*Model podataka*

**TravelPlan**

*Članovi tima: Naziv tima:*

### Petar Trifunović 16955 Muffin Time

### Maša Nešić 16774

***Sadržaj***

1. [**Model podataka**](#_heading=h.gjdgxs) **3**

1.1. [*User*](#_heading=h.1fob9te) 3

1.2. [*Team*](#_heading=h.1fob9te) 3

1.3.  [*Member*](#_heading=h.1fob9te) 4

1.4.  [*Trip*](#_heading=h.1fob9te) 4

1.5.  [*Item*](#_heading=h.1fob9te) 4

1.6.  [*Location*](#_heading=h.1fob9te) 4

1.7.  [*Accommodation*](#_heading=h.1fob9te) 4

1.8.  [*AccommodationPicture*](#_heading=h.1fob9te) 4

1.9.  [*TripType* hijerarhija](#_heading=h.1fob9te) 5

1.9.1  [*SeaType*](#_heading=h.1fob9te) 5

1.9.2  [*WinterType*](#_heading=h.1fob9te) 5

1.9.3 [*SpaType*](#_heading=h.1fob9te) 5

1.9.4 [*OtherType*](#_heading=h.1fob9te) 5

1.10.  [*AddOn* hijerarhija](#_heading=h.1fob9te) 5

1.11.  [*AbstractFactory* hijerarhija](#_heading=h.1fob9te) 5

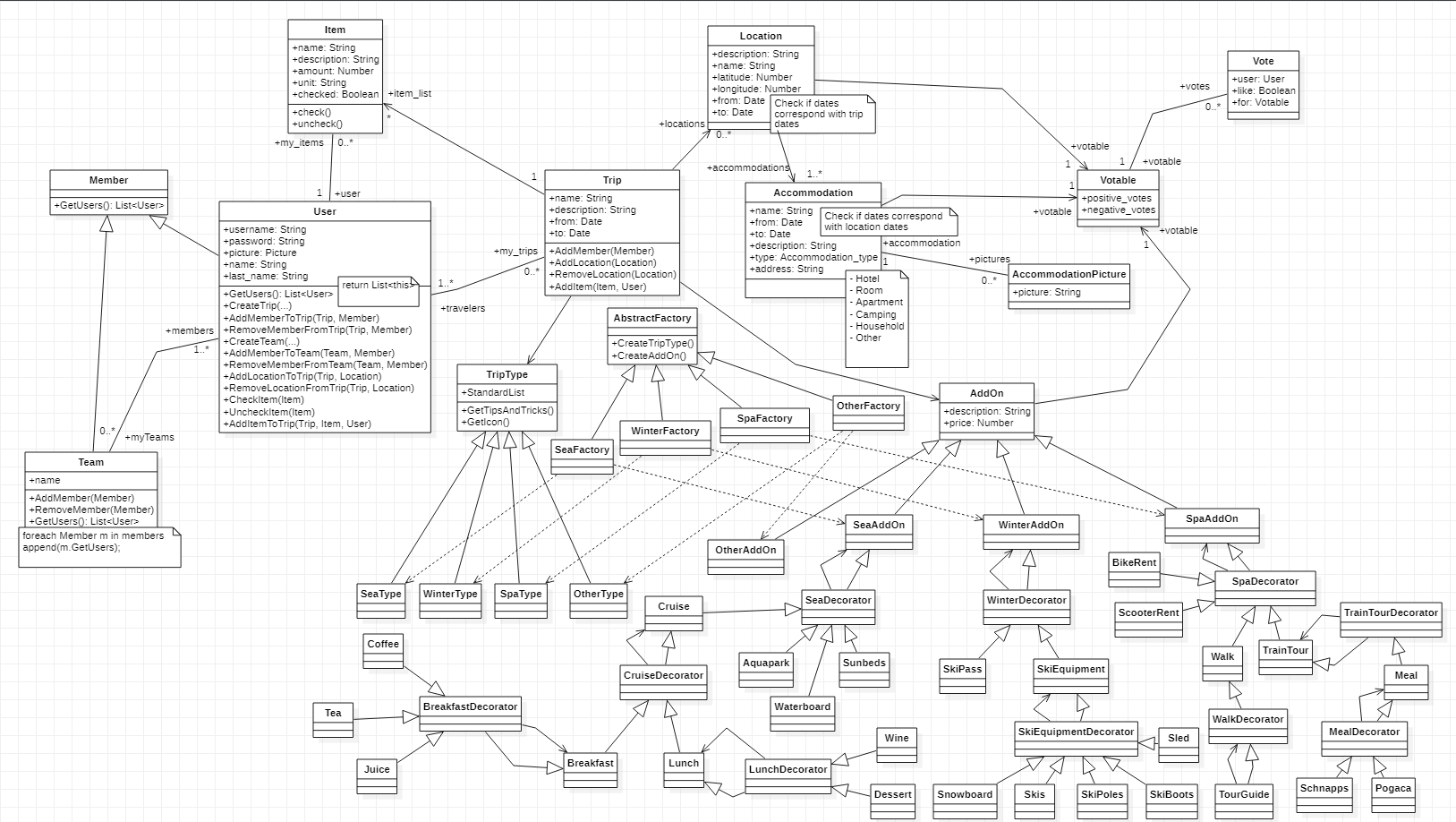
1.12.  [*Votable*](#_heading=h.1fob9te) 5

1.13.  [*Vote*](#_heading=h.1fob9te) 5

1. [**Model entiteta**](#_heading=h.30j0zll) **3**

**3.** [**Mehanizam mapiranja**](#_heading=h.2et92p0) **5**

# 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.

# *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).

# *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).

# *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).

# *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.

# *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.

# *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.

# *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.*

# *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.

# *TripType* hijerarhija