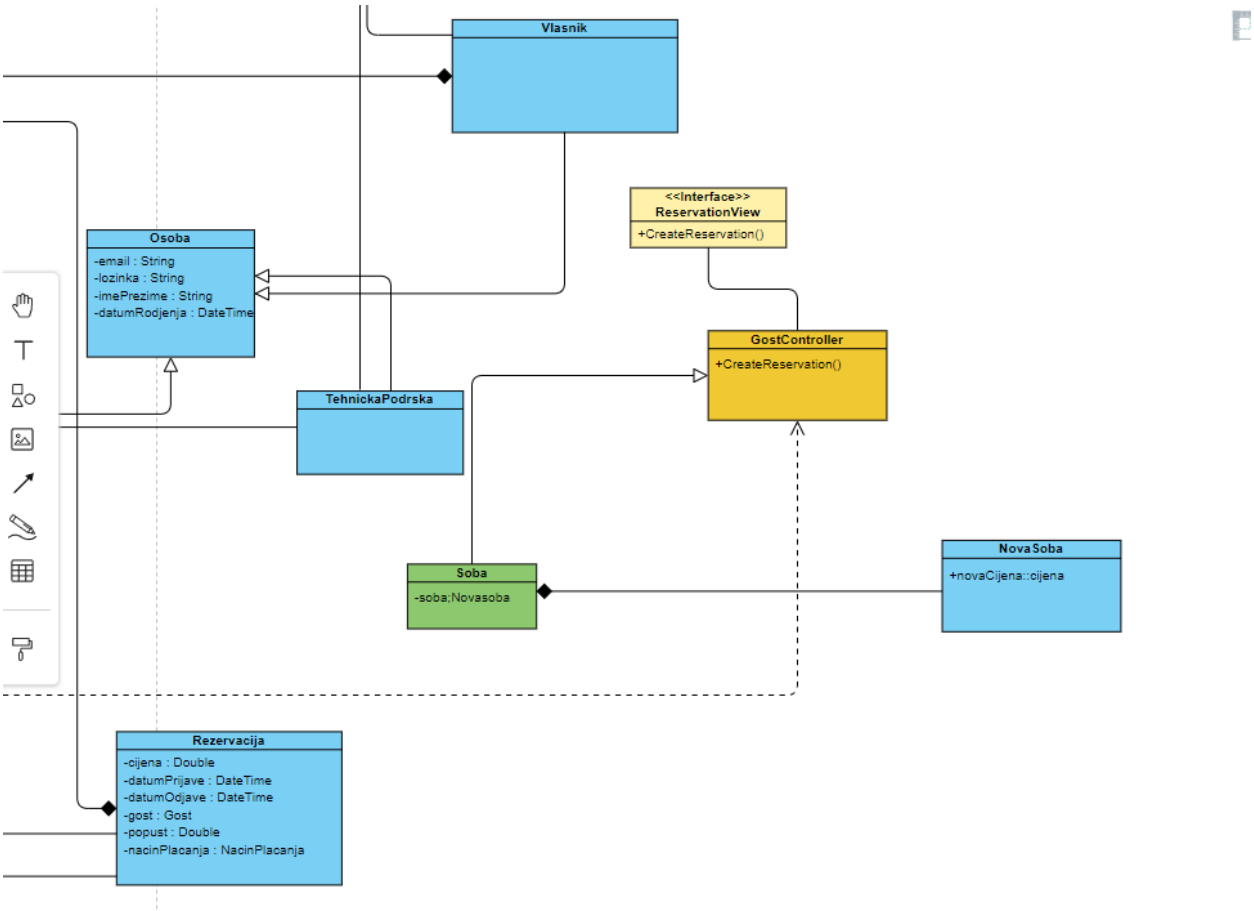
Strukturalni paterni

1.**Adapter patern**

Adapter patern se koristi najčešće radi proširenja upotrebe već postojeće klase. Adapter patern kreira novu adapter klasu koja nam služi kao posrednik između orginalne klase i željenog interfejsa. U našem projektu hotela možemo ga iskoristiti na način da ukoliko dođe do renoviranja neke sobe, odnosno nove cijene, gost je dužan ponovno rezervisati sobu po novoj cijeni ili promijeniti sobu.



2. **Fasadni patern**

Fasadni patern koristimo kada želimo da klijentima osiguramo više pogleda visokog nivoa na podsisteme, odnosno da sakrijemo implementaciju od korisnika. Ovaj patern nećemo koristiti zbog toga što smo projekat bazirali na vrlo jednostavnim klasama, ali hipotetički govoreći o ovom paternu, ukoliko bi recimo implementirali restoran u naš hotel koji će imati više meni-a za odabir hrane, mogli bi napraviti klasu „Konobar“ koji će djelovati kao „fasada“ između implementacije meni-a i naručivanja. Umjesto da korisnik gleda 3-4 meni-a, olakšano će mu biti tako što može pitati konobara npr. za specijalitet hotela.

3.**Decorator patern**

Decorater patern koristimo kada ne želimo praviti veći broj podklasa koje predstavljaju kombinacije već postojećih klasa, već u cilju nam je da koristimo klase koje smo već napravili. U našem sistemu nećemo koristiti ovaj patern iz razloga jer je projekat baziran na vrlo jednostavnim klasama,ali hipotetički govoreći način na koji bi se mogao implementirati decorator patern je taj kada npr. imamo različite boje, dizajn, veličine unaprijed određene. I kada trebamo opisati neku sobu koristimo atribute i metode iz klase „DecoratorSobe“ koje nam olakšavaju da se ne zapetljamo u implementaciji i da je olakšamo, te samim tim korisnik ima detaljniji prikaz te sobe.

