

Specifikacija projekta

1. Osnovne informacije o sistemu

Naziv teme: Levres

Logo:



Naziv tima: Tim 26

Nastavna grupa: 7

Link na repozitorij tima: <https://github.com/OOAD-2023-2024/Tim26-Levres>

Članovi tima:

1. Benjamin Baždar, 19473
2. Ensar Hodžić, 19449
3. Ajla Hrustić, 19339

Namjena sistema:

Opisati sistem i njegovu namjenu sa maksimalno sedam rečenica. U okviru ovog polja potrebno je objasniti šta sistem treba raditi na apstraktnom nivou, bez detaljnog objašnjavanja pojedinačnih funkcionalnosti i načina razlikovanja aktera sistema (što je predmet daljih poglavlja).

Levres je nova autokompanija koja prodaje automobile na novi način. Umjesto fizičkih lokacija, Levres koristi web tehnologije kako bi prodavali nove i polovne automobile na njihovoj web platformi. Na stranici možete također konfigurisati vaš novi automobil po vašoj želji. Također nudi prodaju dodatne opreme, kao i prijava na servis u slučaju kvara na automobilu i na kraju možete uslikati vaš trenutni automobil da dobijete procjenu vrijednosti.

2. Funkcionalnosti (poslovni procesi) sistema

Opisati 6 do 8 najznačajnijih funkcionalnosti sistema (u zavisnosti od broja članova u timu). Funkcionalnosti sistema predstavljaju usluge koje sistem pruža korisnicima. Sve funkcionalnosti pripadaju nekoj od različitih vrsta:

- Usluga sistema - u svrhu ostvarivanja krajne usluge sistema,
- Perzistencija podataka (CRUD operacije)
- Asinhrona operacija - operacije koje koriste principe asinhronne obrade zahtjeva
- Operacija sa specifičnim algoritmom obrade - operacije koje koriste specifične algoritme obrade podataka,
- Korištenje vanjskog uređaja - operacije u kojima se vrši korištenje vanjskih uređaja.

Neophodno je navesti barem po jednu funkcionalnost svake od različitih vrsta.

1) **Naziv funkcionalnosti:** Kupovina novih i polovnih automobila

Vrsta funkcionalnosti: Usluga sistema

Opis funkcionalnosti:

Opisati način ostvarivanja funkcionalnosti sa maksimalno pet rečenica.

Korisnici imaju pregled novih automobila na prodaju koji se trenutno nalaze na stanju, to jest da ne moraju čekati da se proizvede. Također imaju opciju za pregled polovnih automobila. Imaju opcije za filtriranje, tipa filtriranje po tipu auta, boji auta, snazi motora, i sl.

2) **Naziv funkcionalnosti:** Postavljanje automobila na prodaju

Vrsta funkcionalnosti: Perzistencija podataka (CRUD operacija)

Opis funkcionalnosti:

Opisati način ostvarivanja funkcionalnosti sa maksimalno pet rečenica.

Ovom funkcionalnošću se kontroliše prodaja novih i polovnih automobila. To podrazumijeva postavljanje novih automobila na prodaju, promjena podataka na određenom automobilu, kao i sklanjanje sa prodaje.

3) **Naziv funkcionalnosti:** Konfiguracija novih automobila

Vrsta funkcionalnosti: Operacija sa specifičnim algoritmom obrade

Opis funkcionalnosti:

Opisati način ostvarivanja funkcionalnosti sa maksimalno pet rečenica.

Korisnici imaju mogućnost biranja modela novog automobila te konfiguraciju njegovih opcija, poput biranje vanjske boje, biranje boje unutrašnjosti, biranje točkova, i sl. U pozadini program provjerava trenutnu dostupnost tih opcija i prilagođava totalnu cijenu i vrijeme čekanja za automobil. Administratori imaju mogućnost promjene dostupnosti određenih opcija.

4) Naziv funkcionalnosti: Zahtjev za servis automobila

Vrsta funkcionalnosti: Perzistencija podataka (CRUD operacija)

Opis funkcionalnosti:

Opisati način ostvarivanja funkcionalnosti sa maksimalno pet rečenica.

