

Projekt Bazy Danych

Bazy Danych 2021/22

Szymon Słota, Sebastian Kozak, Jakub Łubkowski

1. Temat projektu

Aplikacja webowa, umożliwiająca znalezienie toalety blisko ciebie.

2. Technologie

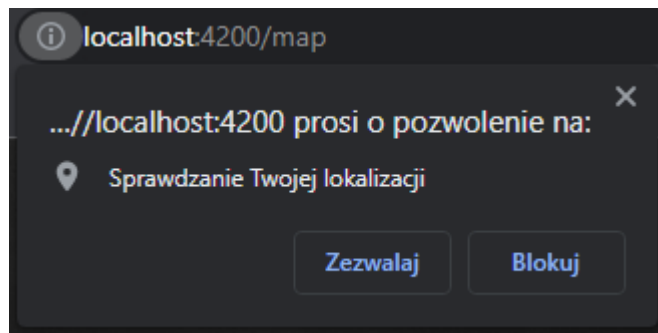
Baza danych: Atlas (MongoDB)

Backend: Spring (Java)

Frontend: Angular (TypeScript)

3. Funkcjonalności

A. Prośba o pozwolenie na sprawdzanie twojej lokalizacji, w celu konfiguracji mapy.



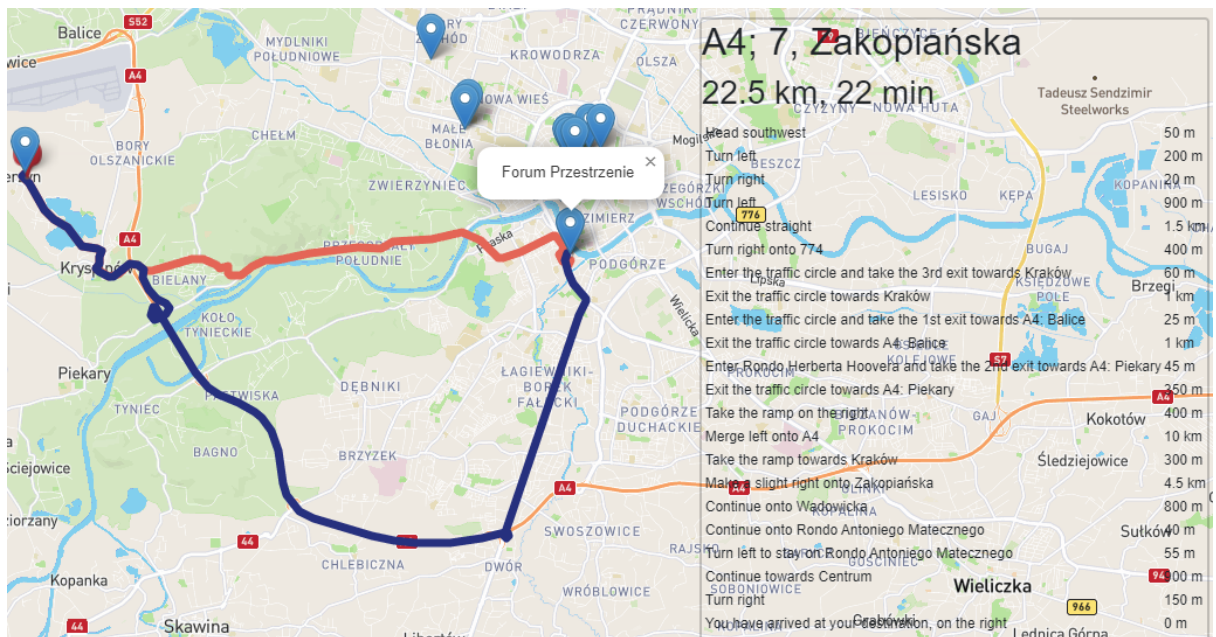
Zrzut ekranu 1. Prośba o zezwolenie na sprawdzanie lokalizacji.

