

**SVEUČILIŠTE U ZAGREBU
FAKULTET ORGANIZACIJE I INFORMATIKE
VARAŽDIN**

Tomislav Kolar

WEB APLIKACIJA ZA POVEZIVANJE LIJEČNIKA, LJEKARNIKA I PACIJENTA

SEMINARSKI RAD

Varaždin, 2022.

SVEUČILIŠTE U ZAGREBU
FAKULTET ORGANIZACIJE I INFORMATIKE
V A R A Ž D I N

Tomislav Kolar

Matični broj: 35918/07–R

Studij: Baze podataka i baze znanja

**WEB APLIKACIJA ZA POVEZIVANJE LIJEČNIKA, LJEKARNIKA I
PACIJENTA**

SEMINARSKI RAD

Mentor:

Dr. sc. Bogdan Okreša Đurić

Varaždin, veljača 2022.

Sadržaj

1. Aplikacijske domene	1
2. Teorijski uvod	2
2.1. Aktivna baza podataka	2
2.2. Temporalna baza podataka	3
2.3. Poopćene baza podataka	3
3. Model baze podataka	4
3.1. Implementacija	5
4. Primjeri korištenja	8
5. Zaključak	11
Popis literature	12
Popis slika	13

1. Aplikacijske domene

Cilj projekta je izrada web aplikacije za povezivanje liječnika, ljekarnika i pacijenta. Aplikacija omogućuje povezivanje liječnika, ljekarnika i pacijenta na način da nakon fizičkog posjeta pacijenta liječnika. Liječnik kreira zapis "Pregled" koji sadrži dijagnozu, popis lijekova koji su mu potrebni za liječenje, ako postoji slika nekog nalaza može ju pohraniti. Kako je liječnik prepisao lijekove, pacijent ih može podići u određenoj ljekarni. Tako ljekarnik i pacijent komuniciraju. Pacijent ima mogućnost naručivanja kod liječnika, kao i mogućnost otkazivanja istog.

Za izradu ove aplikacije korišteni su PHP, JavaScript za komunikaciju između korisnika i servera, a za pohranu podataka korištena je PostgreSQL je sustav za upravljanje bazama podataka. Za odabranu tehnologiju sam se odlučio zbog toga što je poznavanje JavaScript-a traženo na fakultetu, a osim na fakultetu koristi se i na tržištu rada, za PHP sam se odlučio zbog jednostavnosti između komunikacije JavaScripta sa serverom, a komunikacija se vrši AJAX zahtjevima. Za sustav PostgreSQL sam se odlučio što je sustav besplatan, te modeliranje baze je vrlo jednostavno.

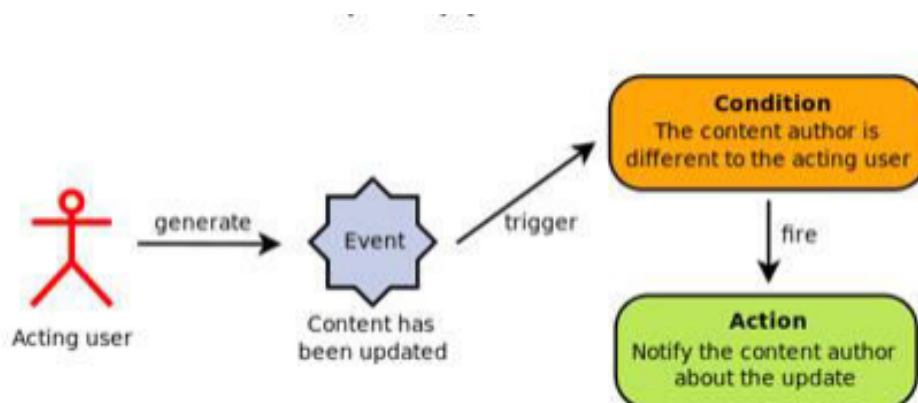
Link za Overleaf: <https://www.overleaf.com/read/yppndccbdyyr>

2. Teorijski uvod

Za izradu ovog projekta bilo je potrebno koristiti aktivnu, temporalnu i poopćenu bazu podataka.

