

Modeliranje baze podataka za uslužne djelatnosti

Marko Domagoj Benković, Matija Fabek, Marija Gegić

1 Uvod

U sklopu ovog projekta napravili smo jednostavnu aplikaciju kojom pokazujemo moć i funkcionalnost Neo4j baze. Aplikacija je implementirana u C#-u i povezana s bazom podataka koju smo izgradili u Cypheru. Za generiranje nekih podataka koristili smo *mockaroo.com*. Podatke smo dodatno obradili ručno, a većinu smo kreirali pomoću Python skripti.

2 Opis zadatka

Uslužna djelatnost koju smo odabrali je hotelijerstvo. Odlučili smo izraditi aplikaciju koja će služiti za rezervacije smještaja u hotelima - poput servisa *booking.com*, *trivago.com* i sličnih. Želimo omogućiti korisnicima pretraživanje hotelskog smještaja u nekom gradu s obzirom na broj zvjezdica hotela, raspoloživost u nekom vremenskom periodu, udaljenost i cijenu. Aplikacija će također moći predložiti korisnicima koje opcije smještaja bi im se mogle svidjeti, s obzirom na to koji su smještaj rezervirali drugi korisnici iste dobne kategorije i spola.

Pri tome, smatramo da su klijenti odnosno korisnici aplikacije ljudi koji će pretraživati hotelski smještaj, pa onda omogućujemo korisnicima da se sami registriraju. Očekujemo da su svi podaci o hotelskoj ponudi već dostupni, kao i podaci o gradovima i sobama, te će se oni nalaziti u bazi pri pokretanju aplikacije. U bazi će se također nalaziti podaci o nekim klijentima i nekim prošlim rezervacijama, ali će aplikacija dopuštati spremanje novih klijenata i rezervacija u bazu.

3 Priprema podataka

Prije izrade same aplikacije, tražili smo najbolji način za generiranje i pre-processing podataka koje ćemo pohraniti u bazu kako bismo simulirali stvarne korisnike te način na koji ćemo organizirati bazu kako bismo mogli podržavati sve tražene funkcionalnosti.

Podatke o korisnicima smo generirali korištenjem stranice *mockaroo.com*. Kreirali smo 200 korisnika, uz podatke o imenu, dobi, spolu i lokaciji za svakog. Lokacije su nam ograničene na SAD pa smo s obzirom na to birali i lokacije hotela.

Koristeći istu stranicu, generirali smo skup američkih gradova i njihovih geografskih koordinata te ga suzili na 30 gradova koje smo smatrali popularnima. Za svaki od gradova smo ručno pronašli nekoliko hotela koji su bili ocijenjeni s 3 do 5 zvjezdica te smo kroz njihovu adresu izvukli njihove pripadne geografske koordinate.

Simulacije prethodnih rezervacija koje su korisnici radili smo generirali korištenjem Python skripte. Za svakog od korisnika koje smo kreirali, simulirali smo 10 rezervacija u razdoblju od 1.1.2019 do 15.6.2022. s nasumičnim duljinama trajanja rezervacije (do 10 dana), s podjednakim brojem rezervacija među svim hotelima.

