

Sveučilište u Zagrebu  
Prirodoslovno-matematički fakultet  
Matematički odsjek

Nora Berdalović, Laura Horvat i Nikola Kašnar

# Upravljanje bolničkim sustavom

Projekt iz Naprednih baza podataka

Profesor: Ognjen Orel

Zagreb, lipanj 2024.

# Sadržaj

<b>1</b>	<b>Uvod</b>	<b>2</b>
<b>2</b>	<b>Opis projekta</b>	<b>3</b>
<b>3</b>	<b>Baza podataka</b>	<b>4</b>
3.1	Relacije . . . . .	4
3.1.1	Administrator . . . . .	4
3.1.2	Bolnica . . . . .	4
3.1.3	Mjesto . . . . .	5
3.1.4	Susjedi . . . . .	5
3.1.5	Pretraga . . . . .	5
3.1.6	Pacijent . . . . .	5
3.1.7	Liječnik opće prakse . . . . .	5
3.1.8	Bolnice i odgovarajuće pretrage . . . . .	5
3.1.9	Zakazani termini pretraga . . . . .	6
3.1.10	Zahtjevi za prebacivanjem pacijenta . . . . .	6
3.1.11	Zahtjevi za pretragom . . . . .	6
3.1.12	Prijedlozi termina . . . . .	6
3.2	Indeksi . . . . .	6
3.3	Funkcije . . . . .	6
3.3.1	Susjedne bolnice . . . . .	8
3.3.2	Popis pacijenata i pretraga . . . . .	8
3.3.3	Lista čekanja i slobodni termini . . . . .	8
3.4	Ubacivanje podataka . . . . .	8
<b>4</b>	<b>Izgradnja aplikacije</b>	<b>10</b>
4.1	Spajanje na bazu . . . . .	10
4.2	Izgled i funkcionalnost aplikacije . . . . .	11
4.2.1	Liječnik . . . . .	14
4.2.2	Pacijent . . . . .	17
4.2.3	Administrator . . . . .	18
<b>5</b>	<b>Zaključak</b>	<b>20</b>
<b>6</b>	<b>Samoevaluacija</b>	<b>21</b>
6.1	Nora Berdalović . . . . .	21
6.2	Laura Horvat . . . . .	21
6.3	Nikola Kašnar . . . . .	21
<b>7</b>	<b>Literatura</b>	<b>22</b>

# 1 Uvod

Koliko se puta dogodilo, nama ili bliskim nam ljudima, da termin za zadanu bolničku pretragu ili pregled dobijemo tek za tri mjeseca, šest mjeseci, godinu dana? Često u takvim situacijama odemo u privatnu kliniku te osiromašimo svoj novčanik ili se jednostavno pomirimo sa sudbinom i križamo brojeve na kalendaru. Nešto su rjeđi slučajevi u kojima pacijenti samoinicijativno zovu bolnice u blizini kako bi provjerili postoji li možda skoriji termin za potrebnu im pretragu. Sve zbog izostanka međusobne povezanosti bolnica. Ne bi li bilo lakše da postoji jedno mjesto ili aplikacija gdje pacijent može vidjeti bolnice u blizini i najbliže termine za traženu pretragu?

Upravo je to svrha ovog projekta.

## 2 Opis projekta

Napravili smo aplikaciju koja pomoću relacijske baze podataka omogućuje fizičkoj osobi da se prijavi u sustav te, na temelju ovlasti, dobije potrebne informacije. Moguće je prijaviti se u ulozi pacijenta, liječnika opće prakse te cjelokupnog administratora.

**Pacijent**, kad je u potrebi, bira vrstu pretrage i šalje zahtjev osobnom liječniku. Ukoliko ga on odobri, pacijentu se nudi nekoliko najbližih bolnica u kojima pretragu može obaviti, zajedno s vremenom i datumom prvog slobodnog termina te bira željeni. Pacijent također može vidjeti povijest svojih pretraga te nadolazeće pretrage.

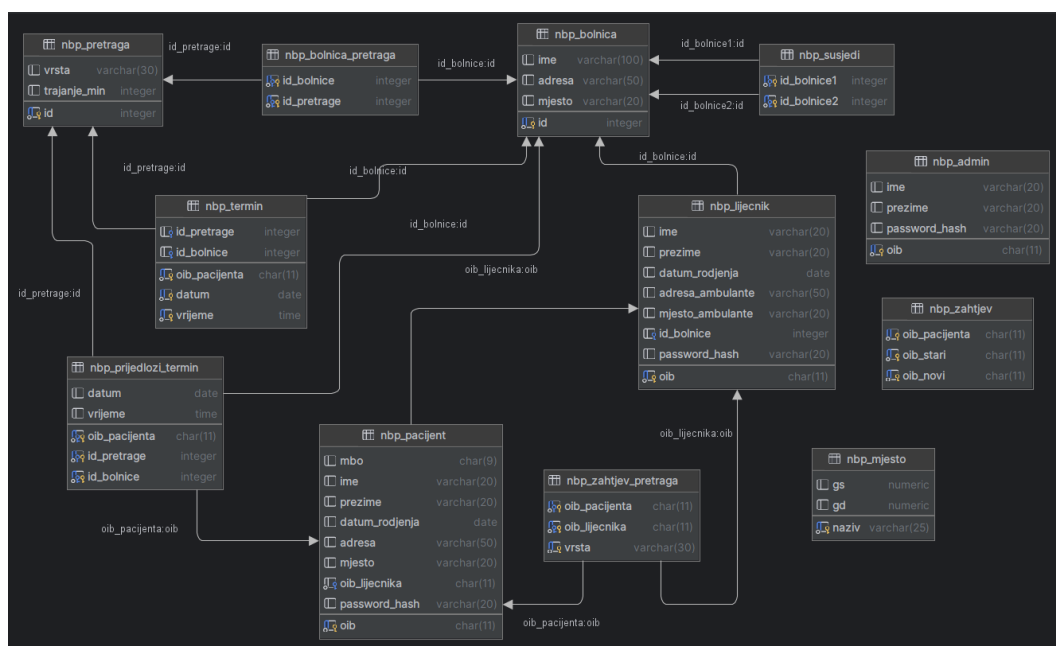
**Liječnik opće prakse** potvrđuje ili odbija pacijentov zahtjev za pretragom, unosi novog pacijenta te šalje zahtjev za prebacivanjem pacijenta drugom liječniku opće prakse. Sukladno tome, može prihvatiti ili odbiti zahtjev da osoba postane njegov pacijent. Također, može vidjeti popis i trajanje svih vrsta pretraga koje se nude u sustavu te popis svojih pacijenata zajedno s njihovim podacima.