2.1. Aktivna baza podataka

Aktivna baza podataka daje mogućnost korisniku manipulaciju nad podacima kako bi podaci koji se spremaju podaci bili točni, konzistentni, ali i bili kontrolirani. Aktivne baze podataka se smatraju one baze podataka koje mogu reagirati na događaje, a na događaje se podrazumijeva unos, brisanje i ažuriranje. Aktivna baza ponaša se tako da na neki događaj koji se pojavio u datom trenutku aktivira svoje okidače i provjeri odgovara li npr. novi unos pravilu koje je zadana u okidaču. Okidači kontroliraju događaje i oni su unutar baze podataka te se samostalno izvode ovisno o uvjetu koji se događa. Prednosti aktivnih baza podataka u odnosu na neaktivne su: efikasno izvođenja funkcija koje se u neaktivnim moraju ugraditi u samu aplikaciju, aktivni sustavi mogu izvoditi zadatke koji u pasivnim sustavima zahtijevaju posebne podsustave. Implementacija aktivnih baza podataka vrši se pomoću okidača (eng. triggera).



Slika 1: Tijek rada aktivne baze (Izvor: <https://www.drupal.org/files/drupalECA.png>)

Prema slici 1 vidi se kako korisnik generira događaj u ovom slučaju je ažuriranje, na taj događaj se aktivira okidač te svojim okidanjem šalje obavijest autoru o njegovom ažuriranju. Događaj je dio aktivne baze podataka, događaj je sve što pokušava raditi s bazom podataka tako da manipulira podacima što može biti jednostavno ili složeno. U bazama podataka složeni događaji najčešće nastaju sšajanjem jednostavnih događaja u složene. Okidači se najčešće kreirali kako bi pratili operacije nad podacima, nad izmjenama vrijednosti vremenskih događaj. Zbog toga su bitna pravila koja određuju što je dopušteno u bazi, a što nije koja provode okidači.**aktivna**

