Model podataka

TravelPlan

Članovi tima:

Petar Trifunović 16955
Maša Nešić 16774

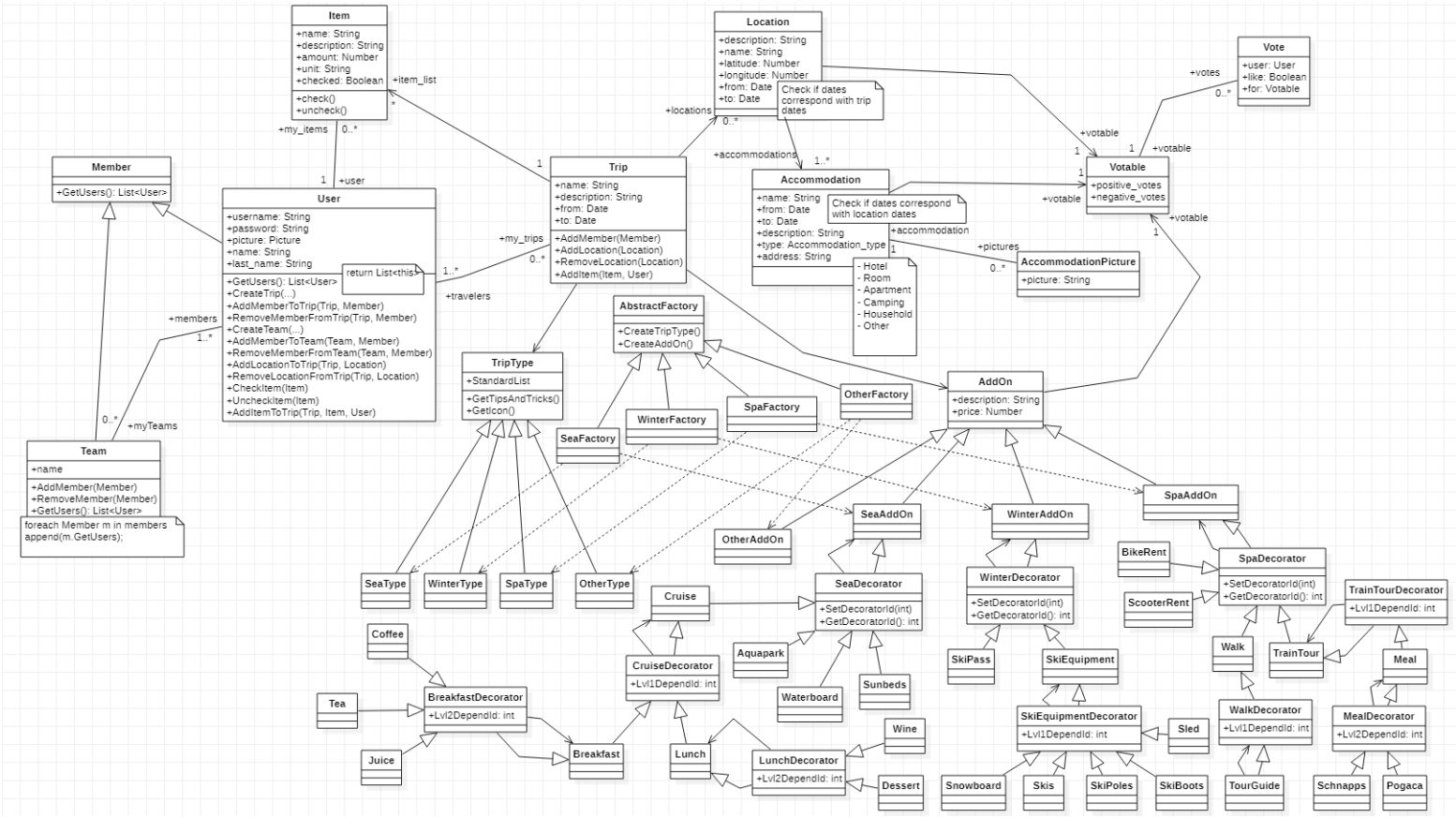
Naziv tima:

Muffin Time

Sadržaj

1. Model podataka

Model podataka *TravelPlan* sistema prikazan je na klasnom dijagramu koji sledi.