Korisnici imaju mogućnost da pregledaju njihovu historiju servisa, kao i da se prijave na za novi servis njihovog automobila. Zaposlenici imaju mogućnost pregleda servisa određenog vozila.

5) Naziv funkcionalnosti: Procjena vrijednosti vozila

Vrsta funkcionalnosti: Korištenje vanjskog uređaja

Opis funkcionalnosti:

Opisati način ostvarivanja funkcionalnosti sa maksimalno pet rečenica.

Koristeći kameru na telefonu, zaposlenik može uslikati polovno vozilo, i uz pomoć dodatnih informacija koju bi unio, dobio bi procjenu vrijednosti vozila.

6) Naziv funkcionalnosti: Pregled zamjenskih auto dijelova

Vrsta funkcionalnosti: Operacija sa specifičnim algoritmom obrade

Opis funkcionalnosti:

Opisati način ostvarivanja funkcionalnosti sa maksimalno pet rečenica.

Sa ovom funkcionalnošću, uposlenik može da odabere model automobila, i koji zamjenski dio mu je potreban, i dobije listu svih dijelova koji odgovaraju tom vozilu te da li su trenutno na stanju ili je potrebno ih naručiti.

3. Akteri sistema

Potrebno je navesti najmanje tri aktera sistema.

Vrste aktera:

- Korisnik sistema
- Zaposlenik sistema
- Administrator

Neophodno je navesti barem po jednog aktera za svaku od različitih vrsta.

Korisnici usluga sistema

a) **Naziv aktera: Kupac**

Vrsta aktera: Korisnik usluge

Funkcionalnosti u kojima akter učestvuje:

Funkcionalnost sistema	Način učešća
1. Kupovina polovnih i novih automobila	Mogućnost pregleda
3. Konfiguracija novih automobila	Mogućnost uređivanja
4. Zahtjev za servis automobila	Mogućnost uređivanja

b) **Naziv aktera: Uposlenik**

Vrsta aktera: Zaposlenik sistema

Funkcionalnosti u kojima akter učestvuje:

Funkcionalnost sistema	Način učešća
5. Procjena vrijednosti vozila	Mogućnost pregleda
6. Pregled zamjenskih auto dijelova	Mogućnost uređivanja
4. Zahtjev za servis automobila	Mogućnost pregleda

c) **Naziv aktera: Administrator**

Vrsta aktera: Administrator

Funkcionalnosti u kojima akter učestvuje:

Način učešća:

- Mogućnost pregleda*
- Mogućnost uređivanja*

Funkcionalnost sistema	Način učešća
2. Postavljanje automobila na prodaju	Mogućnost uređivanja
3. Konfiguracija novih automobila	Mogućnost uređivanja
1. Kupovina novih i polovnih automobila	Mogućnost pregleda
4. Zahtjev za servis automobila	Mogućnost pregleda
5. Procjena vrijednosti vozila	Mogućnost uređivanja
6. Pregled zamjenskih auto dijelova	Mogućnost uređivanja

4. Nefunkcionalni zahtjevi sistema

Opisati najmanje tri najznačajnija nefunkcionalna zahtjeva sistema. Nefunkcionalni zahtjevi predstavljaju ograničenja koja sistem mora zadovoljiti kako bi mogao ispravno obavljati svoje funkcionalnosti. Validacije polja za unos vrijednosti ne predstavljaju nefunkcionalne zahtjeve.

- 1) **Naziv nefunkcionalnog zahtjeva:** Provjera dostupnosti opcija vozila pri konfiguraciji

Opis:

Opisati ograničenje sistema i način na koje se ono ispoljava.

Korisnici pri konfiguraciji novog vozila, mora imati tačne informacije o tome da li su trenutne opcije dostupne, te koliko koštaju i koliko je vrijeme čekanja na završetak fabriciranja automobila.

- 2) **Naziv nefunkcionalnog zahtjeva:** Sistem za registraciju različitih korisnika

Opis:

Opisati ograničenje sistema i način na koje se ono ispoljava.

