Opis

Ovaj projekat predstavlja backend aplikaciju za primanje zahteva i slanje odgovora klijentskom delu Uber aplikacije, kao i mobilnoj aplikaciji. Glavni cilj je pojednostaviti proces pronalaženja prevoza. Takođe je fokus da se smanji potreba za direktnom komunikacijom sa vozačem i obezbedi brži, pouzdaniji i sigurniji transport za korisnike.

Platforma je namenjena za četiri glavne vrste korisnika:

1. **Neregistrovani korisnici**: Oni mogu pregledati osnovne informacije o aplikaciji i uneti početnu i krajnju lokaciju kako bi dobili procenu vremena dolaska i cene vožnje.

2. **Registrovani korisnici**: Ovi korisnici mogu zatražiti vožnje, dobijati obaveštenja u realnom vremenu o statusu svoje vožnje, pratiti vozila na mapi, ocenjivati vozače i vozila nakon vožnje, pristupiti istoriji svojih vožnji i zakazivati buduće vožnje radi prioriteta tokom gužvi. Takođe, mogu definisati omiljene rute za brzi izbor i koristiti dugme PANIC tokom vožnje za hitne situacije.

3. **Vozači**: Oni automatski dobijaju vožnje od sistema sa informacijama o početnoj i krajnjoj lokaciji. Mogu ažurirati svoj profil, pregledati istoriju svojih vožnji, generisati izveštaje o vožnjama i pristupiti dugmetu PANIC u hitnim slučajevima. Vozači mogu promeniti svoj status dostupnosti ručno, a ako rade više od 8 sati dnevno, automatski će biti označeni kao nedostupni.

4. **Administratori**: Ovi korisnici imaju privilegije za upravljanje platformom. Oni mogu kreirati vozačke naloge, pregledati informacije i status vožnji svakog vozača, pristupiti istoriji vožnji i generisati izveštaje. Takođe, administratori mogu blokirati korisničke naloge, odgovarati na hitna obaveštenja i pružati podršku putem četa 24/7.

Članovi

- Filip Vuksan SV1/2020
- Andrea Katzenberger SV69/2020
- Bojana Popov SV70/2020
-

Bilo je potrebno oko sat do sat ipo vremena za pronalaženje defekata. U kodu je pronađeno više sličnih defekata. Potrudio sam se da izdvojim samo one defekte koji se razlikuju

Opis pronađenih defekata

```
@PutMapping (value = "/{id}/changePassword", consumes = MediaType.APPLICATION_JSON_VALUE, produces = MediaType.APPLICATION_JSON_VALUE)
public ResponseEntity<?> changePassword(@PathVariable("id") String id, @RequestBody RequestUserChangePasswordDTO requestUserChangePasswordDTO) {

    if(!StringUtils.isNumeric(id)){
        return new ResponseEntity<>(new MessageDTO("Id is not numeric"), HttpStatus.NOT_FOUND);
    }
    if(!userService.existsById(id)){
        return new ResponseEntity<>(new MessageDTO("User does not exist!"), HttpStatus.NOT_FOUND);
    }
    User user = userService.getUser(id).get();

    if(passwordEncoder.matches(requestUserChangePasswordDTO.getOldPassword(), user.getPassword())){
        user.setPassword(passwordEncoder.encode(requestUserChangePasswordDTO.getNewPassword()));
        userService.add(user);
        return new ResponseEntity<>(new MessageDTO("Password successfully changed!"), HttpStatus.NO_CONTENT);
    }
    return new ResponseEntity<>(new MessageDTO("Current password is not matching!"), HttpStatus.BAD_REQUEST);
}
```