B. Mapa wyświetlająca twoją lokalizację oraz toalety blisko ciebie.



Zrzut ekranu 2. Mapa z lokalizacjami.

Mapa zawiera dwa rodzaje markerów: niebieski oraz czerwony, niebieski oznacza toalety natomiast czerwony to lokalizacja urządzenia z, którego korzystamy. Naciśnięcie na niebieski marker automatycznie wyznaczy nam drogę wraz ze szczegółami, poda odległość oraz oszacowany czas dojazdu.



Zrzut ekranu 3. Mapa z lokalizacjami.

Niebieska trasa to czasowo najkrótsza natomiast pomarańczowa to alternatywna opcja.

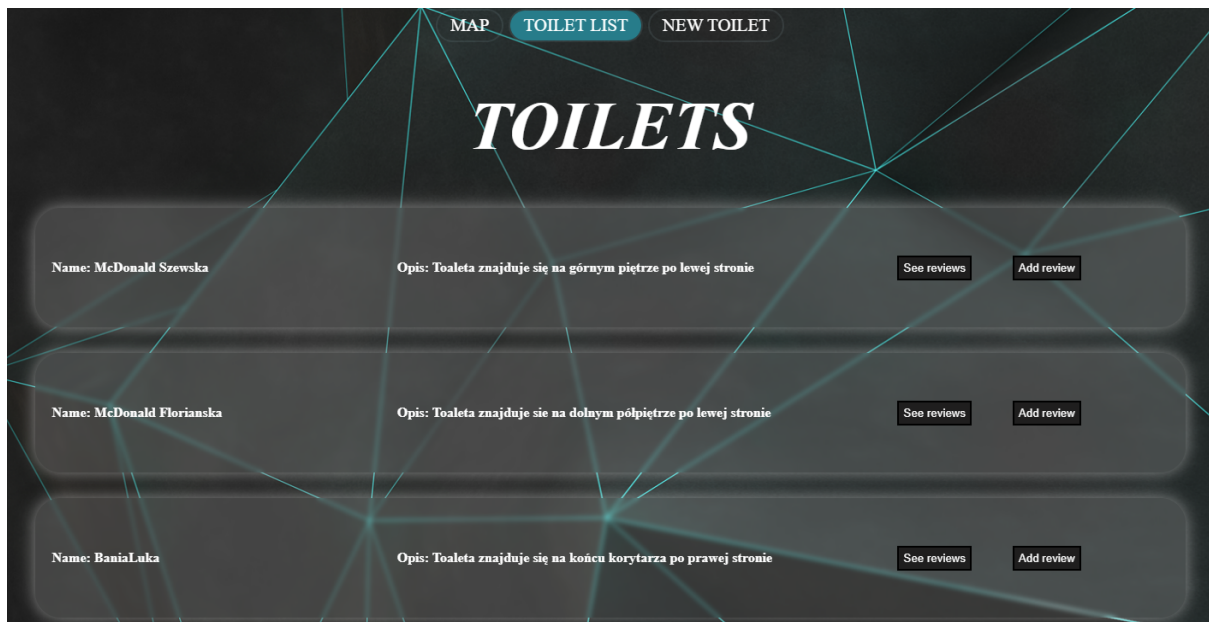
C. Lista toalet znajdująca się pod mapą.

Name	Info	Paid	Track
McDonald Szewska	Toaleta znajduje się na górnym piętrze po lewej stronie	false	
McDonald Florianska	Toaleta znajduje się na dolnym półpiętrze po lewej stronie	false	
BaniaLuka	Toaleta znajduje się na końcu korytarza po prawej stronie	false	
Toaleta publiczna	Toaleta znajduje się na rogu sukienic od strony północnej	true	
WC Teatr Słowackiego	Toaleta znajduje się w przejściu podziemnym, pod skrzyżowaniem ulicy Lubicz oraz Westerplatte	true	

Zrzut ekranu 4. Lista toalet znajdująca się pod mapą.