Za sobe je bilo potrebno generirati cijene po kategorijama. Kao odrednicu smo koristili pronađenu informaciju o prosječnoj cijeni hotelskog noćenja iz 2020. godine, koju smo onda modificirali odnosno o broju zvjezdica hotela, broju kreveta i kategoriji.

```
df_hotels = pd.read_csv("hotels_v2.csv")
stars = df_hotels["stars"]

avg_daily_price = 103.25 # u dolarima - podatak iz 2020.godine

if __name__ == "__main__":
    df_rooms = pd.DataFrame()

    # u dolarima, +10 za half, +30 za full, +50 za all_inclusive
    # ovisi i o broju zvjezdica hotela
    if "half_board" not in df_hotels:
        df_hotels["half_board"] =
            np.round((stars-2)*10 + np.random.normal(0, 1),1)
    if "full_board" not in df_hotels:
        df_hotels["full_board"] =
            np.round((stars-2)*20 + np.random.normal(0, 2),1)
    if "all_inclusive" not in df_hotels:
        df_hotels["all_inclusive"] =
            np.round((stars-2)*30 + np.random.normal(0, 3),1)

    df_hotels.to_csv("hotels_v3.csv", index=False)

    for row in df_hotels.itertuples(): # prodi po retcima=hotelima
        city = row[1]
        hotel_name = row[2]
        star = row[3]
```

```

plus_minus = np.random.normal(0, 10) # centar je 0, stand. dev = 10
for i in range(10): # 10 soba za svaki hotel
    beds = np.random.normal(2, 0.5) # cca 1-4 kreveta, najcesce 2 i 3
    beds = int(np.ceil(beds))

    # avg_daily_price je pocetna cijena za kategoriju 3 zvijezdice
    # i +200 dolara za kategoriju vise
    # 1 krevet nema dodatne naknade, 2 kreveta=+30$, 3 kreveta=+60$ itd.
    # plus 30 dolara za dodatni krevet
    daily_price = avg_daily_price +
                  (star - 3) * 200 +
                  (beds - 1) * 30 + plus_minus

    room = {"city":city,
            "hotel_name": hotel_name,
            "beds": beds,
            "daily_price": np.round(daily_price,1),
            "room_number": i+1}
    df_rooms = df_rooms.append(room, ignore_index=True)

df_rooms.to_csv("rooms.csv", index=False)
)

```

Ovu fazu projekta smo nadograđivali paralelno s razradjivanjem plana za bazu podataka.

4 Baza podataka

Prema uputi u našem zadatku, za bazu podataka smo koristili Neo4j, koju smo onda povezali s C#-om. U bazi smo odlučili držati sve informacije o korisnicima, hotelima, sobama, rezervacijama, gradovima, ponudi koju hoteli imaju itd.

Odlučili smo koristiti 5 tipova čvorova: CLIENT, RESERVATION, HOTEL, ROOM i CITY. Oni su bili povezani s 5 tipova veza, na sljedeći način:

- (:Client) - [:RESERVED] → (:Reservation)
- (:Reservation) - [:IN] → (:Hotel)
- (:Reservation) - [:TAKES] → (:Room)
- (:Hotel) - [:OFFERS] → (:Room)
- (:Hotel) - [:SITUATED_IN] → (:City)

4.1 Čvorovi

4.1.1 Client

Čvor Client predstavlja korisnike koji su registrirani i mogu pretraživati hotelski smještaj. Za svakog klijenta pamtimo sljedeće atribute:

atribut	opis
firstName	string koji predstavlja ime
lastName	string koji predstavlja prezime
gender	M ili F, ovisno o spolu
dateOfBirth	datum rođenja, yyyy-mm-dd
placeOfBirth	mjesto rođenja, string
address	adresa stanovanja, string
pin	OIB, niz znamenki
email	email adresa korisnika, string
password	korisnička zaporka, string

Email adresa korisnika koristi se kao username kod logiranja u aplikaciju.

4.1.2 Reservation

Čvor rezervacija nam predstavlja ključne informacije o rezervaciji koju je neki klijent napravio. Povezana s klijentom, hotelom i sobom, daje sve informacije vezane uz klijentov posjet hotelu.