**Administrator** cjelokupnog sustava uređuje sve podatke i ima uvid u sve što se zbiva u bazi.

### 3 Baza podataka

Kao što je navedeno u prethodnom *odjeljku*, za izradu aplikacije koristili smo relacijsku bazu podataka, čija se realizirana shema može vidjeti na Slici 1.

Ipak, prvotna je ideja bila kombinacija grafovske i relacijske baze podataka kako bismo, uvjetno rečeno, povezali pacijente s mrežom bolnica. U grafovskoj su se bazi trebali čuvati podaci o bolnicama, koje bismo povezali ukoliko su "susjedi", odnosno u gradovima koji su u krugu 75 km zračne udaljenosti jedan od drugog, te o pretragama povezanim s bolnicama koje ih nude. Ipak, zbog tehničkih poteškoća vezanih uz spajanje grafovske baze i PHP-a, odlučili smo se prebaciti samo na relacijsku. Same poteškoće bit će opisane u idućem *odjeljku*.



Slika 1: Dijagram baze

#### 3.1 Relacije

##### 3.1.1 ADMINISTRATOR

Jedna posebna tablica rezervirana je za osobe s administratorskim ovlastima. U njoj su sačuvani samo osnovni podaci, OIB i lozinka s kojima se prijavljuje te ime i prezime. Očekujemo da će u ovoj tablici biti svega nekoliko zapisa.

##### 3.1.2 BOLNICA

Poseban prostor rezerviran je i za pohranu svih kliničkih, općih i specijalnih bolnica, klinika, poliklinika te domova zdravlja u Republici Hrvatskoj ([6]). Pohranjuju se identifikacijski broj, ime i adresa bolnice te naselje u kojem se nalazi.

### 3.1.3 MJESTO

Sva naselja u Republici Hrvatskoj u kojoj se nalazi jedna od navedenih zdravstvenih ustanova čuvaju se, zajedno sa svojim geografskim koordinatama, u tablici `nbp_mjesto`. Ova je tablica važna za određivanje "susjednih" bolnica. Sam način opisan je *kasnije*.

### 3.1.4 SUSJEDI

Ova tablica odgovara vezi "JE\_SUSJED" između dvije bolnice. U njoj navodimo identifikacijske brojeve bolnica koje su u krugu jedna od druge najviše 75 km zračne udaljenosti. Točnije, mjesta u kojima se nalaze su toliko udaljena. Svaki se par bolnica namjerno javlja dva puta jer je samih parova 1498 i neće se više dodavati. 2996 zapisa u tablici nije previše, a omogućava pretraživanje susjednih bolnica samo po prvom atributu, na kojem smo odlučili napraviti i indeks jer je potraga za najbližim bolnicama zapravo poanta i glavni smisao ovog projekta.

### 3.1.5 PRETRAGA

U relaciji `nbp_pretraga` čuvaju se podaci o različitim pregledima i pretragama koje pojedine ustanove nude. Navedeno ih je nekoliko za primjer. Podaci koji se spremaju su šifra (ID) pretrage, vrsta te trajanje u minutama. Posljednji atribut važan je zbog određivanja termina pacijentima. U taj vremenski okvir uključeno je i potrebno čišćenje, sterilizacija, pisanje dokumentacije (nalaza, izvještaja...) te nekoliko minuta između svakog pacijenta. Naravno, ovo trajanje je okvirno, postoji mogućnost da se pretraga nekoj pacijenta oduži ili traje kraće, kao i uvijek. Ipak, znamo da se pacijentima preporuča dolazak neko razumno vrijeme prije zakazanog termina te naoružavanje strpljenjem prilikom svakog posjeta zdravstvenoj ustanovi.

### 3.1.6 PACIJENT

Dolazimo do primarnog korisnika ove aplikacije. Za svakog pacijenta čuvamo OIB i lozinku kojima se prijavljuje, osobne podatke (ime, prezime, datum rođenja, matični broj osiguranika ukoliko ga posjeduje, adresu i mjesto stanovanja) te OIB osobnog liječnika opće prakse, koji također mora biti spremljen u bazi.

### 3.1.7 LIJEČNIK OPĆE PRAKSE

Liječnik opće prakse također se prijavljuje u sustav pomoću osobnog identifikacijskog broja i lozinke. Uz navedeno, u tablici se čuvaju i osobni podaci: ime, prezime, adresa i mjesto njegove ambulate te identifikacijski broj najbliže bolnice. On je važan zbog odabira susjednih bolnica pojedinog pacijenta. Naime, kao što je prethodno opisano, svaki pacijent pripada jednom liječniku te mu se za pretrage nude matična bolnica, čiji se ID čuva kod pripadajućeg osobnog liječnika, i njoj susjedne bolnice navedene u tablici *Susjedi*.

### 3.1.8 BOLNICE I ODGOVARAJUĆE PRETRAGE

U bazi je rezervirano i mjesto gdje čuvamo uređene parove bolnica i tipova pretraga. Ne može se svaka pretraga obaviti u svakoj zdravstvenoj ustanovi, već samo u onima koje nude takvu vrstu pretrage. Ova tablica odgovarala bi vezi "NUDI" između bolnice i vrste pretrage.

### 3.1.9 ZAKAZANI TERMINI PRETRAGA

U posebnoj tablici pohranjujemo podatke o zakazanim terminima, bilo da su već prošli ili se još nisu održali. Pohranjujemo datum i vrijeme termina te OIB pacijenta, koji jedinstveno određuju svaki zakazani termin, zajedno s oznakama pretrage i zdravstvene ustanove.