Ovaj sistem ima tri različita aktera, tako da u zavisnosti od koje vrste je akter, aplikacija će pružiti različite mogućnosti u zavisnosti da li je zaposlenik ili administrator ili kupac

- 3) **Naziv nefunkcionalnog zahtjeva:** Sistem za aproksimaciju vrijednosti vozila

Opis:

Opisati ograničenje sistema i način na koje se ono ispoljava.

Funkcionalnost koja vrši procjenu vrijednosti polovnih automobila mora biti konstantno ažurirana sa trenutnim vrijednostima različitih modela automobila, te koliko različite specifikacije automobila utiču na cijenu.

Analiza i dizajn sistema

U nastavku je potrebno definisati sve potencijalne klase koje će se koristiti u sistemu. Za određivanje klasa koje će biti neophodne za rad sistema potrebno je koristiti specifikaciju sistema i prethodno kreirane dijagrame.

Template za jednu klasu potrebno je iskopirati onoliko puta koliko je neophodno da bi se definisale sve klase u sistemu.

Definicija klase u sistemu

Naziv klase: Automobil(Apstraktna)

Funkcionalni zahtjevi u kojima klasa učestvuje:

FZ br. 01: Kupovina novih i polovnih automobila

FZ br. 02: Postavljanje automobila na prodaju

FZ br. 03: Konfiguracija novih automobila

Atributi koje klasa posjeduje:

Naziv atributa	Tip varijable	Dodatne napomene
ID	int	<input type="checkbox"/> Atribut je statički <input type="checkbox"/> Atribut je enumeration
Model	string	<input type="checkbox"/> Atribut je statički <input type="checkbox"/> Atribut je enumeration
Godina proizvodnje	date	<input type="checkbox"/> Atribut je statički <input type="checkbox"/> Atribut je enumeration
Gorivo	string	<input type="checkbox"/> Atribut je statički <input checked="" type="checkbox"/> Atribut je enumeration
Transmisijska skupina	string	<input type="checkbox"/> Atribut je statički <input checked="" type="checkbox"/> Atribut je enumeration
Broj vrata	int	<input type="checkbox"/> Atribut je statički <input checked="" type="checkbox"/> Atribut je enumeration
Boja	string	<input type="checkbox"/> Atribut je statički <input checked="" type="checkbox"/> Atribut je enumeration
Pogon	string	<input type="checkbox"/> Atribut je statički <input checked="" type="checkbox"/> Atribut je enumeration
Veličina felgi	int	<input type="checkbox"/> Atribut je statički <input checked="" type="checkbox"/> Atribut je enumeration

Emisioni standard	string	<input type="checkbox"/> Atribut je statički <input checked="" type="checkbox"/> Atribut je <i>enumeration</i>
Sjedeća mjesta	int	<input type="checkbox"/> Atribut je statički <input checked="" type="checkbox"/> Atribut je <i>enumeration</i>
Masa/Težina	double	<input type="checkbox"/> Atribut je statički <input checked="" type="checkbox"/> Atribut je <i>enumeration</i>
Vrsta enterijera	string	<input type="checkbox"/> Atribut je statički <input checked="" type="checkbox"/> Atribut je <i>enumeration</i>
Svjetla	string	<input type="checkbox"/> Atribut je statički <input checked="" type="checkbox"/> Atribut je <i>enumeration</i>
Cijena	double	<input type="checkbox"/> Atribut je statički <input type="checkbox"/> Atribut je <i>enumeration</i>
Slike	string	<input type="checkbox"/> Atribut je statički <input type="checkbox"/> Atribut je <i>enumeration</i>
Motor	string	<input type="checkbox"/> Atribut je statički <input type="checkbox"/> Atribut je <i>enumeration</i>

Naziv klase: Novi Automobil(Izveden iz Automobil)

Funkcionalni zahtjevi u kojima klasa učestvuje:

FZ br. 01: Kupovina novih i polovnih automobila

FZ br. 02: Postavljanje automobila na prodaju

FZ br. 03: Konfiguracija novih automobila

Atributi koje klasa posjeduje:

Naziv atributa	Tip varijable	Dodatne napomene
Garancija	int	<input type="checkbox"/> Atribut je statički <input type="checkbox"/> Atribut je <i>enumeration</i>

Naziv klase: Polovni Automobil(Izveden iz Automobil)

Funkcionalni zahtjevi u kojima klasa učestvuje:

FZ br. 01: Kupovina novih i polovnih automobila

FZ br. 02: Postavljanje automobila na prodaju

FZ br. 03: Konfiguracija novih automobila

Atributi koje klasa posjeduje:

Naziv atributa	Tip varijable	Dodatne napomene
Kilometraža	double	<input type="checkbox"/> Atribut je statički <input type="checkbox"/> Atribut je <i>enumeration</i>
Štete	string	<input type="checkbox"/> Atribut je statički <input type="checkbox"/> Atribut je <i>enumeration</i>
Broj vlasnika	int	<input type="checkbox"/> Atribut je statički <input type="checkbox"/> Atribut je <i>enumeration</i>

Naziv klase: Kupac

Funkcionalni zahtjevi u kojima klasa učestvuje:

FZ br. 01: Kupovina novih i polovnih automobila

FZ br. 03: Konfiguracija novih automobila

FZ br. 04: Zahtjev za servis automobila

Atributi koje klasa posjeduje:

Naziv atributa	Tip varijable	Dodatne napomene
ID	int	<input type="checkbox"/> Atribut je statički <input type="checkbox"/> Atribut je <i>enumeration</i>
Ime	string	<input type="checkbox"/> Atribut je statički <input type="checkbox"/> Atribut je <i>enumeration</i>
Prezime	string	<input type="checkbox"/> Atribut je statički <input type="checkbox"/> Atribut je <i>enumeration</i>
Email	string	<input type="checkbox"/> Atribut je statički <input type="checkbox"/> Atribut je <i>enumeration</i>
Šifra	string	<input type="checkbox"/> Atribut je statički <input type="checkbox"/> Atribut je <i>enumeration</i>
Datum rođenja	date	<input type="checkbox"/> Atribut je statički <input type="checkbox"/> Atribut je <i>enumeration</i>
Adresa	string	<input type="checkbox"/> Atribut je statički <input type="checkbox"/> Atribut je <i>enumeration</i>
Država	string	<input type="checkbox"/> Atribut je statički <input type="checkbox"/> Atribut je <i>enumeration</i>
Grad	string	<input type="checkbox"/> Atribut je statički <input type="checkbox"/> Atribut je <i>enumeration</i>

Poštanski broj	string	<input type="checkbox"/> Atribut je statički <input type="checkbox"/> Atribut je <i>enumeration</i>
----------------	--------	--

Naziv klase: Radnik

Funkcionalni zahtjevi u kojima klasa učestvuje:

FZ br. 01: Kupovina novih i polovnih automobila

FZ br. 03: Konfiguracija novih automobila

FZ br. 04: Zahtjev za servis automobila

Atributi koje klasa posjeduje:

Naziv atributa	Tip varijable	Dodatne napomene
ID	int	<input type="checkbox"/> Atribut je statički <input type="checkbox"/> Atribut je <i>enumeration</i>
Ime	string	<input type="checkbox"/> Atribut je statički <input type="checkbox"/> Atribut je <i>enumeration</i>
Prezime	string	<input type="checkbox"/> Atribut je statički <input type="checkbox"/> Atribut je <i>enumeration</i>
Email	string	<input type="checkbox"/> Atribut je statički <input type="checkbox"/> Atribut je <i>enumeration</i>
Šifra	string	<input type="checkbox"/> Atribut je statički <input type="checkbox"/> Atribut je <i>enumeration</i>
Datum rođenja	date	<input type="checkbox"/> Atribut je statički <input type="checkbox"/> Atribut je <i>enumeration</i>
Adresa	string	<input type="checkbox"/> Atribut je statički <input type="checkbox"/> Atribut je <i>enumeration</i>
Država	string	<input type="checkbox"/> Atribut je statički <input type="checkbox"/> Atribut je <i>enumeration</i>
Grad	string	<input type="checkbox"/> Atribut je statički <input type="checkbox"/> Atribut je <i>enumeration</i>
Poštanski broj	string	<input type="checkbox"/> Atribut je statički <input type="checkbox"/> Atribut je <i>enumeration</i>
Plata	int	<input type="checkbox"/> Atribut je statički <input type="checkbox"/> Atribut je <i>enumeration</i>
Datum zaposlenja	date	<input type="checkbox"/> Atribut je statički <input type="checkbox"/> Atribut je <i>enumeration</i>
Trajanje ugovora	date	<input type="checkbox"/> Atribut je statički <input type="checkbox"/> Atribut je <i>enumeration</i>