Za svaku rezervaciju pamtimo sljedeće atribute:

atribut	opis
checkIn	datum dolaska gosta, yyyy-mm-dd
checkOut	datum odlaska gosta, yyyy-mm-dd
guests	broj gostiju (1 ili više)
option	vrsta smještaja: all inclusive, full board, half board
rating	ocjena koju su korisnici dali smještaju, 1-5

4.1.3 City

Čvor City predstavlja sve gradove unutar kojih se nalaze hoteli koje imamo u bazi. Za svaki grad pamtimo sljedeće atribute:

atribut	opis
name	ime grada, string
lat	geografska širina središta grada, float
lon	geografska dužina središta grada, float

4.1.4 Hotel

Čvor Hotel predstavlja svaki od hotela koji se nalazi u bazi. Za svaki hotel pamtimo sljedeće atribute:

atribut	opis
name	ime hotela, string
stars	broj zvjezdica u kategorizaciji hotela, int
lat	geografska širina lokacije hotela, float
lon	geografska dužina lokacije hotela, float
halfBoard	cijena pripadne opcije, float
fullBoard	cijena pripadne opcije, float
allInclusive	cijena pripadne opcije, float

4.1.5 Room

Čvor Room predstavlja jednu individualnu sobu koja se nalazi u nekom hotelu. Moguće je da unutar istog hotela imamo više soba jednake cijene i kapaciteta, svaka od njih je numerirana i predstavljena zasebnim čvorom. Za svaku sobu pamtimo sljedeće atribute:

atribut	opis
beds	broj kreveta, int
dailyPrice	cijena sobe po danu, float
roomNumber	redni broj sobe u hotelu, int

4.2 Veze

Niti jedna veza navedena na početku odjeljka 4 nema vlastite atribute. Sve potrebne informacije nalaze se u samim čvorovima, a veze koristimo kako bismo predstavili njihovu interakciju.

Brid koji predstavlja vezu između klijenta i rezervacije je brid tipa RESERVED. Dakle, nakon potvrđene rezervacije, kreira se brid između klijenta i odabrane rezervacije. Za tu rezervaciju, može se saznati u kojem je hotelu te na koju sobu se odnosi. Naravno, jedan klijent može imati više rezervacija, ali ne mora imati niti jednu.

Veza između odabrane rezervacije i pripadnog hotela je veza tipa IN. Slično kao i kod veze tipa RESERVED, veza tipa IN nastaje između rezervacije i hotela kad klijent odabere željenu rezervaciju u određenom hotelu.

Svaki hotel ima određeni broj soba, stoga koristimo vezu tipa OFFERS kako

bismo ih povezali. Jasno, jedan hotel može imati više soba, ali svaka soba pripada točno jednom hotelu.

Kako bismo znali odrediti i ponuditi klijentu slobodnu sobu, koristimo vezu tipa FOR između rezervacije i sobe. Odnosno, ukoliko neka soba u hotelu nema vezu na rezervaciju, to znači da je tog trenutka slobodna te može ići u ponudu. Prilikom isteka rezervacije, uklanja se brid između pripradne rezervacije i zadužene sobe.

Posljedna veza koju koristimo je tipa SITUATED_IN. Ona spaja hotel sa gradom u kojem se nalazi. Služi nam za pronalazak hotela koji se nalaze u gradu kojeg klijent odabere u aplikaciji prilikom rezervacije smještaja. Slično kao i kod veze OFFERS, pojedini grad ima nekoliko hotela, ali svaki hotel pripada samo jednom gradu.

4.3 Razvijanje baze

Komentirat ćemo ukratko proces razvijanja baze. Prosli smo kroz više iteracija u razvitku, mijenjajući zamisljenu ideju baze ovisno o tome kako smo razrađivali ideju zadatka. Početno nam je bilo jasno da će baza sigurno morati sadržavati čvorove za klijente, rezervacije i hotele. Nakon što smo odlučili da ćemo podržavati mogućnost različitih kapaciteta soba, odlučili smo dodati čvor za sobe, te smo dodali i atribut cijene i vrste smještaja. Zatim smo odlučili da ćemo nuditi opciju pretraživanja hotela tako da korisnik primarno bira grad, te smo u tu svrhu dodali poseban čvor za gradove.