Lista zawiera nazwy toalet, opisy, informację o opłatach (false oznacza, iż toaleta jest darmowa, natomiast true świadczy o toalecie płatnej) oraz tracker, który pozwala z punktu listy nawigować do wybranej przez siebie toalety.

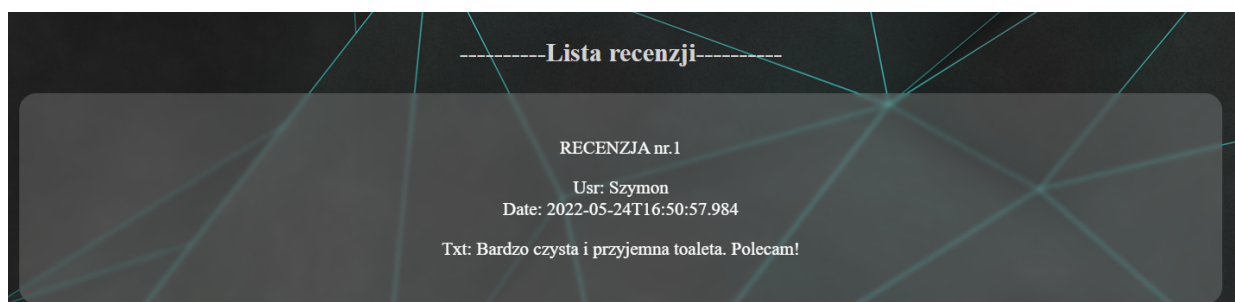
D. Zakładka lista toalet.



Zrzut ekranu 5. Mapa z lokalizacjami.

Zakładka służy do wylistowania wszystkich toalet oraz umożliwia nam dodanie recenzji do poszczególnej toalety bądź przeglądnięcia istniejących już opinii.