Klasni dijagram
(./Class_diagram.png)

U nastavku će ukratko biti opisana svaka od klasa.

1.1. User

User je klasa koja sadrži podatke o korisniku. Čuva njegovo korisničko ime (*username),* koje je jedinstveno za ceo sistem, šifru, ime, prezime i sliku. Korisnik ne može menjati svoje korisničko ime nakon registrovanja, može menjati ime, prezime i sliku. Može promeniti i šifru samo uz unošenje prethodne validne šifre. Klasa *User* takođe sadrži i listu timova kojima korisnik pripada, listu *item-a* za koje je korisnik zadužen na svim svojim putovanjima i listu putovanja čiji je deo. Implementira interfejs *Member,* koji omogućuje ravnopravno tretiranje korisnika i tima pri dodavanju (u tim ili putovanje).

1.2. Team

Team je klasa koja čuva informacije o jednom timu – njegovo ime i sve korisnike koji mu pripadaju. Nakon što korisnik kreira tim, on i postaje njegov jedini član. Naknadno u tim može dodati druge korisnike ili druge timove, pri čemu će se svi korisnici dodatih timova koji već nisu članovi tekućeg tima, dodati kao članovi. Implementira interfejs *Member,* koji omogućuje ravnopravno tretiranje korisnika i tima pri dodavanju (u tim ili

putovanje).

1.3. Member

Member predstavlja interfejs koji omogućuje uniformno posmatranje objekata klase *User* i kompozitnih objekata klase *Team.* Ovaj interfejs po *Composite* projektnom obrascu omogućuje da se prilikom dodavanja u putovanje ili tim na isti način izvršava funkcionalnost, nezavisno od toga da li se dodaje jedan korisnik ili grupa korisnika (tim).

1.4. Trip

Trip predstavlja klasu koja opisuje jedno putovanje. Putovanje kreira korisnik, i u početku je taj korisnik i jedini putnik. Naknadno mogu biti dodati drugi korisnici ili timovi u putovanje. Korisnici takođe mogu napustiti putovanje, pri čemu će putovanje biti obrisano onda kada ga svi putnici napuste. Putovanje od osnovnih informacija ima naziv, opis i datume kada počinje i kada se završava. U svakom trenutku postojanja putovanja mora biti zadovoljen uslov da je datum kraja isti ili kasniji od datuma početka putovanja. Putovanje ima skup korisnika koji na njemu učestvuju, ima skup *Item-a* koji su za njega vezani i skup lokacija koje obuhvata. Pri kreiranju putovanja, ono nema lokacije i *item-e,* tako da oni mogu biti dodati naknadno od strane bilo kod putnika koji učestvuje u tom putovanju. Svako putovanje ima svoj tip koji mora biti naveden prilikom kreiranja i ostaje fiksan tokom života putovanja, i u zavisnosti od tipa, putovanja može imati različite dodatke koji ga opisuju po *Decorator* projektnom obrascu. Kako bi se obezbedilo da se kreira odgovarajući tip putovanja i odgovarajući deo hijerarhije dekorisanja, koristi se *Abstract fastory* projektni obrazac.

1.5. Item

Item predstavlja klasu koja čuva podatke o stavci koja pripada jednom putovanju i za koju je zadužen jedan korisnik koji na njemu putuje. Stavci se može pristupiti kroz kolekciju stavki koje sadrži jedno putovanje ili kroz kolekciju stavki za koje je zadužen jedan korisnik. Jedna stavka ima ime, opis, količinu, jedinicu u kojoj se izražava ta količina i oznaku da li je ta stavka čekirana (obezbeđena) ili ne. Putovanje za koje je stavka vezana je fiksno i navodi se prilikom kreiranja, dok je u toku postovanja stavke moguće da ona menja korisnika koji je za nju zadužen. Moguće je menjati i ostale parametre – količinu, jedinicu, čekiranost.