2.2. Temporalna baza podataka

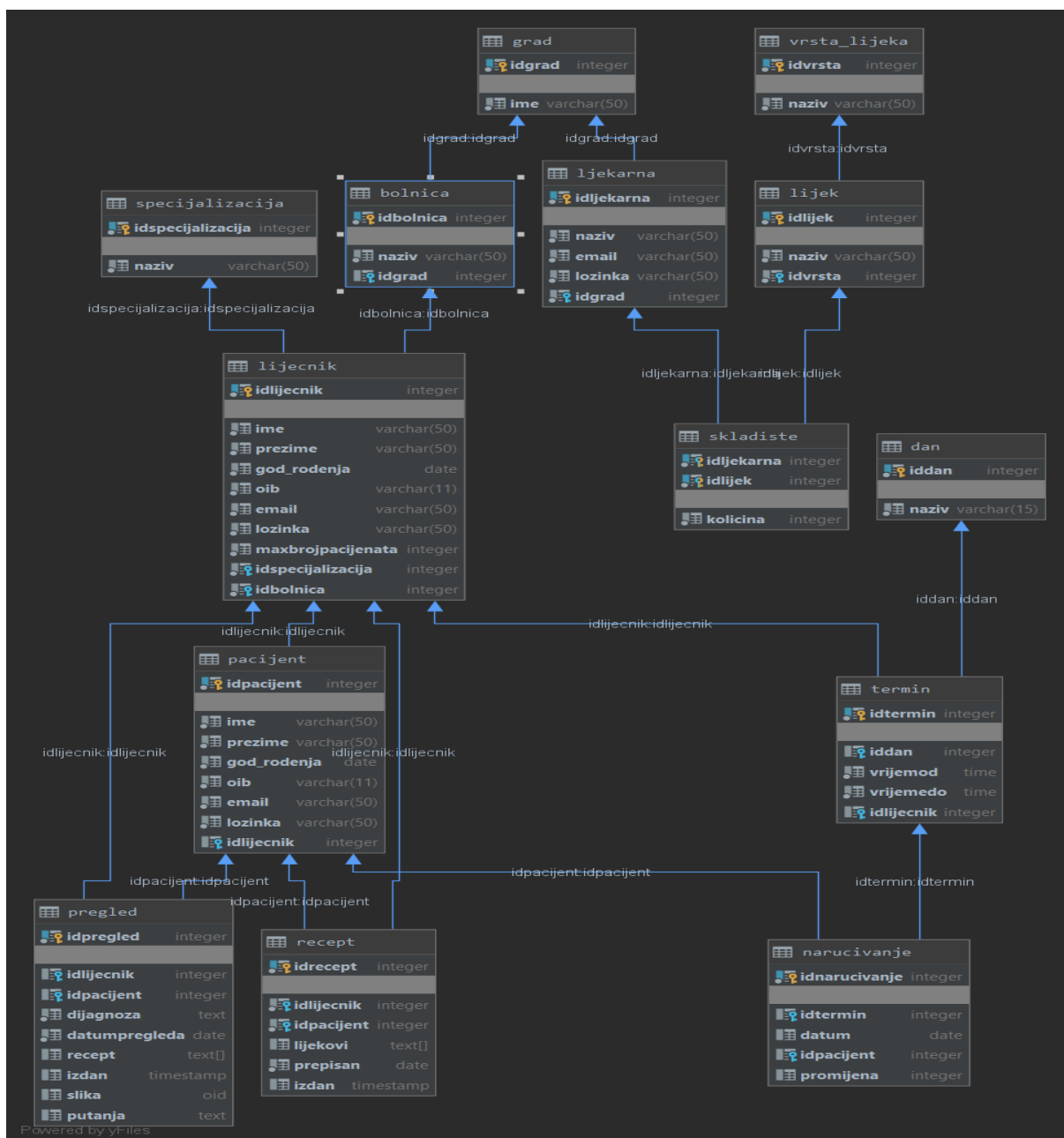
Osim aktivnih potrebno je bilo koristiti i temporalne baze podataka. Temporalne baze podataka za razliku od drugih baza podataka kao što su aktivne i poopćene, temporalna baza podataka koristi podatke koji su vremenske dimenzije. Što znači ukoliko je neki datum zapisan u tablicu, a tablica nije dugo ažurirana taj podatak može postati lažna te nepotrebno zauzima memoriju. Vremenske tipove koje podržava PostgreSQL DATE (datum), TIME (vrijeme), TIMETZ (vrijeme sa vremenskom zonom), TIMESTAMP (datum i vrijeme kao jedna varijabla) i TIMESTAMPTZ (isto kao TIMESTAMP uz dodatak vremenske zone) na taj način se u bazu pohranjuju podaci vremenskog tipa. Što u pravilu čini temporalnu bazu. [1]

2.3. Poopćene baza podataka

Poopćena baza podataka je baza koja omogućuje pohranjivanje velikih objekata unutar jedne relacije. Poopćene baze podataka koriste se kako bi se izbjegle operacije spoja nad tablicama jer su u današnje vrijeme podaci sve veći. Kako bi se kompleksniji objekti mogli spremiti u bazu podataka PostgreSQL koristi BLOB (eng. Binary Large Object) koji se koristi za pohranu binarnih podataka kao što su slike, videa i dr. te za takve objekte koristi se OID tip podataka. Također za poopćivanje koriste se jednodimenzionalna i dvodimenzionalna polja, prebrojene vrijednosti i složeni tipovi.[2]

3. Model baze podataka

Model baze podataka sastoji se od 12 tablica koji su liječnik, pacijent, ljekarna, lijek, grad, skladište, bolnica, specijalizacija, dan, termin, narucivanje i pregled. U ovom modelu postoji veza veza više na više, više na jedan. Što se može vidjeti i na slici 2.