1) Nedostatak autorizacije: Nije jasno definisano ko može pristupiti ovom endpointu
2) Bezbednosni propusti: Nedostatak provere indentiteta korisnika koji menja lozinku
3) Nedostatak zaštite od CSRF napada prilikom promene lozinke
4) Nedostak enkripcije: Lozinka se možda ne enkriptuje pravilno pre čuvanja u bazi podataka
5) XSS napadi: Neprovereni input zahteva

```
@GetMapping(value = "/{id}/ride", produces = MediaType.APPLICATION_JSON_VALUE)
@PreAuthorize("hasAnyAuthority('ADMIN', 'DRIVER', 'PASSENGER')")
public ResponseEntity<?> getUserRides(@PathVariable("id") String id, Pageable page) {

    if(!StringUtils.isNumeric(id)){ return new ResponseEntity<>(new MessageDTO("Id is not numeric"), HttpStatus.NOT_FOUND);}
    if(!userService.existsById(id)){return new ResponseEntity<>(new MessageDTO("User does not exist!"), HttpStatus.NOT_FOUND);}
    User user = this.userService.getUser(id).get();

    List<ResponseRideNoStatusDTO> responseRides = new ArrayList<>();

    if(user.getRole() == Role.PASSENGER){
        Page<Ride> rides = this.rideService.getRidesForPassenger(user.getId().toString(), page);
        for(Ride r: rides){
            responseRides.add(r.parseToResponseNoStatus());
        }
        return new ResponseEntity<>(new ResponsePageDTO(rides.getNumberOfElements(), Arrays.asList(responseRides.toArray())), HttpStatus.OK);
    }

    else if(user.getRole() == Role.DRIVER){
        Page<Ride> rides = this.rideService.getRidesForDriver(user.getId().toString(), page);
        for(Ride r: rides){
            responseRides.add(r.parseToResponseNoStatus());
        }
        return new ResponseEntity<>(new ResponsePageDTO(rides.getNumberOfElements(), Arrays.asList(responseRides.toArray())), HttpStatus.OK);
    }
    return new ResponseEntity<>(new MessageDTO("Cant get rides for ADMIN"), HttpStatus.NOT_FOUND);
}
```

1. Otkrivanje korisnika: Endpoint može otkriti postojanje korisnika na osnovu greške koja se vraća
2. XSS napadi: Neprovereni input zahteva
3. Napadi na paging: Nedovoljna validacija paging parametara može dovesti do različitih vrsta napada, kao što su SQL injection napadi.

```
@GetMapping(value = "/{id}/resetPassword", produces = MediaType.APPLICATION_JSON_VALUE)
@PreAuthorize("hasAnyAuthority('ADMIN', 'DRIVER', 'PASSENGER')")
public ResponseEntity<?> resetPassword(@PathVariable("id") String id) throws MessagingException, UnsupportedEncodingException {

    if(!StringUtils.isNumeric(id)){return new ResponseEntity<>(new MessageDTO("Id is not numeric"), HttpStatus.NOT_FOUND);}
    if(!userService.existsById(id)){return new ResponseEntity<>(new MessageDTO("User does not exist!"), HttpStatus.NOT_FOUND);}

    User user = userService.getUser(id).get();
    String token = String.valueOf( i: new Random().nextInt( bound: 900000) + 100000);
    user.setResetPasswordToken(token);
    user.setResetPasswordTokenExpiration(LocalDateTime.now().plusMinutes(10));

    mailService.sendMail( recipientEmail: "filipvuksan.iphone@gmail.com", token);
    userService.add(user);

    return new ResponseEntity<>( body: "Email with reset code has been sent!", HttpStatus.NO_CONTENT);
}
```

1. Nebezbedan reset lozinke: Token za resetovanje lozinke nije generisan sigurno, što može dovesti do zloupotrebe

Sažeta preporuka sa kojima se kod poboljšava

1. **Autorizacija**: Potrebno je jasno definisati ko može pristupiti svakom endpointu i primeniti odgovarajuće mehanizme autorizacije
2. **Bezbednost lozinke**: Treba osigurati siguran proces resetovanja lozinke, generisajući slučajne i jadnokratne tokene koristeći sigurne kanale komunikacije za slanje reset linkova ili kodova
3. **Validacija korisničkog unosa**: Proveriti i validirati sve ulazne podatke kako bi sprečili napade poput XSS i SQL injection.
4. **Zaštita od CSRF napada**: Implementirati mehanizme zaštite od CSRF napada prilikom izvršavanja akcija kao što je promena lozinke.
5. **Enkripcija lozinke**: Osigurati da se lozinke enkriptuju pre čuvanja u bazi podataka kako bi se zaštitili osetljivi podaci korisnika.
6. **Sigurnost paging parametara**: Validirati i ograničiti paging parametre kako bi sprečili napade poput SQL injection.
7. **Upravljanje greškama**: Paziti na upravljanje greškama i ne otkrivati suvišne informacija o korisnicima ili sistemu.