1.6. Location

Location predstavlja klasu čije je postojanje vezano za putovanje i koja opisuje jednu od lokacija koje putovanje posećuje. Prilikom kreiranja lokacije navodi se kom putovanju pripada i ta veza ostaje nepromenljiva tokom života objekta lokacije. Lokacija ima naziv, opis, latitudu i longitudu, koje opisuju poziciju te lokacije u Ekliptičkom koordinatnom sistemu i datume od kada i do kada se planira boravak na toj lokaciji. Prilikom kreiranja lokacije i svake izmene njenih datuma mora biti zadovoljen uslov da je datum kraja isti ili kasniji od datuma početka boravka na lokaciji kao i da je period boravka na toj lokaciji deo perioda boravka na čitavom putovanju. Jedna lokacija može imati više različitih smeštaja.

1.7. Accommodation

Accommodation predstavlja klasu koja opisuje smeštaj. Od osnovnih parametara ima naziv, opis, adresu na kojoj se nalazi i tip smeštaja koji se bira iz ponuđenog skupa – hotel, soba, apartman, kampovnaje, domaćinstvo ili ostalo. Svaki objekat smeštaja vezan je za jednu lokaciju. Smeštaj takođe čuva informacije o tome od kada do kada se planira boravak u toj smeštajnoj jedinici. Validnost ovih datuma samostalno i u vezi sa datumima lokacije za koju je vezan smeštaj proverava se na sličan način kao provera datuma u odnosu lokacija – putovanje. Smeštaj takođe može imati i više slika koje ga opisuju pa sadrži i kolekciju objekata tipa *AccommodationPicture.*

1.8. AccommodationPicture

AccommodationPicture predstavlja klasu koja služi da čuva informacije o slici nekog smeštaja. Vezana je za 1 objekat tipa *Accommodation*. Postojanje ove klase uslovljeno je tehnologijama koje se koriste za realizaciju sistema. Kako bi se omogućilo korišćenje *Entity Framework ORM-a* uz *code-first approach,* nije moguće da entitet ima kolekciju podataka primitivnog tipa (u ovom slučaju, stringova koji bi opisivali slike), već je neophodno da se svaka slika predstavi zasebnim objektom, kako bi se isti mogao perzistirati.

1.9. TripType hijerarhija

Svako putovanje ima svoj tip, ovaj tip ostaje fiksan tokom čitavog postojanja objekta putovanja i opisan je hijerarhijom klasa *TripType.* Sama *TripType* klasa predstavlja apstraktnu klasu koja ima vezu sa putovanjem i definiše apstraktne, virtuelne funkcije koje će biti predefinisane u konkretnim izvedenim klasama, a služe za pribavljanje ikonice koja odgovara tipu putovanja i spiska saveta koji mogu biti korisni za odgovarajući tip putovanja. Ova klasa takođe sadrži i parametar koji čuva listu za pakovanje koja može biti od pomoći putnicima. Ova lista će biti napunjena predefinisanim vrednostima, karakterističnim za svaki tip putovanja u odgovarajućoj podklasi. Korisnici koji su deo nekog putovanja imaju mogućnost da tu predefinisanu listu prošire nekim dodatnim stavkama koje smatraju za korisne, te stavke će onda biti vidljive svim ostalim korisnicima koji su deo tog putovanja. Svaka od konkretnih podklasa – *SeaType, WinterType i SpaType* generiše početnu vrednost liste za pakovanje i predefiniše pomenute apstraktne funkcije tako da se u jednoj vraća spisak saveta vezanih za dati tip putovanja, a u drugoj se vrši učitavanje odgovarajuće ikonice i njena konverzija u *base64,* ukoliko nije već pristupano datoj ikonici, ili se već konvertovana ikonica samo vrati, ukoliko jeste.