### 3.1.10 ZAHTJEVI ZA PREBACIVANJEM PACIJENTA

Tablica `nbp_zah_tjev` služi za spremanje zahtjeva za prebacivanjem pacijenta kod drugog liječnika opće prakse, a koji može postaviti sam liječnik preko aplikacije. Prilikom prihvatanja zahtjeva od strane drugog liječnika ili administratora, on se iz tablice briše. Pohranjujemo OIB pacijenta te OIB-e njegovog trenutnog i budućeg liječnika.

### 3.1.11 ZAHTJEVI ZA PRETRAGOM

U trenutku kad pacijent putem aplikacije pošalje zahtjev za novom pretragom svom liječniku, on se sprema u tablicu `nbp_zah_tjev_pretraga`. Ona sadrži OIB-e pacijenta i njegovog liječnika te vrstu tražene pretrage. Nakon što liječnik prihvati ili odbije pacijentov zahtjev, on se iz tablice briše.

### 3.1.12 PRIJEDLOZI TERMINA

Ako je pacijentov zahtjev za pretragom prihvaćen, ponudit će mu se više opcija za termin u svim susjednim bolnicama koje nude tu pretragu. Te opcije bit će spremljene u `nbp_prijedlozi_termin`. Relacija sadrži OIB pacijenta, vrstu pretrage i ID bolnice, koji zajedno čine primarni ključ, te datum i vrijeme ponuđenog termina. Kad se pacijent odluči za termin pretrage, on će se pohraniti u `nbp_termin`, a ponuđene opcije se iz ove tablice brišu.

## 3.2 Indeksi

Zbog prirode aplikacije, prirodno su se nametnula tri indeksa za ubrzavanje pretraživanja.

```
1 CREATE INDEX pacijent_ime_idx ON nbp_pacijent(prezime, ime);
2 CREATE INDEX termin_pacijent_idx ON nbp_termin(oib_pacijenta);
3 CREATE INDEX susjedi_bolnica_idx ON nbp_susjedi(id_bolnice1);
```

Predviđamo da će se često pretraživati baza pacijenata po prezimenu i imenu pacijenta te njegovi dogovoreni termini po osobnom identifikacijskom broju. Također, sama poanta aplikacije traženje je bliskih bolnica svakom pacijentu, što smo dodatno ubrzali dodavanjem indeksa na prvi stupac tablice susjeda, odnosno identifikacijski broj jedne od susjednih bolnica.

## 3.3 Funkcije

U samoj bazi napisano je šest funkcija. U nastavku je naveden primjer jedne, dok su ostale samo opisane. Svaka se funkcija u cijelosti može pronaći na *linku*.

```
1 -- prvi slobodan termin
2
3 CREATE FUNCTION prvi_termin (v_id_bolnice INT, vrsta_P CHAR VARYING(20))
4 RETURNS table (
5     datum_termina TEXT,
6     vrijeme_termina TIME
7 )
8 AS $$
```

```

9 DECLARE
10     v_id_pretrage    INT;
11     v_trajanje       INT;
12     v_datum          DATE;
13     v_vrijeme        TIME;
14 BEGIN
15     SELECT id, trajanje_min INTO v_id_pretrage, v_trajanje
16     FROM nbp_pretraga
17     WHERE vrsta = vrsta_P;
18
19     SELECT datum, vrijeme INTO v_datum, v_vrijeme
20     FROM (
21         SELECT datum, vrijeme
22         FROM nbp_termin
23         WHERE id_pretrage = v_id_pretrage
24             AND id_bolnice = v_id_bolnice
25             AND datum > current_date
26         ORDER BY datum DESC, vrijeme DESC
27         LIMIT 1) AS tablica;
28
29     IF v_datum IS NULL
30     THEN
31         IF to_char(current_date, 'Day') = 'Friday' -- u ponedjeljak, ne u subotu
32         THEN
33             RETURN QUERY
34             SELECT (current_date + INTERVAL '3 days')::DATE AS datum, '
35             7:00'::TIME AS vrijeme;
36         ELSEIF to_char(current_date, 'Day') = 'Saturday' -- u ponedjeljak, ne u
37         nedjelju
38         THEN
39             RETURN QUERY
40             SELECT (current_date + INTERVAL '2 days')::DATE AS datum, '
41             7:00'::TIME AS vrijeme;
42         ELSE -- iduci dan
43             RETURN QUERY
44             SELECT (current_date + INTERVAL '1 day')::DATE AS datum, '7:00'::
45             TIME AS vrijeme;
46         END IF;
47     ELSIF v_vrijeme + v_trajanje * INTERVAL '1 minute' <= '18:00'::TIME
48     THEN
49         RETURN QUERY
50         SELECT v_datum::DATE AS datum, (v_vrijeme + v_trajanje * INTERVAL '1
51         minute')::TIME AS vrijeme;
52     ELSE
53         IF to_char(current_date, 'Day') <> 'Friday'
54         THEN
55             RETURN QUERY
56             SELECT (v_datum + INTERVAL '1 day')::DATE AS datum, '7:00'::TIME
57             AS vrijeme;
58         ELSE
59             RETURN QUERY
60             SELECT (v_datum + INTERVAL '3 days')::DATE AS datum, '7:00'::TIME AS
61             vrijeme;
62         END IF;
63     END IF;
64 END;
65 $$ LANGUAGE plpgsql;

```



### 3.3.1 Susjedne bolnice

Jedno smo vrijeme razmišljali kako odrediti koje su bolnice jedna blizu druge. Bilo je različitih ideja, no na kraju smo problemu doskočili na sljedeći način.

Implementirali smo funkciju `udaljenost(mjesto1 character varying(25), mjesto2 character varying(25))` koja na temelju koordinata naselja u kojima se nalaze bolnice računa njihovu zračnu udaljenost u kilometrima. Druga potrebna funkcija je `punjenje_susjedi()` koja prolazi po parovima svih zdravstvenih ustanova u bazi te ih ubacuje u tablicu `nbp_susjedi` ukoliko njihova zračna udaljenost nije veća od 75 km. Sama granica za zračnu udaljenost odabrana je otprilike, željeli smo da cestovna ili željeznička udaljenost bude najviše stotinjak kilometara, a tome otprilike odgovara ova zračna udaljenost.