Naziv klase: Narudžba

Funkcionalni zahtjevi u kojima klasa učestvuje:

FZ br. 01: Kupovina novih i polovnih automobila

FZ br. 03: Konfiguracija novih automobila

Atributi koje klasa posjeduje:

Naziv atributa	Tip varijable	Dodatne napomene
ID	int	<input type="checkbox"/> Atribut je statički <input type="checkbox"/> Atribut je enumeration
Automobil_ID	int	<input type="checkbox"/> Atribut je statički <input type="checkbox"/> Atribut je enumeration
Kupac_ID	int	<input type="checkbox"/> Atribut je statički <input type="checkbox"/> Atribut je enumeration
Datum	date	<input checked="" type="checkbox"/> Atribut je statički <input type="checkbox"/> Atribut je enumeration
Tip	string	<input type="checkbox"/> Atribut je statički <input type="checkbox"/> Atribut je enumeration

Naziv klase: Servis

Funkcionalni zahtjevi u kojima klasa učestvuje:

FZ br. 04: Zahtjev za servis automobila

Atributi koje klasa posjeduje:

Naziv atributa	Tip varijable	Dodatne napomene
ID	int	<input type="checkbox"/> Atribut je statički <input type="checkbox"/> Atribut je enumeration
Kupac_ID	int	<input type="checkbox"/> Atribut je statički <input type="checkbox"/> Atribut je enumeration
Model	string	<input type="checkbox"/> Atribut je statički <input checked="" type="checkbox"/> Atribut je enumeration
Registracijske tablice	string	<input type="checkbox"/> Atribut je statički <input type="checkbox"/> Atribut je enumeration
Opis	string	<input type="checkbox"/> Atribut je statički <input type="checkbox"/> Atribut je enumeration

Naziv klase: Oprema

Funkcionalni zahtjevi u kojima klasa učestvuje:

FZ br. 06: Pregled zamjenskih auto dijelova

Atributi koje klasa posjeduje:

Naziv atributa	Tip varijable	Dodatne napomene
ID	int	<input type="checkbox"/> Atribut je statički <input type="checkbox"/> Atribut je enumeration
Naziv dijela	string	<input type="checkbox"/> Atribut je statički <input type="checkbox"/> Atribut je enumeration
Model	string	<input type="checkbox"/> Atribut je statički <input checked="" type="checkbox"/> Atribut je enumeration
Količina	int	<input type="checkbox"/> Atribut je statički <input type="checkbox"/> Atribut je enumeration
Cijena	double	<input type="checkbox"/> Atribut je statički <input type="checkbox"/> Atribut je enumeration

Naziv klase: Konfiguracija

Funkcionalni zahtjevi u kojima klasa učestvuje:

FZ br. 03: Konfiguracija novih automobila

Atributi koje klasa posjeduje:

Naziv atributa	Tip varijable	Dodatne napomene
ID	int	<input type="checkbox"/> Atribut je statički <input type="checkbox"/> Atribut je enumeration
Model	string	<input type="checkbox"/> Atribut je statički <input checked="" type="checkbox"/> Atribut je enumeration
Boja	string	<input type="checkbox"/> Atribut je statički <input checked="" type="checkbox"/> Atribut je enumeration
Veličina felgi	int	<input type="checkbox"/> Atribut je statički <input checked="" type="checkbox"/> Atribut je enumeration
Vrsta interijera	string	<input type="checkbox"/> Atribut je statički <input checked="" type="checkbox"/> Atribut je enumeration
Svjetla	string	<input type="checkbox"/> Atribut je statički <input checked="" type="checkbox"/> Atribut je enumeration