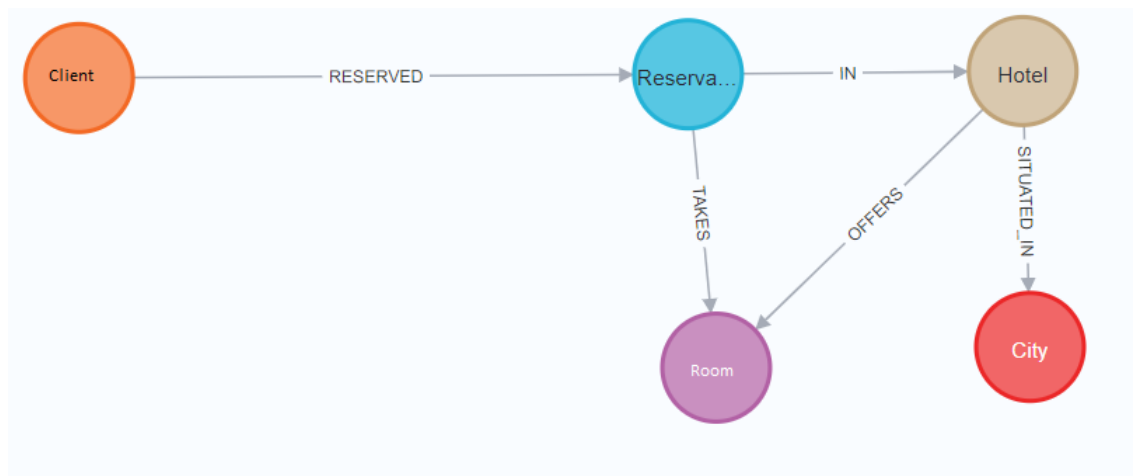
Posljednja faza u razvijanju baze sadržavala je dodavanje veze između soba i rezervacija. Shvatili smo da nam je ta veza potrebna kako bismo mogli pratiti koje su sobe zauzete/slobodne. Ova odluka nam je bila najkompliciraniji dio razvitka baze, jer smo razmatrali više opcija, uključujući opciju da ne numeriramo sobe, već da samo pamtimo koliko soba postoji u svakom hotelu, te da zasebnim upitima dobijemo postoji li slobodna soba tako da provjerimo broj aktivnih rezervacija. Naposljetku smo se odlučili za dodavanje veze između rezervacije i sobe, radi pojednostavljenja procesa.

4.4 Završno stanje

Donja slika prikazuje vizualizaciju baze koju koristimo, sa svim čvorovima i vezama.

4.5 Upiti za kreiranje baze

U ovom odlomku navodimo niz svih upita potrebnih za kreiranje baze, do stanja u kojem je očekujemo prije pokretanja aplikacije. Datoteke clients.csv, reservations.csv, cities.csv, hotels_V3.csv i rooms.csv smo prethodno kreirali korištenjem gore navedenih metoda.



Za pocetak, stvaramo čvorove za sve generirane korisnike.

```

LOAD CSV WITH HEADERS FROM 'file:///clients.csv' AS line
CREATE (:Client {
    pin: toInteger(line.pin),
    firstName: line.first_name,
    lastName: line.last_name,
    email: line.email,
    gender: line.gender,
    address: line.address,
    dateOfBirth: date(line.date_of_birth),
    placeOfBirth: line.place_of_birth,
    password: line.password
})
  
```

Zatim stvaramo čvorove za sve simulirane rezervacije.

```

LOAD CSV WITH HEADERS FROM 'file:///reservations.csv' AS line
CREATE (:Reservation {
    clientPin: toInteger(line.client_pin),
    checkIn: date(line.checkIn),
    checkOut: date(line.checkOut),
    guests: toInteger(line.guests),
    rating: toInteger(line.rating),
    option: line.option,
    hotelName: line.hotel_name
})
  
```

Zatim stvaramo veze između rezervacija i klijenata koji su ih napravili. Za to koristimo polje clientPin.

```
MATCH (c:Client),(r:Reservation) WHERE c.pin=r.clientPin
CREATE (c)-[:RESERVED]->(r);
```