### 3.3.2 Popis pacijenata i pretraga

Treća funkcija, `popis_pacijenata(oib_L CHAR(11))`, služi za dohvaćanje popisa pacijenata i njegovih podataka traženog liječnika.

Svaki pacijent može vidjeti povijest svojih pretraga. Naravno, to može vidjeti i njegov osobni liječnik. Funkcija `povijest_pretraga(oib CHAR(11))` vraća datum, vrijeme, naziv, zdravstvenu ustanovu te mjesto svake pretrage koju je pacijent obavio.

### 3.3.3 Lista čekanja i slobodni termini

Administrator može za svaku bolnicu i vrstu pretrage vidjeti listu čekanja. Ta funkcionalnost nije javno dostupna zbog zaštite podataka, s obzirom da funkcija `lista_cekanja(ime_bolnice CHAR VARYING(30), mjesto_bolnice CHAR VARYING(20), vrsta_P CHAR VARYING(20))` vraća datum i vrijeme pretrage te osobni identifikacijski broj pacijenta. Važno je naglasiti da je kao argument navedeno i mjesto bolnice jer se u bazi nalaze i neke bolnice istog naziva, ali u različitim mjestima.

Ipak, pacijent, nakon što mu je odobren zahtjev za pretragom, može vidjeti prvi slobodan termin za svaku bolnicu u susjedstvu i svaku željenu vrstu pretrage. Za to je zadužena funkcija `prvi_termin(v_id_bolnice INT, vrsta_P CHAR VARYING(20))`, čiji se čitav kod može vidjeti na početku ovog odjeljka.

## 3.4 Ubacivanje podataka

Bazu smo testnim podacima punili na različite načine.

Bolnice su unesene ručno pomoću podataka s izvora [6]. Naredbom `select distinct mjesto from nbp_bolnica`; nađen je popis svih naselja u kojima se nalaze bolnice. Nije ih bilo previše, stoga su se ručno unijele koordinate za svaki prema stranici s izvora broj [1].

Tablica sa susjednim bolnicama popunjena je pomoću dvije funkcije iz odjeljka 3.3.1.

Pacijenti i liječnici opće prakse osmišljeni su i uneseni ručno.

Što se pretraga tiče, nakon konzultacija s kolegom na Medicinskom fakultetu, unesene su neke pretrage koje bi doista trebale biti u aplikaciji. Na to, primjera radi, dodane su još neke pretrage na koje bi se mogli pacijenti naručivati i u druge zdravstvene ustanove osim matične. Svakako bi se ova tablica popunila mnogobrojnim pretragama nakon konzultacija s medicinskim stručnjacima. Nadalje, parovi bolnica i pretraga koje nude unesene su pomoću online *generatora nasumičnih parova brojeva*.

Termini su uneseni ručno te smo naknadno funkcijama u SQL-u provjerili da navedeni datumi nisu subota ili nedjelja kako bi nam podaci bili konzistentni. Naravno da smo nasumično ubaci-

vali termine, neke iz prošlosti, neke iz budućnosti, no u stvarnosti bi se očekivalo da svaki termin do najkasnijeg datuma bude popunjen. Na toj pretpostavci temelji se i implementacija funkcije *prvi\_termin*.

## 4 Izgradnja aplikacije

Budući da dvije trećine našeg tima sluša kolegij Računarski praktikum 2 (RP2), odlučili smo izgraditi aplikaciju koristeći PHP i MVC (Model-View-Controller) oblik aplikacije.

### 4.1 Spajanje na bazu

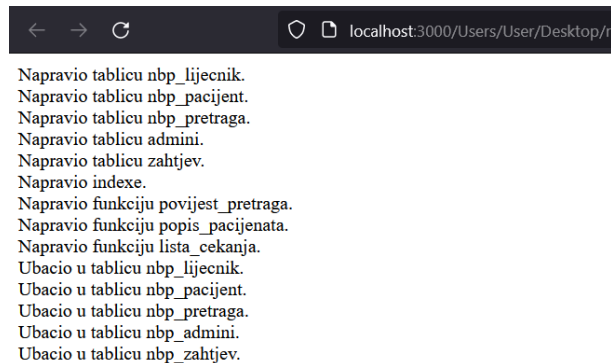
Odmah na početku naišli smo na više problema. Na kolegiju RP2 koristili smo phpmyadmin za našu bazu (<https://www.phpmyadmin.net/>) te smo ga iz istog razloga planirali koristiti i na ovom projektu, ali s obzirom da tamo nismo radili kompliciranije baze, nismo znali da tamo ne postoje stvari poput funkcija, okidača itd. Također jedan od problema je bio što phpmyadmin koristi MySQL, dok je nama trebao postgresql. Zbog toga smo bili prisiljeni tražiti alternativu.

Kratkim istraživanjem, saznali smo način za spajanje na bazu "odbojka" korištenu u prvoj domaćoj zadaći. Jedino je bilo potrebno instalirati drivere za rad sa postgresom (detalji instalacije nalaze se u *GitHub repozitoriju*). Nakon toga zatraži se spajanje preko VCL sustava Srca i prekopira se IP adresa u datoteku *db.class.php*.

```
1 public static function getConnection()
2 {
3     if( DB::$db === null )
4     {
5         try
6         {
7             //ovdje treba promijeniti IP hosta nakon povezivanja sa VCL
8
9             DB::$db = new PDO('pgsql:host=31.147.200.189; port=5432; dbname=odbojka;
10 user=postgres; password=password' );
11             DB::$db->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
12         }
13         catch( PDOException $e ) { exit( 'PDO Error: ' . $e->getMessage() ); }
14     }
15     return DB::$db;
16 }
```

Za sljedeći korak potrebno je imati neki oblik PHP servera. Mi smo koristili Apache i XXAMP (detalje instalacije možete pronaći u [5]). Naravno, uz sve to, potrebno je imati instaliran i sam PHP.

