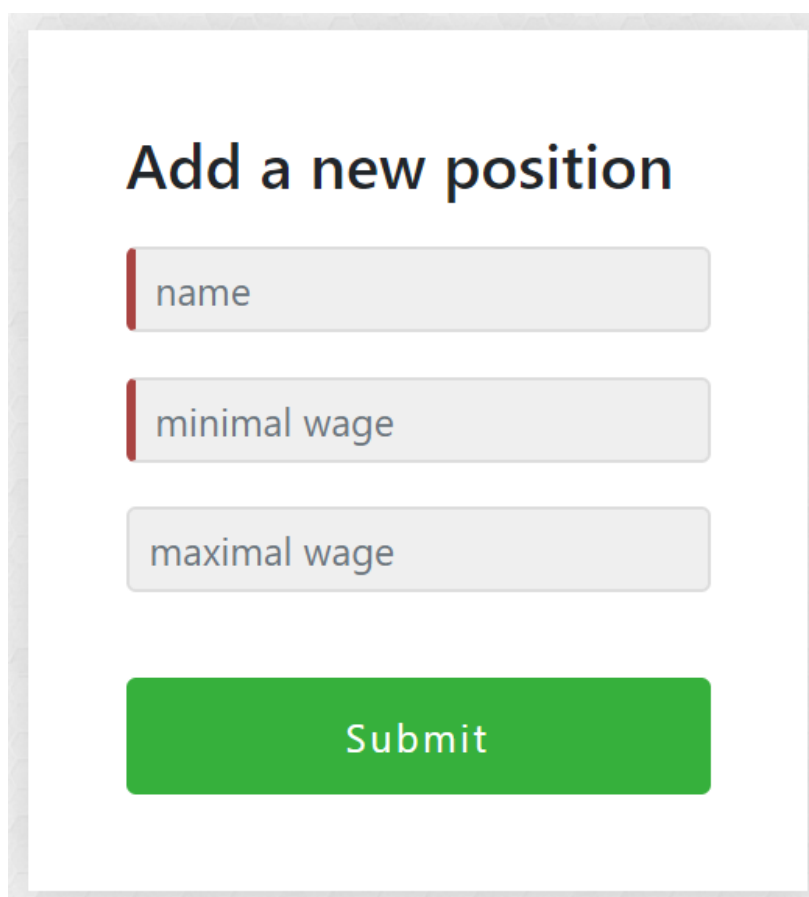
Po przejściu dalej należy zapisać stworzony pokój:



The screenshot shows a progress bar at the top with four steps: Step 1, Step 2, Step 3, and Step 4. Step 4 is highlighted with a purple circle and the number 4. Below the progress bar, the text "Save your room!" is centered. Underneath, there are two buttons: "Back" and "Submit".

- **New Position**

Ta podstrona jest bardzo prosta, polega na wpisaniu nazwy stanowiska pracy oraz jego minimalnej i maksymalnej płacy:



The screenshot shows a form titled "Add a new position". It contains three input fields: "name", "minimal wage", and "maximal wage". Each input field has a red vertical bar on its left side. Below the input fields is a green button labeled "Submit".

- **Summary**

Ta podstrona to jedynie podsumowanie systemu. Pozwala zobaczyć ile jest pokoi oraz ile jest w nich pracowników, ile jest stanowisk pracy oraz ile jest zatrudnionych na nich pracowników i inne ogólne dane:

Summary

Rooms

- Manager's room (3/3) - 1
- Programmer's room (1/5) - 2
- Secretary's room (2/5) - 5
- Test (2/4) - 101
- room123 (0/6) - 123

Positions

- Manager (hired: 4)
- Programmer (hired: 1)
- Secretary (hired: 2)
- Test (hired: 1)

General

- Employees: 8
- Rooms: 5
- Positions: 4
- Total spendings: \$24800

6. Podsumowanie

Cała aplikacja działa poprawnie i spełnia swoje zadanie. Różne opcje wyboru są zabezpieczone przez odpowiednią walidację, tak więc nie można stworzyć pokoju dla 7 osób jeśli system obliczył, że maksymalnie może być 6. Jedynym problemem może być to, że można stworzyć dwa pokoje z tym samym numerem lub tą samą nazwą oraz dwa stanowiska pracy o tej samej nazwie. Jednak jest to czynność celowa, ponieważ pozostawia to dla osoby zarządzającej dowolność działania. Projekt może się wydawać dość prosty, jednak najwięcej trudności sprawiło stworzenie mechanizmu drag'n'drop do obsługiwanie rozmieszczania biurek.