1.10. AddOn hijerarhija

AddOn hijerarhija omogućuje postavljanje dodataka za neko putovanje, odnosno dekorisanje tog putovanja po principu *Decorator* projektnog obrasca. Putovanje ima referencu na jedan objekat ove hijerarhije, dok se ostali na dalje lančaju po principu bliskom lančanoj listi. Objekat na koji putovanje ima referencu u početku predstavlja objekat direktno izveden iz klase *AddOn* koja je apstraktna. Tim prvim nivoom klasa – *SeaAddOn, WinterAddOn, SpaAddOn, OtherAddOn,* obezbeđuje se da se ograniči korišćenje adekvatne podhijerarhije dekoratora, zavisno od tipa putovanja, dok sami objekti ovih klasa ne predstavljaju nikakve konkretne dodatke na putovanje. Iz svake od ovih klasa izvedena je po jedna *Decorator* klasa koja obezbeđuje mogućnost lančanja dekoratora istog tipa nad jednim putovanjem. *Decorator* klase najvišeg nivoa - *SeaDecorator, SpaDecorator i WinterDecorator* obezbeđuju referencu ka bilo kojoj instanci *AddOn* klase odgovarajućeg tipa. Ova referenca biće dalje predefinisana u dekorator hijerarhiji tako da klase koje se nalaze dublje imaju striktnija ograničenja povodom toga na šta treba da ukazuje ta veza. Ispod ovih *Decorator* klasa nalaze se dekoratori prvog nivoa, ovo su dekoratori koji ne zavise od drugih instanci i mogu se proizvoljno dodati na putovanje odgovarajućeg tipa, klase koje pripadaju ovom tipu su – *Cruise, Aquapark, Sunbeds, Waterboard, SkiPass, SkiEquipment, BikeRent, ScooterRent, Walk i TrainTour.* Neke od ovih klasa mogu dalje biti dekorisane dodatnim detaljima. Iz ovih klasa izvedene su *Decoator* klase drugog nivoa – *CruiseDecorator, SkiEquipmentDecorator, WalkDecorator i TrainTourDecorator.* Pri čemu svaka od ovih klasa uvodi ograničenje o tome na koji tip objekta treba da ukazuje referenca za lančanje (to je tip objekta iz kojeg je izvedena ova *Decorator* klasa). Ono što dodatno uvode klase ovog nivoa jesu veze zavisnosti prvog nivoa. Pored reference koja predstavlja *next* polje u lančanoj listi dodataka, ove klase zavise i od jednog od dekoratora prvog nivoa i bez njega ne mogu da postoje, tako da čuvaju i referencu (id) na objekat, odnosno dekorator prvog nivoa, od kojeg zavise. Iz ovih *Decorator* klasa sada se izvode konkretni dekoratori drugog nivoa – *Breakfast, Lunch, Snowboard, Skis, SkiPoles, SkiBoots, Sled, TourGuide, Meal.* U čitavoj ovoj hijerarhiji *Decorator* klase su uvek apstraktne, one unose neka nova ograničenja, ili neke nove parametre, dok se iz njih izove konkretne klase koje zapravo mogu biti instancirane. Neki od dekoratora drugog nivoa mogu biti i dalje dekorisani, zato se iz njih izvode *Decorator* klase trećeg nivoa – *BreakfastDecorator, LunchDecorator i MealDecorator.* Ove klase ponove uvode još striktnija ograničenja povodom toga kog tipa može biti *next* referenca, ali uvode i još jednu referencu jer sada ovi dekoratori, trećeg nivoa, ne zavise samo od dekoratora prvog, nego i od dekoratora drugog nivoa, pa se mora pamtiti i ova zavisnost. Iz *Decorator* klasa trećeg nivoa izvode se i konkretni dekoratori trećeg nivoa – *Coffee, Tea, Juice, Wine, Dessert, Schnapps i Pogaca.* Ovakvom organizacijom klasa omogućeno je ulančavanje dozvoljenih dekoratora na

adekvatan način i pamćenje svih vrsta zavisnosi dekoratora višeg nivoa od dekoratora nižeg nivoa.

1.11. AbstractFactory hijerarhija

Ova hijerarhija klasa obezbeđuje implementaciju *Abstract Factory* projektnog obrasca. Osnovna klasa hijerarhije je apstraktna i definiše apstraktne funkcije za kreiranje objekta tipa *AddOn* i objekta tipa *TripType.* Iz ove klase izvode se konkretne klase, pri čemu postoji po jedna za svaki mogući tip putovanja – *SeaFactory, SpaFactory, WinterFactory i OtherFactory.* Ove klase predefinišu apstraktne funkcije tako da kreiraju i dodatak i tip putovanja odgovarajućeg tipa.