Slika 2: ERA (Izvor: Osoba izrada)

Na slici 2 se također mogu vidjeti i tipovi podataka pa tako treba primjetiti da tablica pregled sadrži atribut slika koji je OID tipa, dok je recept tipa polje koji je tekst što znači da ovaj model baze podataka je uključio poopćenu. Osim tih poopćenih tipova postoje temporalni TIMESTAMP i DATE također u tablici pregled.

3.1. Implementacija

U nastavku će biti prikazani naredbe za kreiranje nekih tablica i okidača te će okidači biti opisani. Najzanimljivija tablica je tablica pregled, u kojoj vidimo poopćene i temporalne tipove podataka.

```
create table pregled(  
    idpregled serial primary key,  
    idlijecnik int references lijecnik(idlijecnik),  
    idpacijent int references pacijent(idpacijent),  
    dijagnoza text not null,  
    datumpregleda date not null,  
    recept text[],  
    izdan timestamp,  
    slika oid  
);
```

Za unosenje podataka u ovu tablicu koristi se dodatne funkcije za sažimanje objekta, a to su:

- lo_import - za unos objektata,
- lo_export - za čitanje pohranjenih podataka
- lo_unlink - za brisanje objekata

U našem slučaju koristimo lo_import za unos objekta u tablicu.

```
INSERT INTO pregled(idlijecnik,idpacijent,dijagnoza,datumpregleda,recept,slika)  
VALUES  
(1,1,'Upala grla',now(),'{Lanzul:1,Rantol:1,Ortonol:1}',lo_import(' /xampp/apache/  
public.png' ));
```

U datum pregleda smo pohranili vrijeme tog dana kada je pregled kreiran s funkcijom now() koji vraća npr. "2021-08-22", u polje recept smo dodali 3 lijeka i kolicinu koju pacijent treba preuzeti u ljekarni, te na kraju lo_import su dodali putanju datoteke koja će biti spremljena u OID obliku. S tim definiranjem tablice postigli smo poopćenje i temporalnost baze podataka.

Kako baza ne bi ostala samo poopćena i temporalana nego i aktivna kreirani su okidači koji služe za spriječavanje nekih događaja, ali omogućavanje nekih novih. Kako nam zadatak nije bio definirat skladište i njihovu strukturu nabava lijekova se vrlo brzo odvija tako što je kreiran okidač koji služi kada razina lijeka padne ispod 10, okidač se aktivira i automatski naruči još 30 odnosno u tablicu se doda stari broj + 30 novih.

```
CREATE OR REPLACE FUNCTION narucivanjeProizvoda()  
RETURNS TRIGGER  
LANGUAGE PLPGSQL  
AS  
$$  
begin  
    if new.kolicina < 10 then  
update skladiste set kolicina=new.kolicina+30 where idlijelek=new.idlijelek and  
    idljekarna=new.idljekarna;
```



```

end if;
return new;

end;
$$
CREATE TRIGGER triggernarucivanjeProizvoda
  AFTER UPDATE OR INSERT
  ON skladiste
  FOR EACH ROW
  EXECUTE PROCEDURE narucivanjeProizvoda()

```

Okidač se kreira tako da se kreira okidač funkcija koja vraća okidač, onda u sebi sadrži pravilo koje se treba zadovoljiti kako bi se moglo nešto izvesti. Zatim je kreiran okidač koji se postavlja kada će se okinuti i poziva se okidač funkcija. Na ovom primjeru vidimo da je okidač postavljen da se okine kada se izvrši ažuriranje ili unos, a funkcija narucivanjeProizvoda() će povećati količinu tog lijeka na skladištu isključivao ako je manji od 10. Kako bi se spriječilo da se ne bi izdalo više nego što ima na skladištu kreirana je okidač funkcija koja se aktivira prije ažuriranja i unosa

```

CREATE OR REPLACE FUNCTION proizvodNaSkladistu()
  RETURNS TRIGGER
  LANGUAGE PLPGSQL
  AS
$$
begin
    if new.kolicina > 0 THEN
        return new;
    else
        return null;
    end if;
end;
$$

```

Koja dopušta da se ažuriranje ili unos izvrši ako je nova količina veća od 0, a kako znamo da je veća, to pročitamo iz upita koji se postavlja za tu tablicu.