Nakon svih instalacija otvorimo datoteku *prepareDB.php*. U njoj se nalaze potrebne naredbe za stvaranje baze, svih popratnih funkcija, okidača i slično, te atributi sa popunjavanje same baze. Nakon pokretanja otvara se prozor u pregledniku te se ispisuju popratne poruke ako su sve baze pravilno napravljene i popunjene. U suprotnom, ispisuje se gdje je nastao problem.



```
← → ↺ localhost:3000/Users/User/Desktop/n
Napravio tablicu nbp_lijecnik.
Napravio tablicu nbp_pacijent.
Napravio tablicu nbp_pretraga.
Napravio tablicu admini.
Napravio tablicu zahtjev.
Napravio indexe.
Napravio funkciju povijest_pretraga.
Napravio funkciju popis_pacijenata.
Napravio funkciju lista_cekanja.
Ubacio u tablicu nbp_lijecnik.
Ubacio u tablicu nbp_pacijent.
Ubacio u tablicu nbp_pretraga.
Ubacio u tablicu nbp_admini.
Ubacio u tablicu nbp_zahtjev.
```

Slika 2: Primjer poruka za pravilno napravljenu i popunjenu bazu

Nakon uspostavljanja relacijske baze, krenuli smo razvijati aplikaciju misleći da ćemo kasnije pronaći i način za spajanje na grafovsku bazu. Nažalost, napravili smo već 50% posla sigurni da s time neće biti problema. Pokušali smo više načina:

- (a) Connect to Neo4j with PHP  
(<https://medium.com/neo4j/connect-to-neo4j-with-php-e10e24afedff>)
- (b) <https://github.com/neo4j-php/neo4j-php-client>
- (c) <https://github.com/neo4j-php/Bolt>
- (d) <https://github.com/mdujic/student-subject-network/tree/main>

Proveli smo više dana bezuspješno slijedeći sve upute. Posebno obećavajuća činila nam se četvrta metoda, s obzirom da ju je napravio tim za svoj projekt iz Naprednih baza podataka prije dvije godine, ali ni ona nije urodila plodom. Nikoli je zapelo pri instalaciji Composer-a, dok je Nora uspjela doći sve do instalacije Dockera, gdje se našla pred zidom. Na profesorovo dopuštenje, ipak smo prebacili cijeli projekt u relacijsku bazu.

## 4.2 Izgled i funkcionalnost aplikacije

Nakon riješenog posla oko spajanja na bazu, krenuli smo s razvojem aplikacije koja će tu bazu i koristiti.

Ako koristite neki od prije navedenih lokalnih PHP servera, aplikaciju možete pokrenuti iz datoteke *index.php*. Nakon pokretanja, odvest će vas na ekran za prijavu. Na njemu korisnik može odabrati želi li se prijaviti kao liječnik, pacijent ili administrator. Na početku smo imali jednu formu za prijavu, ali smo shvatili da se može dogoditi situacija gdje imamo liječnika koji je ujedno i pacijent. U tom slučaju nam naša php logika ne bi funkcionirala jer smo prvo provjeravali nalazi li se korisnik u tablici liječnika, zatim ako se ne nalazi u toj tablici, gledamo tablicu pacijenta, te na kraju ako se ne nalazi ni u toj, gledamo tablicu administratora. Kod takve implementacije, aplikacija bi prvo provjerila nalazi li se korisnik u tablici s liječnicima, pronašla ga i prijavila kao liječnika, iako se liječnik htio prijaviti kao pacijent. Raspodjelom prijave na tri različite prijave riješili smo taj problem. Na sljedećoj slici može se vidjeti navedena forma:

Slika 3: Ekran za prijavu (login)

Korisnici aplikacije prijavljuju se pomoću svog OIB-a i lozinke. Podijelili smo ih u tri kategorije:

- (a) Liječnik - razina prava 0
- (b) Pacijent - razina prava 1
- (c) Administrator - razina prava 2

Nakon prijave, ovisno o korisnikovim pravima, pojavljuje se odgovarajuća početna stranica. Ovdje navodimo 3 korisnika pomoću kojih je moguća prijava u sustav:

- (a) Liječnik: OIB="10000444444" i lozinka="gandalf123"
- (b) Pacijent: OIB="10000338099" i lozinka="frodo123"
- (c) Administrator: OIB="10000338023" i lozinka="admin123"

Za ostale korisnike istog tipa možete pogledati u *prepareDB.php*.

Slika 4: Primjer početne stranice za administratora

Početna stranica je stranica profila prijavljenog korisnika. Glavna razliku između korisnika čini navigacijska traka. Ovisno o njegovoj razini prava, mijenja se i navigacijska traka. Primjerice, dok administrator ima "apsolutnu moć", obični pacijent neće imati sve mogućnosti kao administrator, a ni kao liječnik.



Slika 5: Primjer početne stranice za pacijenta

O provjeri razine prava brinu se *loginController.class.php* u direktoriju Controller i *loginservice.class.php* u direktoriju model. Oni pri prijavi u bazi provjeravaju gdje se točno nalazi korisnik. Ovisno o tome u kojoj se tablici nalazi (liječnici, pacijenti ili administratori), oni spremaju kolačić za ovlast tog korisnika, koji se provjerava pri ispisu pripadajuće navigacijske trake kod svakog prelaska s jedne stranice na drugu. Možemo vidjeti dio koda u *loginservice.class.php* gdje se to postavlja:

```

1  // Je li dobra lozinka?
2  if( password_verify( $password, $hash ))
3  {
4      // Ako je korisnik ulogiran od prije
5      // znaci da je ovo provjera pri mijenjanju sifre
6      if(isset($_COOKIE['oib'])){
7          return 1;
8      }
9      // Dobar password. Ulogiraj ga i posalji na pocetni ekran.
10     // Moramo dohvatiti i njegove ovlasti
11     setcookie('oib',$oib,time()+(10*365*24*60*60));
12     setcookie('ovlasti',$ovlasti,time()+(10*365*24*60*60));
13     // Ova linija je potrebna da se cookie zapamti pri prvom ulasku na stranicu
14     header("Location: index.php");
15
16     require_once __DIR__ . '/../controller/profilController.class.php';
17     $od=new ProfilController();
18     $od->index();
19     return 1;
20 }

```

Također dio koda u *\_header.php* gdje se provjerava kolačić s ovlasti:

```

1  <?php
2  if(isset($_COOKIE['ovlasti']) && $_COOKIE['ovlasti'] == '0'){
3      require_once __DIR__ . '/navigacija-lijecnik.php';
4  }
5  else if(isset($_COOKIE['ovlasti']) && $_COOKIE['ovlasti'] == '1'){
6      require_once __DIR__ . '/navigacija-pacijent.php';
7  }
8  else{
9      require_once __DIR__ . '/navigacija-admin.php';
10 }
11 ?>

```

U odjeljku *Moji podaci* korisnik može vidjeti i promijeniti svoje podatke:

Slika 6: Promjena podataka

Naravno, ne dopuštamo mijenjanje nekih podataka poput OIB-a ili OIB-a liječnika, već samo podatke poput imena, prezimena, datuma rođenja i slično. Ovisno jesu li uneseni pravilni ili nepravilni podaci, ispiše se prikladna poruka.

