Kratki opis korištenja aplikacije:

Pri pokretanju aplikacije korisniku se otvara početna stranica gdje ima mogućnost da se registruje ili loguje u zavisnosti od čega se otvara i odgovarajući prozor. Dalje ovisno o tome ko se logovao otvara se njegova korisnička forma gdje dalje odabire akcije koje želi obavljati što je već opisano na šablonima formi koje su urađene. Klase koje koristi aplikacija kao i objašnjenje veza između njih date su u nastavku.

Klase

Usluga:
Apstraktna klasa:
Atributi:
- korisnik (Korisnik)
- naziv usluge (string)
- cijena usluge (double)
- sifra usluge (integer)

Metode:
- Getteri i setteri
- Konstruktor sa svim parametrima

Sahrana:
Izvedena iz klase Usluga
Atributi:
- datum sahrane (date)
- pokojnik (UmrlaOsoba)
- vjerski obred (boolean)

Metode:
- Getteri i setteri
- Konstruktor sa svim parametrima

Kremiranje:
Izvedena iz klase Usluga
Atributi:
- datum kremiranja (date)
- pokojnik (UmrlaOsoba)

Metode:
- Getteri i setteri
- Konstruktor sa svim parametrima

PrevozUmrleOsobe:
Izvedena iz klase Usluga
Atributi:
- datum prevoza (date)
- pokojnik (UmrlaOsoba)
- vozač (Radnik)

Metode:
- Getteri i setteri
- Konstruktor sa svim parametrima

Smrtovnica:
Izvedena iz klase Usluga
Atributi:
- pokojnik (UmrlaOsoba)
- slika (file)
- tekst smrtovnice (file)

Metode:
- Getteri i setteri
- Konstruktor sa svim parametrima

GrobnoMjesto:
Atributi:
- parcela (string)
- zauzetost (boolean)
- tip (string) *(da li je muslimansko, pravoslavno, katoličko, jevrejsko, ateističko itd.)*
- vlasnik (Korisnik)
- pokojnik (UmrlaOsoba)

Metode:
- Getteri i setteri
- Konstruktor bez parametara
- Konstruktor sa svim parametrima

Groblje:
Atribut:
- naziv groblja (string)
- grobna mjesta(List<GrobnoMjesto>);
- maksimalan Broj Grobnih Mjesta (int)

Metode:
- Getteri i setteri
- Konstruktor bez parametara
- Konstruktor sa svim parametrima
- DajSlobodnaGrobnaMjesta
- DajZauzetaGrobnaMjesta
- DodajGrobnoMjesto
- IzbrišiGrobnoMjesto
- DajPoUslovu
- IzbrišiPoUslovu

Članarina:
Atribut:
- naziv (string)
- cijena (double)

Metode:
- Getteri i setteri
- Konstruktor sa svim parametrima

Radnik:
Atributi:
- ime i prezime (string)
- naziv radnog mjesta (string)
- bonus (double)

Metode:
- Getteri i setteri
- Konstruktor bez parametara
- Konstruktor sa svim parametrima

Korisnik:
Atributi:
- ime i prezime (string)
- username (string)
- broj telefona (string)
- adresa stanovanja (string)
- datum rođenja (date)
- email (string)
- clanarina (Članarina)
- lista usluga (List<Usluga>)
- poruke (file)

Metode:
- Getteri i setteri
- Konstruktor bez parametara
- Konstruktor sa svim parametrima
- Pošalji zahtjev za uslugu
- Pošalji poruku
- Odaberi paket članarine
- Pregled računa
- Obriši nalog

Šef:
Atributi:
- ime i prezime (string)
- username (string)
- broj telefona (string)
- adresa stanovanja (string)
- datum rođenja (date)
- email (string)
- zahtjevi (List<Usluga>)
- radnici (List<Radnik>)

Metode:
- Odobri zahtjev
- Odbiji zahtjev
- Pregledaj radnike
- Kreiraj radni nalog
- Dodaj radnika
- Obriši radnika

UmrlaOsoba:
Atributi:
- ime prezime (string)
- datum rođenja (date)
- datum smrti (date)

Metode:
- Getteri i setteri
- Konstruktor bez parametara
- Konstruktor sa svim parametrima

Administrator:
Atributi:
- ime i prezime (string)
- username (string)
- broj telefona (string)
- adresa stanovanja (string)
- datum rođenja (date)
- email (string)
- korisnici(List<Korisnik>)
- šefovi (List<Šef>)