Kako ne bi samo na skladištu ostali, kreirani su i okidači za doktora kojemu se spriječava unos radnog vremena koji se poklapa s drugim radnim vremenom toga dana.

```

CREATE OR REPLACE FUNCTION kontrolaTermina()
  RETURNS TRIGGER
  LANGUAGE PLPGSQL
  AS
$$
declare
  zapis record;
begin
    for zapis in select iddan, vrijeme, vrijeme, idlijecnik
    from termin where iddan=new.iddan and idlijecnik=new.idlijecnik
    loop
        if (zapis.vrijeme=new.vrijeme or zapis.vrijemedo=new.vrijemedo) then
            return null;
        end if;
    end loop;
end;

```

```

        end loop;
        return new;
end;
$$
CREATE TRIGGER triggerkontrolaTermina
    BEFORE UPDATE OR INSERT
    ON termin
    FOR EACH ROW
    EXECUTE PROCEDURE kontrolaTermina()

```

U ovom okidaču se koristi petlja koja prolazi kroz sve redove koji su prethodno zapisani u varijablu zapis i ona se uspoređuje s novim vrijednostima koje se pokušavaju unjeti. Ukoliko petlja naiđe na iste vrijednosti trigger funkcija vraća null vrijednost i željeni upit se ne izvršava. Jedan od zanimljivih je okidača je okidač koji sprječava unos ako je pacijent odabrao pregled toga dana.

```

CREATE OR REPLACE FUNCTION brojacTermina()
RETURNS TRIGGER
LANGUAGE PLPGSQL
AS
$$
declare
brojac integer;
begin
    SELECT COUNT(*) FROM narucivanje WHERE datum=new.datum and idpacijent=new.
        idpacijent into brojac;
    raise notice 'brojac %',brojac;
    if brojac<1 THEN
        return new;
        else return null;
    end if;
    return new;

return new;
end;
$$
CREATE TRIGGER triggerBrojacTermina
    before INSERT
    ON narucivanje
    FOR EACH ROW
    EXECUTE PROCEDURE brojacTermina()

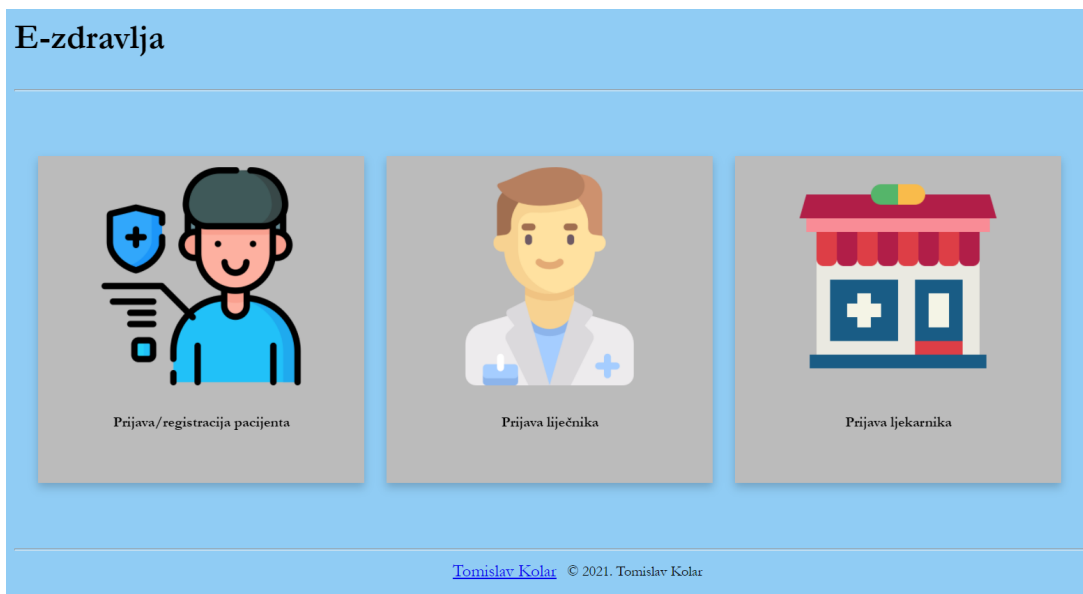
```