Sada ćemo proći kroz poglede svakog od tri tipa korisnika i koje funkcionalnosti oni imaju.

#### 4.2.1 Liječnik

Prva funkcionalnost koju liječnik ima vezana je uz pacijente. U slučaju da klikne na gumb *Pacijenti*, ponudit će mu se više opcija. Odabere li *Popis mojih pacijenata*, ispisat će mu se popis svih pacijenata u bazi koji su povezani liječnikovim OIB-om.

(a) Padajući izbornik za liječnika

(b) Pacijenti liječnika Gandalfa

Slika 7: Primjer liječnikovih mogućnosti

Ovdje možemo vidjeti kod za dohvat svih pacijenata koji se nalazi u *pacijentservice.class.php*:

```

1 function getmojipacijenti($oib_lijecnika){
2     try
3     {
4         $db = DB::getConnection();
5         $st = $db->prepare("SELECT * FROM popis_pacijenata(:oib_lijecnika) ORDER BY
        prezime, ime");
    
```

```

6      $st->bindParam(':oib_lijecnika', $oib_lijecnika, PDO::PARAM_STR);
7      $st->execute();
8  }
9  catch( PDOException $e ) { exit( 'PDO error ' . $e->getMessage() ); }
10
11  $arr=array();
12  while(1){
13      $row = $st->fetch();
14      if( $row === false )
15          return $arr;
16      else{
17          $i=new Pacijent($row['oib'],$row['mbo'],
18              $row['ime'],$row['prezime'],$row['datum_rodjenja'],
19              $row['adresa'],$row['mjesto'],$oib_lijecnika);
20          $arr[]=$i;
21      }
22  }
23  return $arr;
24  }

```

U slučaju da želi unijeti novog pacijenta, liječnik mora kliknuti na *Unos novog pacijenta*, na što će mu se ispisati forma. U slučaju pravilnog ispunjavanja, pacijent se dodaje u bazu i ispisuje se prikladna poruka.

**Unos novog pacijenta**

OIB:

MBO:

Ime:

Prezime:

Lozinka:

Potvrdite lozinku:

Datum rođenja:

Adresa:

Mjesto:

**Dodaj!**

Slika 8: Forma za unos novog pacijenta



Liječnik također može poslati ili primiti zahtjev za prebacivanjem pacijenata. U slučaju da želi prebaciti pacijenta nekog drugom liječniku, mora odabrati opciju *Novi zahtjev*. Na tom će ekranu u padajućem izborniku moći odabrati kojem liječniku želi poslati zahtjev za prebacivanjem odabranog pacijenta. Pritom, u ponudi će mu biti samo drugi liječnici s područja njegove bolnice. U ovom slučaju pretpostavljamo da pacijent želi samo promijeniti liječnika na određenom području, a ne da se seli.

### Zahtjev za prebacivanjem pacijenta

Odaberite drugog liječnika iz Vaše ambulante za željenog pacijenta.

	OIB	MBO	Ime	Prezime	Datum rođenja	Adresa	Mjesto	
Ednew, Theoden ▾	10000338099	100338099	Frodo	Baggins	2000-07-02	Glavna 1	Hobbiton	Spremi
Ednew, Theoden ▾	10000395731	100395731	Peregrin	Took	2001-07-02	Glavna 4	Buckland	Spremi

Slika 9: Postavljanje zahtjeva za prebacivanje pacijenta

Nakon što pošalje zahtjev, on stiže odabranom liječniku koji ga može vidjeti u odjeljku *Pristigli zahtjevi*. Tamo bira želi li primiti tog pacijenta. Na prošlom smo zaslonu kao liječnik Gandalf odabrali slanje zahtjeva liječniku Theodenu, a na sljedećoj slici vidimo da je Theoden i primio taj zahtjev.

### Zahtjevi za prebacivanje pacijenta

Zahtjev postavlja:	OIB pacijenta	MBO	Ime	Prezime	Datum rođenja	Adresa	Mjesto	
Mithrandir, Gandalf	10000338099	100338099	Frodo	Baggins	2000-07-02	Glavna 1	Hobbiton	Prihvati

Slika 10: Zaslom za prihvaćanje pacijenata

Sljedeća funkcionalnost liječnika su *Pretrage*. Postavljanjem miša iznad tog gumba ponudit će nam se dvije opcije. Prva je *Popis pretraga* koja pokaže popis svih pretraga koje se mogu izvršiti u našem sustavu bolnica, te njihovo trajanje. Druga opcija je *Zahtjevi za pretragama* koja ispiše sve zaprimljene zahtjeve za pretragama od pacijenata tog doktora. Obje se opcije mogu vidjeti na Slici 11.

Naravno, liječnik može prihvatiti ili odbiti zahtjev. U slučaju da ga odbije, zahtjev će se samo obrisati iz baze, a u slučaju da prihvati, pacijentu će se ponuditi prvi slobodni termini u svim najbližim bolnicama ustanove povezane s njegovim liječnikom. Za to smo iskoristili tablicu susjednih bolnica te funkciju *prvi termin*, koja je u cijelosti navedena u prošlom *odjeljku*. Nakon toga, zahtjev se briše iz baze.