Motor	string	<input type="checkbox"/> Atribut je statički <input checked="" type="checkbox"/> Atribut je <i>enumeration</i>
-------	--------	---

Slučajevi upotrebe

Scenario 1: Kupovina polovnih i novih automobila

Naziv slučaja upotrebe	Kupovina polovnih i novih automobila
Opis slučaja upotrebe	Kupac vrši pregled trenutnih polovnih i novih vozila na stanju
Vezani zahtjevi	Nema vezanih zahtjeva
Preduslovi	Korisnik mora biti registrovan i prijavljen
Posljedice – uspješan završetak	Kupac uspješno izvrši registraciju i kupovinu automobila
Posljedice – neuspješan završetak	Kupac nije uspio se registrirati ili nije uspio kupiti automobil jer više nije na stanju
Primarni akteri	Kupac
Ostali akteri	/
Glavni tok	Kupac otvara pregled svih dostupnih automobila, filtrira po sva vozila po dostupnim opcijama po želji, klikom na automobil pregleda detaljne informacije te kupuje automobil
Alternative/proširenja	Alternativni tok događaja 1: 1. Korisnik pristupa stranici 2. Korisnik otvara stranicu za kupovinu automobila <i>Koraci 3. do 8. isti kao koraci toka događaja 1. od 7. do 12.</i> 9. Sistem traži od kupca da se registruje ili prijavi 10. Korisnik vrši registraciju 11. Sistem provjerava dostupnost vozila 12. Sistem traži autentifikaciju korisnika 13. Sistem traži način plaćanja vozila 14. Korisnik unosi svoje detalje plaćanja 15. Sistem obavijesti korisnika o uspješnoj prodaji

Tok događaja 1: Uspješan završetak (Kupovina automobila)

Korisnik	Sistem
1. Korisnik pristupa stranici	

2. Korisnik pritisne na dugme za registraciju	
	3. Sistem otvara stranicu za prijavu
4. Korisnik unosi svoje podatke	
	5. Sistem izvrši prijavu korisnika i vrati ga na početnu stranicu
6. Korisnik otvara stranicu za kupovinu polovnih i novih automobila	
	7. Sistem učitava sva trenutna dostupna vozila
8. Korisnik vrši filtriranje vozila po želji	
	9. Sistem učitava filtrirana vozila
10. Korisnik klikne na vozilo koje ga interesuje	
	11. Sistem učitava detaljan pregled vozila
12. Korisnik pritisne dugme za kupovinu	
	13. Sistem provjerava dostupnost vozila
	14. Sistem traži autentifikaciju korisnika
	15. Sistem traži način plaćanja vozila
16. Korisnik unosi svoje detalje plaćanja	
	17. Sistem obavijesti korisnika o uspješnoj prodaji

Tok događaja 2: Neuspješan završetak (Automobil nije dostupan)

Korisnik	Sistem
1. Korisnik pristupa stranici	
2. Korisnik pritisne na dugme za registraciju	
	3. Sistem otvara stranicu za prijavu
4. Korisnik unosi svoje podatke	
	5. Sistem izvrši prijavu korisnika i vrati ga na početnu stranicu

6. Korisnik otvara stranicu za kupovinu polovnih i novih automobila	
	7. Sistem učitava sva trenutna dostupna vozila
8. Korisnik vrši filtriranje vozila po želji	
	9. Sistem učitava filtrirana vozila
10. Korisnik klikne na vozilo koje ga interesuje	
	11. Sistem učitava detaljan pregled vozila
12. Korisnik pritisne dugme za kupovinu	
	13. Sistem provjerava dostupnost vozila
	14. Sistem obavijesti korisnika da vozilo više nije dostupno na prodaju