Zatim stvaramo čvorove za sve gradove.

```
LOAD CSV WITH HEADERS FROM 'file:///cities.csv' AS line
CREATE (:City {
    name: line.city,
    lat: toFloat(line.lat),
    toFloat(lon: line.lon)
})
```

Zatim stvaramo čvorove za sve hotele.

```
LOAD CSV WITH HEADERS FROM 'file:///hotels_v3.csv' AS line
CREATE (:Hotel {
    city: line.city,
    name: line.name,
    stars: toInteger(line.stars),
    lat: toFloat(line.lat),
    lon: toFloat(line.lon),
    halfBoard: toFloat(line.half_board),
    fullBoard: toFloat(line.full_board),
    allInclusive: toFloat(line.all_inclusive)
})
```

Naposlijetku, stvaramo čvorove za sve sobe.

```
LOAD CSV WITH HEADERS FROM 'file:///rooms.csv' AS line
CREATE (:Room {
    city: line.city,
    hotelName: line.hotel_name,
    beds: toInteger(line.beds),
    dailyPrice: toFloat(line.daily_price),
    roomNumber: toInteger(line.room_number)
```



```
}  
)
```

Potrebno je povezati hotele i sobe, hotele i gradove te rezervacije i hotele.

```
MATCH (h:Hotel),(r:Room) WHERE h.city=r.city AND h.name=r.hotelName  
CREATE (h)-[:OFFERS]->(r);  
MATCH (h:Hotel),(c:City) WHERE h.city=c.name  
CREATE (h)-[:SITUATED_IN]->(c);  
MATCH (r:Reservation),(h:Hotel) WHERE r.hotelName=h.name  
CREATE (r)-[:IN]->(h);
```

Zatim uklanjamo određena svojstva čvorova koja nam više ne trebaju, a služila su nam za generiranje veza između čvorova.

```
MATCH (r:Reservation) REMOVE r.clientPin, r.hotelName;  
MATCH (h:Hotel) REMOVE h.city;  
MATCH (r:Room) REMOVE r.city, r.hotelName;
```

Nakon izvršavanja prethodnih naredbi, u bazi se nalaze svi podaci koje očekujemo prije pokretanja aplikacije.

5 Aplikacija

Za implementaciju aplikacije i njezinog grafičkog sučelja koristili smo C#. Za povezivanje aplikacije s bazom, koristili smo Neo4j biblioteku i Neo4j.Driver unutar nje.

Sljedeći isjecak koda prikazuje povezivanje s bazom:

```
private readonly IDriver driver;  
public DatabaseService()  
{  
    this.driver = GraphDatabase.Driver(  
        "bolt://localhost:7687",  
        AuthTokens.Basic("neo4j", "password")  
    );  
}  
)
```

Upite smo izvršavali unutar transakcija, koristenjem naredbe tx.Run. Sljedeći isjecak koda prikazuje primjer jednog upita kojim smo vršili provjeru passworda korisnika koji se pokušava logirati.

```

public bool Login(string username, string password)
{
    using (var session = driver.Session())
    {
        List<string> passList = session.ReadTransaction(tx =>
        {
            var result = tx.Run(" " +
                "MATCH (a: Client) " +
                "WHERE a.email = $username " +
                "RETURN a.password",
                new {username});
            List<string> passwords = new List<string>();
            foreach(var record in result)
            {
                passwords.Add(record["a.password"].ToString());
            }
            return passwords;
        });
        if (passList.Count != 1)
        {
            return false;
        }
        else
        {
            return passList[0] == password;
        }
    }
}
)


```

5.1 Funkcionalnost

Kroz ovo poglavlje ćemo kratko predstaviti funkcionalnost izrađene aplikacije.

Prilikom pokretanja aplikacije, prikazuje se prozor za Login. Korisnik može upisati svoj username i password, ako je već registriran, ili se registrirati, ako već nije. Izgled tog prozora prikazan je na sljedećoj slici.

Login



Uusername

Password

Exit

Login

Don't have an account?