1.12. Votable

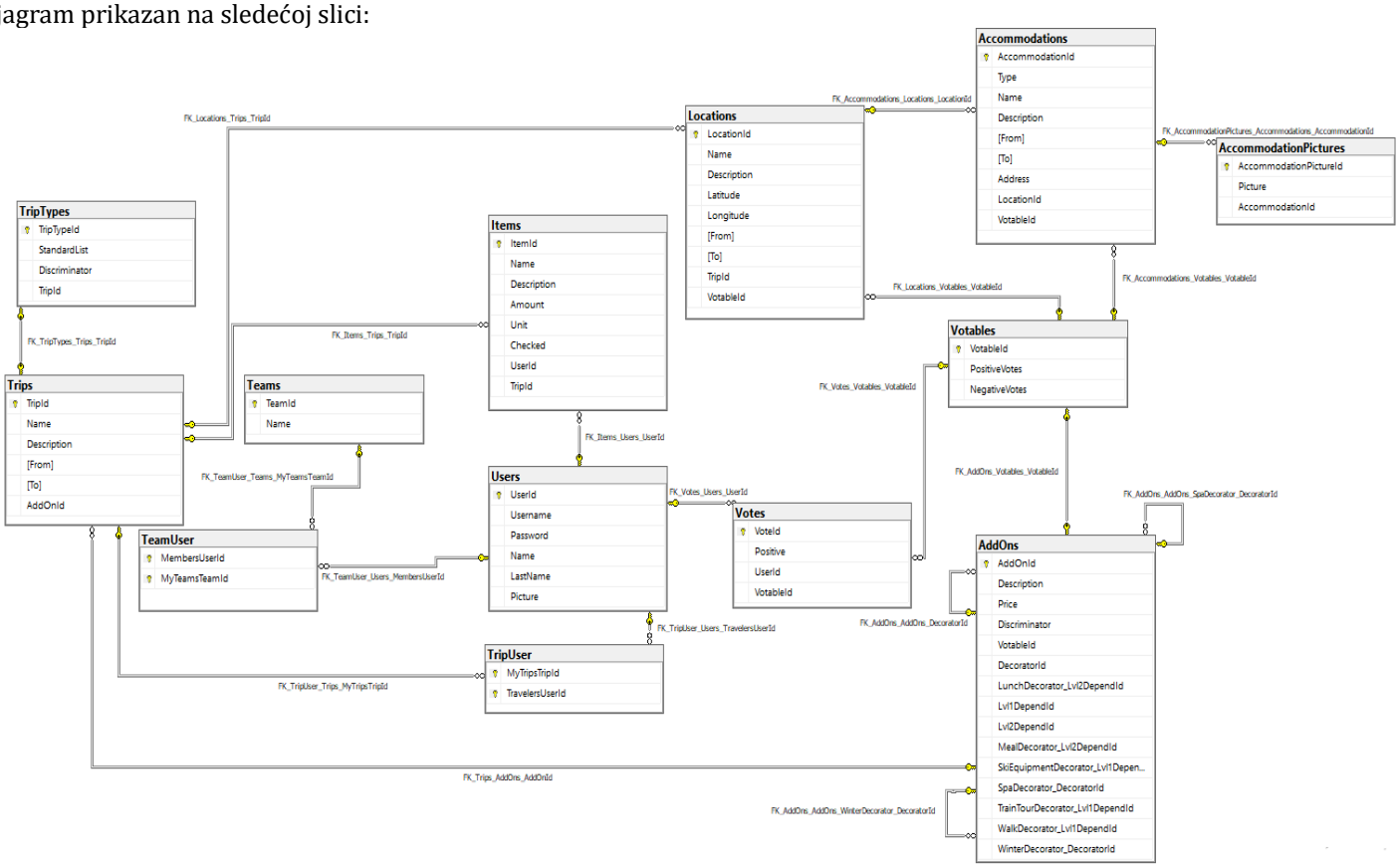
Za objekte tipa *Location, Accommodation* i bilo koji objekat iz *AddOn* hijerarhije je moguće glasati. Svaki korisnik, kojeg se taj objekat tiče, može da da jedan glas za njega koji je pozitivan ili negativan. Kako bi se omogućilo uniformno glasanje za objekte svih različitih tipova, svaki od njih ima referencu na objekat tipa *Votable* koji pruža usluge glasanja i čuva informacije o svim glasovima i o broju pozitivnih i negativnih glasova za objekat za koji je zadužen. Objekti ove klase imaju niz objekata klase *Vote* u kojem se pamte svi glasovi za dati objekat, a kako se sam broj pozitivnih i negativnih glasova vrlo često pribavlja, oni se čuvaju u samom objektu klase *Votalbe* kako bi bili lako dostupni.