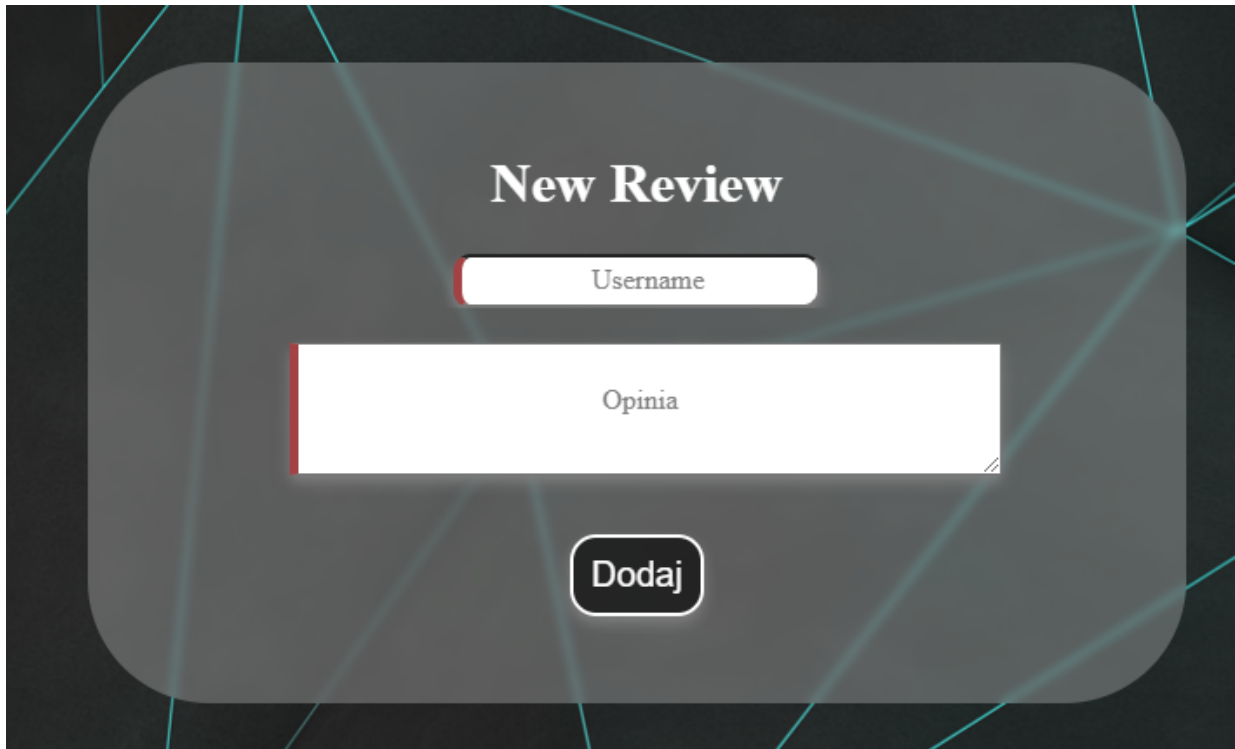
E. See reviews



Zrzut ekranu 6. Lista opinii.

Lista recenzji ukazuje nam wszystkie opinie, które zamieścili użytkownicy do danej toalety.

F. Add review

A screenshot of a web form titled "New Review". The form is centered on a dark background with a light gray rounded rectangle. It contains three main elements: a text input field labeled "Username", a larger text area labeled "Opinia", and a button labeled "Dodaj". The "Username" field has a red vertical bar on its left side. The "Opinia" field has a red vertical bar on its left side and a small diagonal line in its bottom right corner. The "Dodaj" button is a rounded rectangle with a white border and a dark background.

Zrzut ekranu 7. Dodawanie opinii.

Dodawanie recenzji do poszczególnej toalety, wymagane pola:
nazwa użytkownika oraz tekst opinii.

G. New toilet

MAP TOILET LIST NEW TOILET

New TOILET

Name

Street

City

0

0

Description

Paid: ☒

Dodaj

Zrzut ekranu 8. Dodawanie toalety.

Dodawanie toalety przez użytkownika, wymagane pola: nazwa, ulica, miasto, szerokość i długość geograficzna, krótki opis toalety, który zawiera np. dokładne umiejscowienie toalety w budynku oraz pole w, którym wybieramy czy dana lokalizacja jest płatna bądź darmowa (domyślnie jest płatna).

Poprawne wypełnienie powyższego formularza warunkujemy patternami w pliku add-toilet.component.html, przykładowo pattern dla długości geograficznej:

```
<div class="form-group">
  <input type="text" class="form-control" id="lng" placeholder=" "
    required [(ngModel)]="model.coordinates.lng"
    name="lng" #lng="ngModel"
    pattern="(
      [-][1-9][0-7][0,9][.][0-9]*|
      [-][1-9][0-9][.][0-9]*|
      [-][1-9][.][0-9]*|
      [1-9][0-7][0,9][.][0-9]*|
      [1-9][0-9][.][0-9]*|
      [1-9][.][0-9]*)">
  <label for="lng">Lng</label>
</div>
```

Zrzut ekranu 9. Pattern długości geograficznej.

Leaflet

Konfiguracja w angularze

npm i leaflet

npm i --save-dev @types/leaflet

npm i leaflet-routing-machine

npm i --save-dev @types/leaflet-routing-machine

Tworzenie mapy z markerem gdzie się znajdujemy

```
navigator.geolocation.getCurrentPosition((position) => {  
  MapComponent.gps = [position.coords.latitude, position.coords.longitude];  
  MapComponent.map = L.map('map').setView(MapComponent.gps, 13);  
  MapComponent.map.on('click', (i:LeafletMouseEvent)=>{  
    if(MapComponent.route!=null) MapComponent.map.removeControl(MapComponent.route);  
  });  
});
```

```
let marker = L.marker(MapComponent.gps,{icon: redIcon}).addTo(MapComponent.map);  
marker.bindPopup('<b>You!</b>').openPopup();
```