Sign up

Ako se korisnik želi registrirati, klikom na odgovarajući gumb mu se otvara novi prozor unutar kojeg može unijeti svoje podatke. Od korisnika trazimo da upise svoje osnovne informacije, koje su nam kasnije potrebne za kreiranje rezervacije i za kreiranje preporuka. Izgled tog prozora prikazan je na sljedećoj slici.

Register

First name

Last name

Gender

▼

Email

Address

Date of birth

Thursday . June 16, 2022

▼

Place of birth

PIN

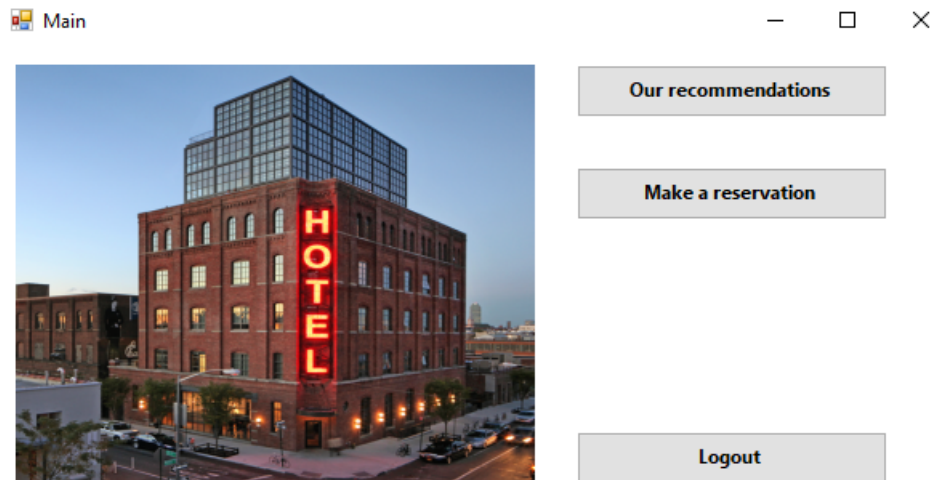
Password

Register

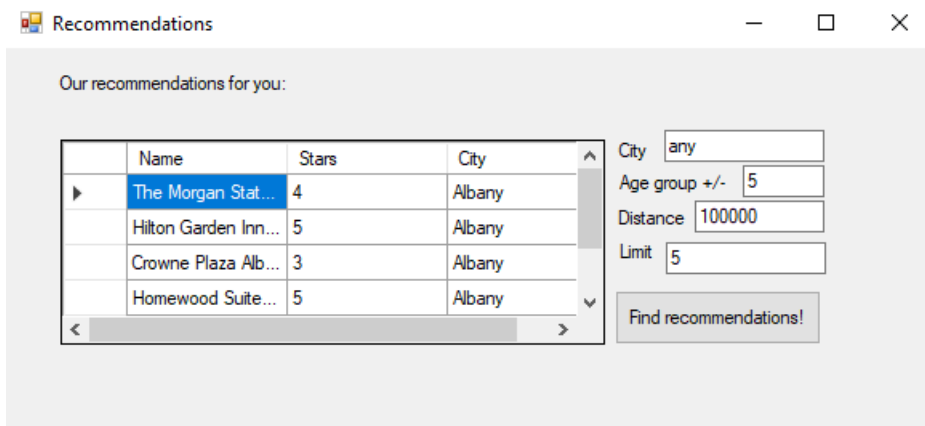
Kako bismo osigurali da su klijenti jedinstveni, prilikom registracije novih klijenata provjeravamo jesu li već registrirani.

```
public bool ValidateEmail(string email)
{
    using (var session = driver.Session())
    {
        bool emailExists = session.ReadTransaction(tx =>
        {
            var result = tx.Run("MATCH (c:Client) " +
                                "WHERE c.email = $email " +
                                "RETURN c.email;",
                                new { email });
            return result.Any();
        });
        if (emailExists) return false;
        return true;
    }
}
```

Ako se korisnik uspješno ulogirao ili registrirao, aplikacija ga vodi na glavni izbornik, gdje ima opciju pregledati preporuke ili napraviti novu rezervaciju.