1.13. Vote

Klasa *Vote* obezbeđuje čuvanje informacija o jednom glasu. Pamti da li je taj glas pozitivan ili ne, koji je korisnik glasao i za koji to objekat.

2. Model entiteta

Opisan model podataka ima odgovarajuću reprezentaciju u bazi podataka, u vidu modela entiteta, čiji je dijagram prikazan na sledećoj slici:



Osnovne klase (*User, Team, Trip, Item, Location, Accommodation, AccommodationPicture, Votable i Vote)* imaju odgovarajuće tabele u bazi, a reference koje postoje između ovih klasa predstavljene su odgovarajućim stranim ključevima. *TeamUser* i *TripUser* tabele generisane su radi predstavljanja *many-to-many* veza između korisnika i timova, i korisnika i putovanja.

Čitava *AddOn* hijerarhija, odnosno sve klase koje je čine, predstavljene su jednom tabelom (*AddOns* tabela). Atribut *Discriminator* je pomoćni atribut uveden od strane same baze, i nosi informaciju o tome na koju konkretnu klasu iz *AddOn* hijerarhije se odnosi određena vrsta. Atributi koji u nazivu sadrže *DecoratorId* se koriste kako bi predstavili lančanu strukturu ove hijerarhije, opisanu u odeljku **1.10 *AddOn* hijerarhija** (atribut čiji je naziv upravo *DecoratorId* se odnosi na granu hijerarhije vezanu za *SeaAddOn* klasu, a iz naziva druga dva *DecoratorId* atributa se zaključuje na koje se grane odnose). Jedan ovakav atribut jedne vrste u *AddOns* tabeli predstavlja strani ključ ka drugoj vrsti iste tabele. Na sam početak ove lančane strukture ukazuje atribut *AddOnId* u tabeli *Trips*. Atributi jedne vrste tabele *AddOns* koji u nazivu sadrže *Lvl1DependId* ukazuju na onu *AddOn* jedinku nivoa jedan koju ta vrsta dekoriše. Po istom principu, atributi koji u nazivu sadrže *Lvl2DependId* se odnose na *AddOn*-ove nivoa dva. Na ovaj način, entitetima u bazi podataka predstavljene su instance i dekoratori hijerarhije *AddOn*.

3. Mehanizam mapiranja

3.1. Mapiranje između entiteta baze podataka i objekata entitetskih klasa

TravelPlan projekat koristi SQL Server bazu za skladištenje entiteta i perzistenciju stanja sistema. Za mapiranje između objekata entitetskih klasa i samih entiteta baze podataka korišćen je Entity Framework Core objektno-relacioni maper. Pristup mapiranju je *code-first*, tako da su tabele relacione baze automatski kreirane na osnovu POCO klasa kojima je prethodno uspostavljen već opisani model podataka.

3.2. Mapiranje između objekata entitetskih klasa i Data-transfer objekata

Komunikacija serverske i klijentske strane aplikacije omogućena je slanjem *Data-transfer objekata* (DTO) između ove dve strane, putem RESTful API-ja. DTO klase su uvedene kako bi se klijentu omogućio uvid u tačno određene podatke iz čitavog skupa podataka koje entitetske klase sadrže. Takođe, ove DTO klase uređuju podatke na način koji odgovara konkretnim API pozivima, kako bi se smanjilo slanje redundatnih podataka prilikom komunikacije. Za mapiranje entitetskih u Data-transfer objekte, korišćen je .NET paket *AutoMapper*.