U ovom okidaču se prvo prebrojava koliko je idpacijent zapisan u tablici narucivanje i ako je zapisan već jednom okidač će blokirati nove upise za taj dan, sve dok ga korisnik ne obriše.

4. Primjeri korištenja

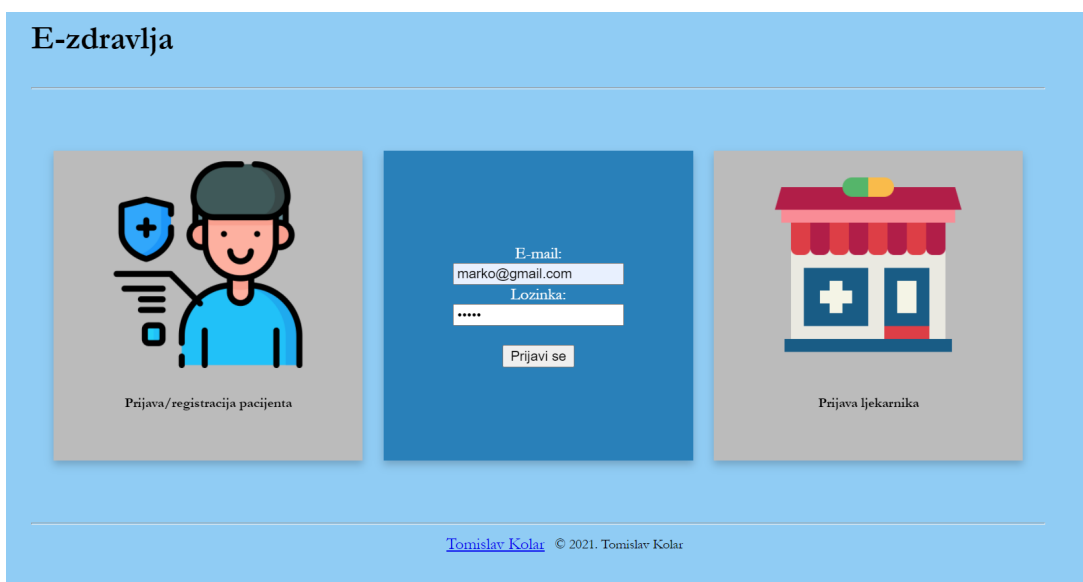
Kako bi se mogla pokretati ova Web aplikacija potrebno je imati server odnosno web poslužitelja, u ovom projektu korišten je xampp, a za hostanje baze koristimo pgAdmin4.

Na početku se potrebno logirati jer svaki uloga ima svoje mogućnosti



Slika 3: Početni zaslon (Izvor: Osoba izrada)

Prva interakcija baze i korisnika je uzimanje podataka korisnika.



Slika 4: Logiranje (Izvor: Osoba izrada)

Korisnik ima mogućnost odabira termina kod doktora određenog dana. Mogućnost pregleda bolesti i pregled recepata koji nisu podignuti. Kreiranje radnog tjedna doktor

Mogućnost pregleda nalaza od pojedinog pregleda ukoliko postoji.