Ako korisnik odabere pregled preporuka, otvorit će mu se novi prozor poput onog prikazanog na sljedećoj slici.



Unutar tog prozora se nalazi skup hotela kojeg predlazemo korisniku. Inicijalno se tu nalazi prijedlog 5 hotela iz cijele Amerike (odnosno među svim hotelima u bazi) koje su najčešće birali korisnici istog spola u istoj dobnoj kategoriji (do 5 godina stariji ili mlađi od korisnika koji vrši pretragu).

```
public List<Tuple<string, string, string>>
GetHotelRecommendationsForClient(string email, int ageDiff, int limit)
{
    using (var session = driver.Session())
    {
        List<Tuple<string, string, string>> hotels =
            session.ReadTransaction(tx =>
            {
                var result = tx.Run("MATCH(curr_client: Client), (destination: City), " +
                    "(client: Client) -[RESERVED] - (r: Reservation) " +
                    "-[IN] - (h: Hotel) -[SITUATED_IN] - (city: City)" +
                    "WHERE curr_client.email = $email " +
                    "AND client.gender = curr_client.gender " +
                    "AND client.dateOfBirth > " +
                    "    curr_client.dateOfBirth - Duration({ years: $ageDiff}) " +
                    "AND client.dateOfBirth < " +
                    "    curr_client.dateOfBirth + Duration({ years: $ageDiff}) " +
                    "RETURN h.name, h.stars, city.name, count(*) as cnt " +
                    "ORDER BY cnt DESC LIMIT $limit",
                    new { email, ageDiff, limit }
                );
            });

        List<Tuple<string, string, string>> fetchedHotels =
            new List<Tuple<string, string, string>>();

        foreach (var record in result)
```

```

    {
        Tuple<string, string, string> item =
            new Tuple<string, string, string>(
                record["h.name"].ToString(),
                record["h.stars"].ToString(),
                record["city.name"].ToString()
            );
        fetchedHotels.Add(item);
    }

    return fetchedHotels;
});
return hotels;
}
}
)

```

U slučajevima kad je korisnik zainteresiran za neki grad i područje oko njega, provjeravali smo i udaljenost do traženog grada.

```

public List<Tuple<string, string, string>>
GetHotelRecommendations(string email, string destination, int ageDiff,
int distance, int limit)
{
    using (var session = driver.Session())
    {
        List<Tuple<string, string, string>> hotels =
            session.ReadTransaction(tx =>
            {
                var result = tx.Run("MATCH(curr_client: Client), (destination: City), " +
                    "(client: Client) -[RESERVED] - (r: Reservation) " +
                    "-[IN] - (h: Hotel) -[SITUATED_IN] - (city: City) " +
                    "WHERE curr_client.email = $email " +
                    "AND destination.name = $destination " +
                    "AND client.dateOfBirth > " +
                    "curr_client.dateOfBirth - Duration({ years: $ageDiff}) " +
                    "AND client.dateOfBirth < " +
                    "curr_client.dateOfBirth + Duration({ years: $ageDiff}) " +
                    "WITH point.distance(" +
                    "point({ " +
                    "    longitude: destination.lon, " +
                    "    latitude: destination.lat, crs: 'WGS-84'})", " +
                    "point({ " +
                    "    longitude: h.lon, " +
                    "    latitude: h.lat, " +
                    "    crs: 'WGS-84'})")) " +
                    " as dist, h as h, city as city " +

```

```

        "WHERE dist < $distance " +
        "RETURN h.name, count(*) as cnt, h.stars, city.name " +
        "ORDER BY cnt DESC LIMIT $limit",
        new { email, destination, ageDiff, distance, limit }
    );


    List<Tuple<string, string, string>> fetchedHotels =
        new List<Tuple<string, string, string>>();

    foreach (var record in result)
    {
        Tuple<string, string, string> item =
            new Tuple<string, string, string>(
                record["h.name"].ToString(),
                record["h.stars"].ToString(),
                record["city.name"].ToString());
        fetchedHotels.Add(item);
    }

    return fetchedHotels;
});
return hotels;
}
}
)

```

Kada korisnik želi napraviti novu rezervaciju, klikom na odgovarajući gumb otvorit će mu se prozor za rezervacije.