Tok događaja 3: Neuspješan završetak (Neuspješna registracija)

Korisnik	Sistem
1. Korisnik pristupa stranici	
2. Korisnik pritisne na dugme za registraciju	
	3. Sistem otvara stranicu za prijavu
4. Korisnik unosi svoje podatke	
	5. Sistem obavijesti korisnika da registracija nije uspjela

Scenario 2: Konfiguracija novih automobila

Naziv slučaja upotrebe	Konfiguracija novih automobila
Opis slučaja upotrebe	Kupac vrši konfiguraciju svog novog vozila
Vezani zahtjevi	Nema vezanih zahtjeva
Preduslovi	Korisnik mora biti registrovan i prijavljen
Posljedice – uspješan završetak	Kupac uspješno izvrši registraciju i konfiguraciju automobila
Posljedice – neuspješan završetak	Kupac nije uspio se registrirati ili trenutno nisu dostupne odabrane opcije
Primarni akteri	Kupac
Ostali akteri	/
Glavni tok	Kupac otvara stranicu za konfiguraciju, vrši odabir modela i dijelova vozila, te kupuje vozilo.
Alternative/proširenja	<p>Alternativni tok događaja 1:</p> <ol style="list-style-type: none"> 1. Korisnik pristupa stranici 2. Korisnik otvara stranicu za konfiguraciju automobila <i>Koraci 3. do 7. isti kao koraci toka događaja 1. od 7. do 11.</i> 8. Sistem traži od kupca da se registruje ili prijavi 9. Korisnik vrši registraciju 10. Sistem provjerava dostupnost opcija vozila 11. Sistem traži autentifikaciju korisnika 12. Sistem traži način plaćanja vozila 13. Korisnik unosi svoje detalje plaćanja 14. Sistem obavijesti korisnika o uspješnoj prodaji

Tok događaja 1: Uspješan završetak (Kupovina automobila)

Korisnik	Sistem
1. Korisnik pristupa stranici	
2. Korisnik pritisne na dugme za registraciju	
	3. Sistem otvara stranicu za prijavu
4. Korisnik unosi svoje podatke	

	5. Sistem izvrši prijavu korisnika i vrati ga na početnu stranicu
6. Korisnik otvara stranicu za konfiguraciju novih vozila	
	7. Sistem učitava stranicu i sve trenutne modele
8. Korisnik bira model koji želi konfigurisati	
	9. Sistem otvara novu stranicu sa tim modelom vozila i svim opcijama koje se mogu izabrati.
10. Korisnik vrši odabir svih opcija	
11. Korisnik pritisne na dugme za kupovinu	
	12. Sistem provjerava dostupnost opcija
	13. Sistem traži autentifikaciju korisnika
	14. Sistem traži način plaćanja vozila
15. Korisnik unosi svoje detalje plaćanja	
	16. Sistem obavijesti korisnika o uspješnoj prodaji

Tok događaja 2: Neuspješan završetak (Opcije nisu dostupne)

Korisnik	Sistem
1. Korisnik pristupa stranici	
2. Korisnik pritisne na dugme za registraciju	
	3. Sistem otvara stranicu za prijavu
4. Korisnik unosi svoje podatke	
	5. Sistem izvrši prijavu korisnika i vrati ga na početnu stranicu
6. Korisnik otvara stranicu za konfiguraciju novih vozila	
	7. Sistem otvara stranicu i učitava sve trenutne modele

8. Korisnik bira model koji želi konfigurisati	
	9. Sistem otvara novu stranicu sa tim modelom vozila i svim opcijama koje se mogu izabrati.
10. Korisnik vrši odabir svih opcija	
11. Korisnik pritisne na dugme za kupovinu	
	12. Sistem provjerava dostupnost opcija
	13. Sistem obavijesti korisnika da nisu sve odabранe opcije dostupne

Tok događaja 3: Neuspješan završetak (Neuspješna registracija)

Korisnik	Sistem
1. Korisnik pristupa stranici	
2. Korisnik pritisne na dugme za registraciju	
	3. Sistem otvara stranicu za prijavu
4. Korisnik unosi svoje podatke	
	5. Sistem obavijesti korisnika da registracija nije uspjela