Metode:
- Odobri korisnički račun
- Odbiji korisnički račun
- Dodaj novog korisnika
- Obriši korisnika
- Pošalji poruku korisniku
- Pošalji upozorenje

Vlasnik:
Atributi:
- ime i prezime (string)
- username (string)
- broj telefona (string)
- adresa stanovanja (string)
- datum rođenja (date)
- email (string)
- korisnici(List<Korisnik>)
- šefovi (List<Šef>)
- administratori (List<Administrator>)

Metode:
- Dodaj novog korisnika
- Obriši korisnika
- Dodaj novog administratora
- Obriši administratora
- Dodaj novog radnika
- Obriši radnika
- Promijeni podatke po uslovu
- Pošalji poruku korisniku
- Pošalji upozorenje

Što se tiče veza među klasama možemo zaključiti da klasa Usluga može sadržavati više istih insanci klase Korisnik jer jedan Korisnik može koristiti više puta jednu ili više Usluga.

Klase Sahrana, Kremiranje, PrevozUmrleOsobe i Smrtovnica one su izvedene iz apstraktne klase Korisnik i sadrže instance klase UmrlaOsoba i insanca klase UmrlaOsoba može se samo jednom pojaviti u datim klasama što je i zdravorazumski jer umrla osoba ne može umirati više puta.
Klasa PrevozUmrleOsobe sadrži i klasu Radnik čija se jedna instanca može pojaviti više puta u ovoj klasi jer Radnik isti radnik može vršiti više prevoza.

Klasa GrobnoMjesto sadrži u sebi instance klase Korisnik i UmrlaOsoba, jedan Korisnik može se pojaviti više puta tj. imati više zakupljenih grobnih mjesta dok se klasa UmrlaOsoba može pojaviti samo jednom zbog razloga objašnjenog u prethodnom pasusu.

Klasa Groblje je kontejnerska klasa u kojoj imamo listu elemenata tipa GrobnoMjesto.

Klasa Korisnik opisuje običnog korisnika koji se registruje, a poslije prijavljuje na sistem. U toj klasi imamo instance klasa Članarina čija se jedna instanca može pojaviti više puta u instancama klase Korisnik. U ovoj klasi imamo i listu elementata tipa klase Usluga.

Klasa Šef u sebi ima liste elemenata tipa klasa Usluga i Radnik radi lakše manipulacije sa njima.

Klasa Administrator u sebi ima liste elemenata tipa klasa Korisnik i Šef radi lakše manipulacije.

Klasa Vlasnik kao klasa koja predstavlja korisnika sa najvećim pravom pristupa u sebi sadrži liste elemenata klasa Korisnik, Šef i Administrator.

Veza između klase Usluga sa klasama Sahrana, Kremiranje, PrevozUmrleOsobe i Smrtovnica je generalizacija čije je drugo ime nasljeđivanje što je slučaj u ovom primjeru.

Klase Šef, Administrator i Vlasnik imaju u sebi liste ovih klasa (klasa Šef sadži i klasu Radnik) u zavisnosti o kojoj klasi govorimo i veza ostvarena među njima je asocijacija.

Veza između klasa Sahrana, Kremiranje, PrevozOsobe i Smrtovnica sa klasom Umrla osoba je veza agregacije.

Klasa GrobnoMjesto je sa klasama UmrlaOsoba i Korisnik u vezi agregacije.

Između Korisnik i Usluga veza je asocijacija, dok je između Korisnik i Članarina agregacija.

Solid principi:

Princip *S* :
- Ovaj princip je ispoštovan jer svaka klasa radi ono za šta je i namijenjena.

Princip *O* :
- Prilikom dizajniranja klasa trudili smo se da klase koje su u vezi komuniciraju bez teških radnji da bi izbjegli mijenjanje jedne klase radi druge.

Princip *L* :
- Liskov princip je ispoštovan što se tiče usluga jer sve usluge su izvedene iz apstraktne klase "Usluga" i mogu se koristiti na mjestima gdje kod koristimo klasu "Usluga".

Princip *I* :
- Interfejsi nisu korišteni jer nemamo "debelih klasa" i klase su dizajnirane tako da one imaju metode koje će se sigurno koristiti i bez onih metoda koje nam neće trebat.

Princip *D* :
- Pokušali smo ovaj princip ispoštovati uvodeći klasu "Usluga" koja je bazna klasa a ujedno i apstraktna.