## Vrste pretraga

ID	Vrsta	Trajanje
1	dermatološki pregled	45 minuta
2	oftalmološki pregled	45 minuta
3	fizikalna medicina	45 minuta
4	magnetska rezonanca	50 minuta
5	serologija	10 minuta
6	dijabetologija	30 minuta

(a) Vrste pretraga

## Zahtjevi za novom pretragom

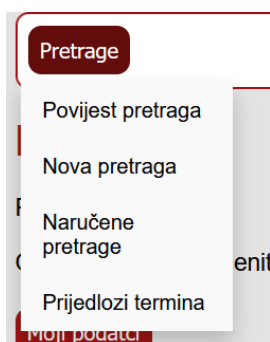
Zahtjev postavlja:	OIB pacijenta	Vrsta pretrage		
Baggins, Frodo	10000338099	dermatološki pregled	Prihvati	Odbij
Baggins, Frodo	10000338099	dijabetologija	Prihvati	Odbij
Took, Peregrin	10000395731	fizikalna medicina	Prihvati	Odbij

(b) Zahtjevi za pretragama liječnika Gandalfa

Slika 11: Pretrage i zahtjevi

### 4.2.2 Pacijent

Funkcionalnost koju pacijent ima vezana je uz pretrage. Klikom na gumb *Pretrage* otvara mu se padajući izbornik s više opcija. Odabere li *Povijest pretraga*, može vidjeti popis svojih obavljenih pretraga. Klikom na *Nova pretraga*, pacijent može poslati zahtjev svom liječniku za neku od ponuđenih pretraga iz padajućeg izbornika. Na Slici 12 pacijent Frodo šalje zahtjev za dermatološki pregled. Na Slici 11 prikazano je kako to izgleda s liječničke strane.



(a) Padajući izbornik za pretrage

## Zahtjev za novom pretragom

Pošaljite svom liječniku opće prakse zahtjev za uputnicom.

Dermatološki pregled ▾

Pošalji!

(b) Slanje zahtjeva za pretragu

## Povijest mojih pretraga

Datum	Vrijeme	Vrsta	Ime bolnice
2011-07-02	11:00:00	dermatološki pregled	Opća bolnica Varaždin
2010-07-02	12:00:00	oftalmološki pregled	Dom zdravlja "Dr. A. Franulović" Vela Luka

(c) Primjer povijesti pretraga pacijenta Frode

Slika 12: Pacijentove funkcionalnosti

U slučaju da liječnik odobri zahtjev pacijenta, pacijent ponuđene termine može vidjeti klikom na *Prijedlozi termina*.

## Prijedlozi termina

Dermatološki pregled:

Datum	Vrijeme	Bolnica	
2024-06-09	07:00:00	Klinika za dječje bolesti, Zagreb	Prihvati
2024-06-09	07:00:00	Specijalna bolnica za medicinsku rehabilitaciju Lipik, Lipik	Prihvati
2024-06-09	07:00:00	Specijalna bolnica za produženo liječenje - Duga Resa, Duga Resa	Prihvati
2024-06-09	07:00:00	Poliklinika za bolesti dišnog sustava, Zagreb	Prihvati
2024-06-09	07:00:00	Poliklinika za fizikalnu medicinu i rehabilitaciju dr. Drago Čop, Zagreb	Prihvati
2024-06-09	07:00:00	Poliklinika za rehabilitaciju slušanja i govora Suvača, Zagreb	Prihvati

Slika 13: Slanje zahtjeva za pretragu

Zadnja funkcionalnost je *Naručene pretrage*. Klikom na tu opciju, pacijent dobije ispis svih pretraga koje ga čekaju.

## Naručene pretrage

Datum	Vrijeme	Vrsta	Ime bolnice
2024-07-02	16:00:00	serologija	Opća bolnica "Dr. Ivo Pedišić" Sisak

Slika 14: Naručene pretrage pacijenta Frode

### 4.2.3 Administrator

Administrator ima "apsolutnu moć". Može nadgledati sve podatke i odobravati zahtjeve u ime liječnika. Jedina razlika s obzirom na prijašnja dva tipa korisnika je da administrator može vidjeti sve pacijente i liječnike te mijenjati podatke svakog od njih, dok je, primjerice, svaki liječnik mogao to raditi samo za svoje pacijente te nije mogao vidjeti ostale liječnike.

Jedna posebna funkcionalnost koju imaju administratori vezana je uz bolnice. Mogu dobiti popis bolnica u bazi, mijenjati podatke svih bolnica te ih brisati iz baze.

## Bolnice

ID	Ime	Adresa	Mjesto	
1	Opća bolnica "Dr. Ivo Pedišić" Sisak	J.J. Strossmayera 59	Sisak	Susjedi
2	Opća bolnica "Dr. Josip Benčević" Slavonski Brod	Andrije Štampara 42	Slavonski Brod	Susjedi
3	Opća bolnica "Dr. Tomislav Bardek" Koprivnica	Željka Selinger bb	Koprivnica	Susjedi
4	Opća bolnica "Hrvatski ponos" Knin	Svetoslava Suronje 12	Knin	Susjedi
5	Opća bolnica Bjelovar	Mihanovićeve 8	Bjelovar	Susjedi
6	Opća bolnica Dubrovnik	Roka Mišetića 2	Dubrovnik	Susjedi
7	Opća bolnica Gospić	Kaniška 111	Gospić	Susjedi
8	Opća bolnica Karlovac	Andrije Štampara 3	Karlovac	Susjedi
9	Opća bolnica Ogulin	Bolnička 38	Ogulin	Susjedi

Slika 15: Ispis svih bolnica u bazi

Klikom na *Susjedi* iskoristava se tablica `nbp_susjedi` opisana u prethodnom *odjeljku* kako bi se ispisali svi susjedi te bolnice.