Dodawanie markerów naszych toalet do mapy

```
this.toiletList.forEach(t => {  
  let marke2r = L.marker([t.coordinates.lat, t.coordinates.lng]).on('click', function(e:any){  
    if(MapComponent.route!=null) {MapComponent.map.removeControl(MapComponent.route);}  
    MapComponent.route = L.Routing.control({  
      router: L.Routing.osrmv1({  
        serviceUrl: `http://router.project-osrm.org/route/v1/`  
      })),  
      showAlternatives: true,  
      lineOptions: {styles: [{color: '#242c81', weight: 7}]},  
      fitSelectedRoutes: false,  
      altLineOptions: {styles: [{color: '#ed6852', weight: 7}]},  
      routeWhileDragging: false,  
      waypoints: [  
        L.latLng(MapComponent.gps),  
        L.latLng(e.latlng)  
      ]  
    }).addTo(MapComponent.map);  
  });  
  console.log(t.coordinates.lat, t.coordinates.lng);  
  marke2r.bindPopup(t.name).openPopup();  
  marke2r.addTo(MapComponent.map);  
});
```

Pokazanie routingu na mapie od nas do klikniętego markera

```
update(t:number[]){
  this.target=t;
  console.log(this.target);
  if(MapComponent.route!=null) {MapComponent.map.removeControl(MapComponent.route);}
  MapComponent.route = L.Routing.control({
    router: L.Routing.osrmv1({
      serviceUrl: `http://router.project-osrm.org/route/v1/`
    }),
    showAlternatives: true,
    lineOptions: {styles: [{color: '#242c81', weight: 7}]},
    fitSelectedRoutes: false,
    altLineOptions: {styles: [{color: '#ed6852', weight: 7}]},
    routeWhileDragging: false,
    waypoints: [
      L.latLng(MapComponent.gps),
      L.latLng(this.target)
    ]
  }).addTo(MapComponent.map);
}
```

Funkcja obserwująca mapę i możliwe obsługująca błędy

```
watchPosition() {
  let aimLat = 0;
  let aimLng = 0;
  let id = navigator.geolocation.watchPosition((position) => {
    console.log(
      'lat: ${position.coords.latitude}, lng: ${position.coords.longitude}'
    );
    if(position.coords.latitude === aimLat) {
      navigator.geolocation.clearWatch(id);
    }
    if(position.coords.longitude === aimLng) {
      navigator.geolocation.clearWatch(id); // it stops requiring gps requests
    }
  },(err) => {
    console.log(err);
    window.location.reload();
  },{
    enableHighAccuracy: true,
    timeout: 5000,
    maximumAge: 0
  });
}
```

Backend

Struktura obiektu toilet w naszej bazie w Mongo

```
import { Address } from "../address";
import { Coordinate } from "../coördiante";
import { Review } from "../review";

export interface Toilet {
  id: string,
  name: string,
  coordinates: Coordinate,
  address: Address,
  description: string,
  reviews: Review[],
  paid?: boolean
}
```

```
export interface Review {
  user: string,
  description: string,
  date: Date
}
```

```
export interface Address {
  street: string,
  city: string
}
```

```
export interface Coordinate {
  lat: number,
  lng: number
}
```

Przykładowy obiekt w bazie

```
{
  "_id": "628cdfe0420f452862e3c0c2",
  "name": "McDonald Szewska",
  "coordinates": {
    "lat": "50.062119",
    "lng": "19.935749"
  },
  "address": {
    "street": "Szewska",
    "city": "Krakow"
  },
  "description": "Toaleta znajduje się na górnym piętrze po lewej stronie",
  "reviews": [
    {
      "user": "Szymon",
      "description": "Bardzo czysta i przyjemna toaleta. Polecam!",
      "date": "2022-05-24T14:50:57.984+00:00"
    }
  ],
  "paid": false,
  "_class": "com.example.backendtoilet_searcher.domain.toilet.Toilet"
}
```