E-zdravlje

Pacijent

Bjelovar

Marko Mamic | Obiteljska medicina

23 / 08 / 2021

Pregled termina

	PONEDJELJAK	UTORAK	SRIJEDA	ČETVRTAK	PETAK
7:00	07:00-08:00	07:00-08:00			
8:00	08:00-09:00	08:00-09:00			
9:00	09:00-10:00	09:00-10:00	09:00-08:00		
10:00	10:00-11:00	10:00-11:00	10:00-11:00		
11:00	11:00-12:00	11:00-12:00			11:00-12:00
12:00	12:00-13:00				12:00-13:00
13:00				13:00-14:00	13:00-14:00
14:00		14:00-15:00			
15:00					
16:00					
17:00					
18:00		18:00-19:00			

Slika 5: Zauzimanje termina (Izvor: Osoba izrada)



E-zdravlje

Pacijent

Odjava


Povijest bolesti

Recepti za podizanje

IME	PREZIME	BOLNICA	SPECIJALIZACIJA	DATUM	PREPISANI LIJEKOVI	NALAZ
Marko	Mamic	Opća bolnica Bjelovar	Obiteljska medicina	2021-08-23	Lanzul:3,Ranital:1,Ortanol:4	Nalaz

[Tomislav Kolar](#)
© 2021. Tomislav Kolar

Slika 6: Povijesti bolesti (Izvor: Osoba izrada)



E-zdravlje

Doktor

Odjava


Ažuriranje

Dodavanje

Brisanje

	PONEDJELJAK	UTORAK	SRJEDA	ČETVRTAK	PETAK
7:00	07:00-08:00	07:00-08:00			
8:00	08:00-09:00	08:00-09:00			
9:00	09:00-10:00	09:00-10:00	09:00-08:00		
10:00	10:00-11:00	10:00-11:00	10:00-11:00		
11:00	11:00-12:00	11:00-12:00			11:00-12:00
12:00	12:00-13:00				12:00-13:00
13:00				13:00-14:00	13:00-14:00
14:00		14:00-15:00			
15:00					
16:00					
17:00					
18:00		18:00-19:00			
19:00					

[illegible]



E-zdravlje
Doktore

Traži

DATUM PREGLEDA	PREPISANI LIJEKOVI	DIJAGNOZA	NALAZ
2021-08-23	Lanzulib,Ranitilid,Ortanolol	Dobro je	Nalaz

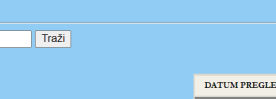
Tekst dijagnoze

+

No file chosen

[Tornislav Kolar](#) © 2021. Tornislav Kolar

Odljava



5. Zaključak

Tijekom izrade projekta uočio sam koliko su bitne temporalne, poopćene i aktivne baze. Koliko se uštedi vremena i prostora. Kao što je i u tekstu navedeno koje su prednosti svih triju baza, najviše pažnje bi ipak pridao aktivnim bazama jer za kreiranje okidača iako možda trivijalno neki izgledaju iz malih problema mogu nastati veliki, ali kad se naprave ispravno baza će biti dobro modelirana i smanjit će dodatne provjere u aplikacijama koje koriste te baze.

Popis literature

- [1] A. Perčinlić, „Temporalne baze podataka,” 2015. adresa: <https://www.cambridge.org>.
- [2] V. Galic, „Poučene relacijske baze podataka,” 2012. adresa: <https://urn.nsk.hr/urn:nbn:hr:211:023119>.

Popis slika

1.	Tijek rada aktivne baze (Izvor: https://www.drupal.org/files/drupalECA.png) . . .	2
2.	ERA (Izvor: Osoba izrada)	4
3.	Početni zaslon (Izvor: Osoba izrada)	8
4.	Logiranje (Izvor: Osoba izrada)	8
5.	Zauzimanje termina (Izvor: Osoba izrada)	9
6.	Povijesti bolesti (Izvor: Osoba izrada)	9
7.	Kreiranje randog tjedna (Izvor: Osoba izrada)	10
8.	Uvid u stanje pacijenta i dodavanje pregleda (Izvor: Osoba izrada)	10
9.	Pregled nalaza (Izvor: Osoba izrada)	10