## Susjedi bolnica

Susjedi od Opća bolnica "Dr. Ivo Pedišić" Sisak, Sisak

Ime	Adresa	Mjesto
Dom zdravlja Sisak	Kralja Tomislava 1	Sisak
Dom zdravlja Duga Resa	Bana J. Jelačića 4	Duga Resa
Dom zdravlja Zagreb - Istok	Švarcova 20	Zagreb
Dom zdravlja Zagrebačke županije	Ljudevita Gaja 37	Samobor
Dom zdravlja Zagreb - Zapad	Prilaz baruna Filipovića 11	Zagreb
Dom zdravlja Zagreb - Centar	Runjaninova 4	Zagreb
Dom zdravlja Vojnić	A.Hebranga 24	Vojnić
Dom zdravlja Slunj	Plitvička 18a	Slunj
Dom zdravlja Petrinja	Matije Gupca 4	Petrinja
Dom zdravlja Ozalj	Koledvorska 2	Ozalj

Slika 16: Primjer ispisa svih susjeda jedne bolnice

## 5 Zaključak

Prilikom izrade ove aplikacije, susreli smo se s brojnim poteškoćama, od nešto manjih poput prvog susreta s nekim tehnologijama, primjerice statističarka + GitHub, do one najzahtjevnije koja nas je na posljetku i porazila, spajanja grafovske baze i PHP-a. Ipak, relacijska baza ne čini se lošom izvedbom jer je ponudila i više mogućnosti nego smo u početku zamislili.

Timskim radom svladali smo prepreke jer je svatko od nas spretniji u nekom području, a kvalitetno izvršavanje ovog projektnog zadatka kombiniralo je nekoliko područja: ideje i kreativnost, modeliranje, programiranje, snalaženje u tehnologijama poput GitHuba i LaTeX-a, prezentacijske vještine pa čak i jezično izražavanje.

Usprkos problemima s kojima smo se susreli, zadovoljni smo konačnim proizvodom za koji nam se čini da vjerodostojno prikazuje količinu truda uloženu u njegov nastanak. Naravno, svaka se aplikacija s vremenom može nadograđivati, a već sada možemo navesti prijedloge za neka poboljšanja.

Prvo i osnovno, kako bi se aplikacija doista mogla upotrebljavati, potrebne su konzultacije s liječnicima i medicinskim osobljem koji bi predložili pretrage za unos u sustav. Već u početnom razgovoru sa studentom medicine, shvatili smo da neke pretrage koje smo željeli ubaciti, primjerice CT ili rendgen, nema smisla imati u aplikaciji zbog brzine njihovog izvršavanja i očitavanja nalaza. Ova sitnica naučila nas je važnosti interdisciplinarne suradnje u ovakvim projektima.

Sljedeća funkcionalnost koju bismo predložili je pohrana termina pretraga starijih od deset godina na nekom drugom mjestu te njihovo brisanje iz baze aplikacije zbog količine zapisa.

Još jedna važna stvar koja bi s vremenom moglo unaprijediti performanse aplikacije su indeksi. Konkretno, s vremenom bi se uočila potreba za eventualnim stvaranjem novih indeksa ili brisanjem postojećih ukoliko smo mi kao laici pogrešno protumačili potrebe korisnika.

## 6 Samoevaluacija

### 6.1 Nora Berdalović

Radila sam pretežno na izradi aplikacije, uključujući veći dio funkcionalnosti koje se nude korisniku. Pri tome sam po potrebi prilagođavala našu bazu potrebama aplikacije, uključujući ubacivanje podataka, dodavanje tablica te manje promjene funkcija. Rekla bih da sam ovim projektom dodatno utvrdila znanje o radu u MVC okviru te povezivanju baze u Postgresu s PHP-om.

### 6.2 Laura Horvat

Osmislila sam ideju te, nakon razgovora s ostalim članovima tima, raspisala i detaljnije ju razradila. Razgovarala sam s kolegom s Medicinskog fakulteta o smislenosti i funkcionalnostima naše ideje te od kolege Dujića tražila pomoć i savjete za spajanje na grafovsku bazu. Nažalost, nismo uspjeli, no to se ipak nije pokazalo toliko loše. Budući da nisam računarac, moj je zadatak bio izgraditi bazu zajedno s potrebnim indeksima, funkcijama i svime što dolazi s njom u paketu. Rekla bih da sam prilično utvrdila ovaj dio gradiva kolegija. Uz Nikolu, s obzirom da smo spretniji u LaTeX-u, pisala sam dokumentaciju. Provjeravala sam i ispravljala jezične, gramatičke i stilske pogreške u tekstu. Manje sam spretna u korištenju gita i GitHuba te sam se u tome susrela s malim poteškoćama, stoga ovim putem zahvaljujem kolegama iz tima na razumijevanju i uskakanju u tom pogledu.

### 6.3 Nikola Kašnar

Uglavnom sam radio na izradi same aplikacije zajedno s Norom. Na početku smo više vremena istraživali načine za spajanje na grafovsku bazu, ispitujući pritom razne kolege sa smjera, ali nažalost nismo u tome uspjeli. Glavni dio koji sam implementirao je raspodjela uloga te briga oko same prijave u aplikaciji i razlika u ispisu ovisno o tome tipu korisnika. Pri razvoju sam rješavao razne bugove i poteškoće koje su se pojavljivale. Zajedno s Laurom pisao sam dokumentaciju u LaTeX-u te sam se u njoj najviše pobrinuo za četvrto poglavlje o samom razvoju aplikacije. Pri razvoju aplikacije, produbio sam znanje o korištenju gita i GitHuba (merge u slučaju konflikata, fetch itd.) i naučio koristiti funkcije u aplikaciji u sklopu baze (na drugim kolegijima dosad smo radili samo s najosnovnijim bazama) te mislim da će mi to puno pomoći pri izradi budućih projekata. Također sam napravio prezentaciju projekta i prezentirao ju.

## 7 Literatura

- [1] Geonames - koordinate gradova (<https://public.opendatasoft.com/explore/dataset/geonames-all-cities-with-a-population-1000>)
- [2] Naš GitHub repozitorij (<https://github.com/ring-bearer/nbp>)
- [3] Prezentacije iz kolegija Napredne baze podataka
- [4] Spajanje na Neo4j bazu s PHP-om (<https://neo4j.com/developer-blog/connect-to-neo4j-with-php/>)
- [5] Stranice kolegija RP2 (<https://web.math.pmf.unizg.hr/nastava/rp2d/>)
- [6] Zdravstvene ustanove u RH (<https://zdravlje.gov.hr/arhiva-80/ministarstvo-zdravlja/zdravstvene-ustanove-u-republici-hrvatskoj/656>)