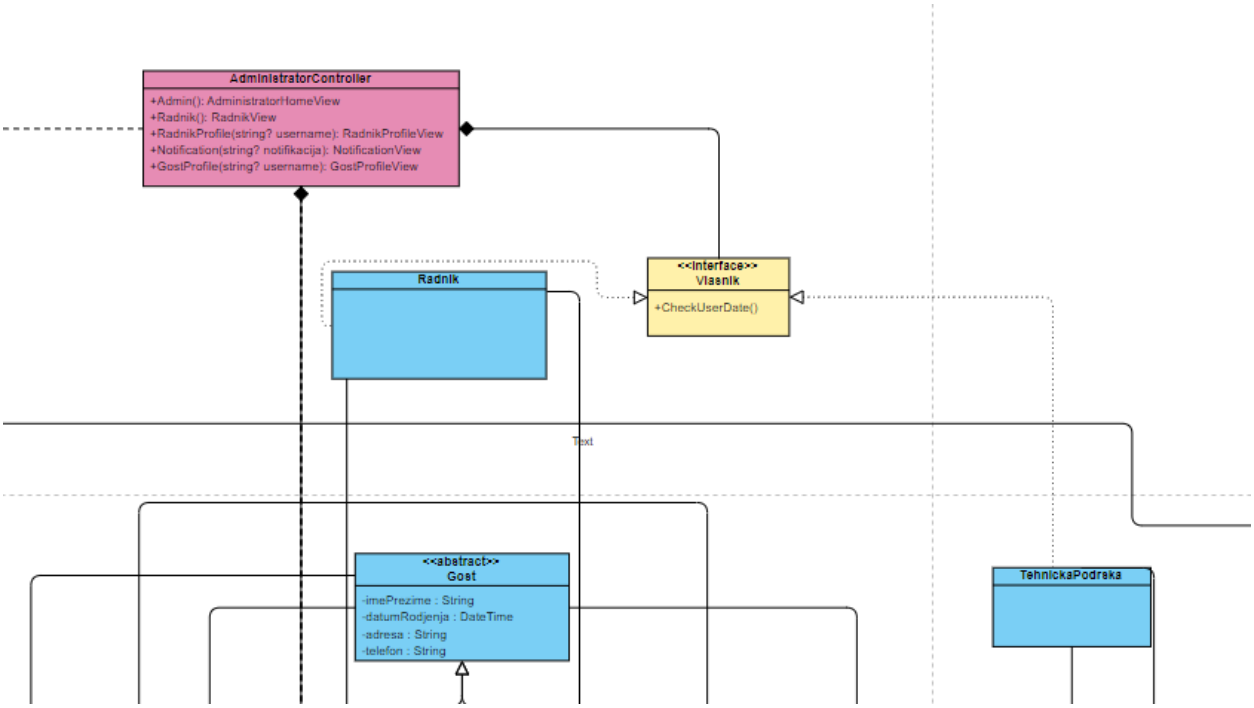
4.**Bridge patern**

Bridge patern nam omogućava da se iste operacije primjenjuju nad različitim podklasama. Omogućava odvajanje apstrakcije i implementacije na takav način da klasa može posjedovati više apstrakcija i više različitih implementacija za pojedine apstrakcije. U

našem projektu moguće je izvesti ovaj patern na način da korisnik tokom kreiranja korisničkog računa dobije potvrdu odnosno kod na mail koji je koristio. I da bi potvrdio svoj identitet on mora ukucati kod na svom profilu, odnosno izvršiti potvrdu, manuelno ili copy-pasteom.

5.**Kompozitni patern**

Kompozitni patern služi za kreiranje hijerarhije objekata. Koristi se najčešće kada svi objekti imaju implementacije koje se razlikuju za određene metode, ali je potreba da im pristupimo na isti način. Cilj je da kroz zajednički interfejs pristupimo svim potrebnim klasama. U našem projektu recimo da hotel želi pristupiti informacijama od „Radnika“ i „Tehničke podrške“.Vlasnik želi iznenaditi svoje radnike sa poklonom. Primjenom kompozitnog paterna datum rođenja radnika i tehničke podrške izvršit će se kroz 1 interfejs koji povezuje 2 klase.



6.**Proxy patern**

Proxy patern nam omogućava da zamijenimo ili da sačuvamo neko mjesto za drugi objekat. Ovaj patern je zaslužan za kontrolu pristupa orginalnom objektu, dozvoljavajući nam da izvršimo akciju prije ili nakon što zahtjev prođe do orginalnog objekta. U našem projektu ovaj patern možemo iskoristiti na način da implementiramo interfejs „Plaćanje“, koji će biti povezan sa „KreditnomKarticom“ i „Kešom“. Klasa „Kreditna kartica“ je proxy za bankovni račun, a bankovni račun je proxy za veću svotu novca. Gost neće morati nositi sa sobom veću količinu novca, dok sa druge strane to odgovara i vlasniku koji neće morati da se brine da li će gost izgubiti taj novac, odnosno mogućnost gubljenja depozita je približna nuli.

7.**Flyweight patern**

Flyweight patern nam služi kako bi se onemogućilo bespotrebno stvaranje velikog broja instanci objekata koji u suštini predstavljaju jedan objekat. Ukoliko postoji potreba za kreiranjem specifičnog objekta sa jedinstvenim karakteristikama, izvršit će se i to nazivamo „special state“. Postojanje „shared object“ malo usporava program, ali za zadatak ima smanjenje troškova memorije. Ovaj patern u našem projektu se može primijeniti na način da ukoliko radnik hotela objavljuje oglas za neku novu sobu na nekom spratu. Nema potrebe da dodaje nove slike, nego može iskoristiti već dodane slike jer na određenom spratu sve sobe su iste.