 Reservation

 — □ ✕

City

Date from

Thursday . June 1 ▾

Date until

Thursday . June 1 ▾

Number of people

1 ▴ ▾

Search

Clear

Submit

Korisnik zatim može unijeti naziv grada, datume i broj osoba za koje želi napraviti rezervaciju. Klikom na gumb za pretragu, dobit će rezultate koji odgovaraju kriterijima pretrage.

U nastavku slijedi dio koda koji koristimo za pretragu.

```
public List<Hotel> GetHotels(string city, DateTime dateFrom, DateTime dateUntil,
int personNumber)
{
    using (var session = driver.Session())
    {
        List<Hotel> hotels = session.ReadTransaction(tx =>
        {
            var result = tx.Run("MATCH (h:Hotel)-[:OFFERS]->(r:Room), (z:Reservation) " +
                "WHERE h.city = $city " +
                "AND r.beds >= $personNumber " +
                "AND (" +
                "(NOT (r)-[:TAKE]->()) " +
                "OR " +
                "((r)-[:TAKE]->(z) " +
                "AND (z.checkIn > $dateUntil OR z.checkOut < $dateFrom))" +
                ") " +
                "RETURN h.name, h.stars, h.half_board, h.full_board, h.all_inclusive",
                new{city, dateFrom, dateUntil, personNumber });

            List<Hotel> fetchedHotels = new List<Hotel>();

            foreach (var record in result)
            {
                fetchedHotels.Add(new Hotel()
```



```

    {
        name = record["h.name"].ToString(),
        stars = record["h.stars"].ToString(),
        halfBoard = record["h.half_board"].ToString(),
        fullBoard = record["h.full_board"].ToString(),
        allInclusive = record["h.all_inclusive"].ToString(),
    });
}
return fetchedHotels;
});
return hotels;
}
}
)

```

Kada korisnik odabere smještaj, stvaramo i novu rezervaciju u bazi i spajamo ju sa sobom.

Posljedično moramo brisati i vezu sa sobom nakon sto istekne rezervacija, kako bismo mogli označiti da je ta soba slobodna.

```

public void CheckExpiredReservations(string today) {
    using (var session = driver.Session())
    {
        var result = session.WriteTransaction(tx =>
        {
            var query_result = tx.Run("" +
                "MATCH (res:Reservation)-[t:TAKES]->(room:Room) " +
                "WHERE res.checkOut < date($today) " +
                "DELETE t;",
                new { today }
            );

            return query_result;
        });
        return;
    }
}
)

```

6 Zaključak

Izradili smo jednostavnu aplikaciju za pretragu hotelskog smještaja. Izradili smo bazu podataka te ju napunili umjetnim podacima koje smo samostalno generirali i obradili. Aplikacija nudi mogućnost registracije korisnika, pretrage

hotela, rezervacije hotela, pregleda preporuka hotela za korisnika. Izradili smo graficko sucelje, povezali aplikaciju s bazom te izložili prikaz rada te aplikacije u ovom seminaru.

Ovo nam je bio prvi susret s ovakvom izradom aplikacije koja je uključivala i osmišljavanje baze i povezivanje baze s aplikacijom, te smo naučili nove stvari vezane uz kreiranje, planiranje i razvitak baze, i vezane uz samo planiranje projekta i project management ovakvog rada - jer nam je oblik baze definirao kako možemo implementirati aplikaciju i koje mogućnosti možemo podržavati.