Analiza korištenih patterna

Singleton pattern: Singleton pattern je nužan za naš sistem jer nema smisla da postoji mogućnost kreiranja više instanci klase Teretana iz razloga što ona treba biti jedinstvena i kao takva treba moći jedinstveno upravljati akcijama u sistemu. Ovaj patern omogućava instanciranje klase Teretana samo onda kada se prvi put javi potreba za tom klasom, odnosno prilikom prvog kreiranja objekta tog tipa.

Adapter pattern: Ovaj pattern zadovoljava Single responsibilty principle i Open/closed principle te je iz tih razloga posebno pogodan za korištenje u sistemu. On omogućava razdavanje pretvorbe tipa podataka od poslovne logike samog sistema, a mi u našem sistemu smo ovaj pattern upravo iskoristili za pretvorbu objekta tipa Trening u objekat tipa Novost kako bi se kreiranjem novog treninga kreirala i novost koja obavještava članove o tom treningu.

Proxy pattern: Proxy pattern smo koristili kako bi se korisnicima koji trenutno nisu u teretani onemogućilo zauzimanje opreme unutar same teretane iz razloga što bi to omogućilo zloupotrebu ove funkcionalnosti (npr. neko bi mogao pomisliti da ne može koristiti neku opremu u teretani iz razloga što piše da je zauzeta iako je tu opremu na sistemu zauzeo neko ko je trenutno kod kuće). Ovim se postigla veća sigurnost sistema i kontrola pristupa sistemu.

Prototype pattern: Ovaj pattern je koristan iz razloga što omogućava kreiranje novih objekata kopirajući samo neki već postojeći prototip za taj objekat (znači bez korištenja operatora new), te samo izvršavanja nekih izmjena nad tim objektom ako su one potrebne. Mi smo ovaj pattern iskoristili za kreiranje novih novosti iz razloga što one često mogu imati isti sadržaj samo će im se razlikovati datum i vrijeme objavljivanja pa samim tim i vrijeme njihove relevantnosti, a novosti također imaju i defaultnu vrijednost za sliku, pa je imalo smisla iskoristiti prototype pattern.