Pobieranie toalety z bazy po jej id

```
@GetMapping("/{id}")
public Toilet getToiletById(@PathVariable("id") String id){
    try {
        return this.toiletService.findToiletById(id);
    } catch (NotFoundException ex){
        throw new ResponseStatusException(
            HttpStatus.NOT_FOUND,
            "Toilet not found",
            ex
        );
    }
}
```

Pobieranie wszystkich toalet z bazy danych

```
@GetMapping()
public List<Toilet> getToilets(){
    return toiletService.findAllToilets();
}
```

Pobieranie recenzji z bazy po id toalety

```
@GetMapping("/{id}/reviews")
public List<Review> getReviews(@PathVariable("id") String id){
    try {
        return this.toiletService.getToiletReviews(id);
    } catch (NotFoundException ex){
        throw new ResponseStatusException(
            HttpStatus.NOT_FOUND,
            "Toilet not found",
            ex
        );
    }
}
```

Dodawanie nowej toalety do bazy danych

```
@PostMapping()  
public Toilet createToilet(@RequestBody ToiletRequestDTO request){  
    System.out.println("Toilet req sent!");  
    try{  
        return toiletService.insertToilet(request);  
    } catch(AlreadyExistsException exception){  
        throw new ResponseStatusException(  
            HttpStatus.BAD_REQUEST,  
            "Toilet already exist",  
            exception  
        );  
    }  
}
```

```
model = {  
    name : '',  
    coordinates : {  
        lat : 0,  
        lng : 0  
    },  
    address : {  
        street : '',  
        city : ''  
    },  
    description : '',  
    paid : true  
};
```

oraz model json który wysyłamy do bazy podczas dodawania nowej toalety do bazy danych

```
public Toilet insertToilet(ToiletRequestDTO toiletRequestDTO){  
    Toilet toilet = Toilet.builder()  
        .name(toiletRequestDTO.getName())  
        .coordinates(toiletRequestDTO.getCoordinates())  
        .address(toiletRequestDTO.getAddress())  
        .description(toiletRequestDTO.getDescription())  
        .reviews(new ArrayList<>())  
        .paid(toiletRequestDTO.getPaid())  
        .build();  
  
    Boolean exists = this.toiletRepository.existsToiletByName(toiletRequestDTO.getName());  
  
    if(exists){  
        System.out.println("toilet already exists");  
        throw new AlreadyExistsException("toilet already exists");  
    }else{  
        return this.toiletRepository.insert(toilet);  
    }  
}
```

Dodawanie nowej recenzji do bazy danych po id toalety

```
@PostMapping("/{id}/reviews")
public Future<Toilet> createReview(@PathVariable("id") String id, @RequestBody ReviewRequestDTO request){
    return this.toiletService.insertReview(id,request).toFuture();
}
```

model json który wysyłamy do bazy podczas dodawania nowej recenzji do bazy danych

```
model = {
  username : '',
  description : ''
};
```

dodatkowo już na backendzie rejestrowana jest data dodania recenzji i zostaje ona zapisana do bazy, gdzie później korzystamy z niej podczas wyświetlania wszystkich recenzji danej toalety

```
public Mono<Toilet> insertReview(String id, ReviewRequestDTO reviewRequest){
    Review review = Review.builder()
        .user(reviewRequest.getUsername())
        .date(LocalDateTime.now())
        .description(reviewRequest.getDescription())
        .build();

    return this.reactiveToiletRepository
        .findById(id)
        .map(toilet -> {
            toilet.addReviewToToilet(review);
            return toilet;
        })
        .flatMap(this.reactiveToiletRepository::save);
}
```

Usunięcie toalety z bazy danych po jej id

```
@DeleteMapping("/{id}")
public void deleteToilet(@PathVariable("id") String id){
    this.toiletService.deleteToilet(id);
}
```


Usunięcie wszystkich toalet z bazy danych

```
@DeleteMapping() 🌐  
public void deleteToilet(){  
    this.toiletService.deleteAllToilets();  
}
```