Scenario 3: Zahtjev za servis automobila

Naziv slučaja upotrebe	Zahtjev za servis automobila
Opis slučaja upotrebe	Kupac vrši prijavu za servis svog automobila
Vezani zahtjevi	Nema vezanih zahtjeva
Preduslovi	Nema preduslova
Posljedice – uspješan završetak	Kupac uspješno izvrši prijavu za servis automobila
Posljedice – neuspješan završetak	Kupac nije uspio izvršiti prijavu za servis automobila
Primarni akteri	Kupac
Ostali akteri	/
Glavni tok	Kupac otvara stranicu za prijavu automobila za servis, unosi svoje podatke i bira jedan od dostupnih termina.
Alternative/proširenja	

Tok događaja 1: Uspješan završetak (Prijava na servis)

Korisnik	Sistem
1. Korisnik otvara stranicu za prijavu na servis automobila	
	2. Sistem učitava stranicu
3. Korisnik unosi sve svoje podatke i traži dostupne termine	
	4. Sistem učitava sve dostupne termine
5. Kupac bira jedan od termina	
	6. Sistem obavijesti kupca o uspješnoj rezervaciji

Tok događaja 2: Neuspješan završetak (Neuspješna prijava na servis)

Korisnik	Sistem
1. Korisnik otvara stranicu za prijavu na servis automobila	
	2. Sistem učitava stranicu
3. Korisnik unosi sve svoje podatke i traži dostupne termine	
	4. Sistem učitava sve dostupne termine
	5. Sistem obavijesti kupca da trenutno nema dostupnih termina

Scenario 4: Procjena vrijednosti vozila

Naziv slučaja upotrebe	Prijava za servis automobila
Opis slučaja upotrebe	Zaposlenik vrši procjenu vrijednosti vozila uz pomoć informacija o vozilu i slika vozila
Vezani zahtjevi	Nema vezanih zahtjeva
Preduslovi	Korisnik mora biti prijavljen
Posljedice – uspješan završetak	Zaposlenik uspješno dobio procjenu vozila
Posljedice – neuspješan završetak	Procjena vozila nije uspješna
Primarni akteri	Zaposlenik
Ostali akteri	/
Glavni tok	Zaposlenik ima mogućnost da odredi aproksimaciju vrijednosti nekog vozila tako što unese informacije o vozilu i uploada slike vozila.
Alternative/proširenja	

Tok događaja 1: Uspješan završetak (Dobivena aproksimacija)

Korisnik	Sistem
1. Korisnik otvara stranicu za aproksimaciju vrijednosti vozila	
	2. Sistem učitava stranicu
3. Korisnik unosi sve informacije o vozilu i uploada slike	
	4. Sistem vrši aproksimaciju na osnovu podataka
	5. Sistem obavijesti korisnika o vrijednosti vozila

Tok događaja 2: Neuspješan završetak (Greška u sistemu)

Korisnik	Sistem
1. Korisnik otvara stranicu za aproksimaciju vrijednosti vozila	
	2. Sistem učitava stranicu
3. Korisnik sve informacije o vozilu i uploada slike	
	4. Sistem vrši aproksimaciju na osnovu podataka
	5. Sistem obavijesti korisnika da je došlo do neke greške u sistemu

Scenario 5: Pregled zamjenskih auto dijelova

Naziv slučaja upotrebe	Pregled zamjenskih auto dijelova
Opis slučaja upotrebe	Zaposlenik vrši pregled odgovarajućih zamjenskih dijelova za određeni model vozila
Vezani zahtjevi	Nema vezanih zahtjeva
Preduslovi	Korisnik mora biti prijavljen
Posljedice – uspješan završetak	Zaposlenik pronašao odgovarajući dio
Posljedice – neuspješan završetak	Zaposlenik nije pronašao odgovarajući dio
Primarni akteri	Zaposlenik
Ostali akteri	/
Glavni tok	Zaposlenik ima mogućnost da veoma lagano pronađe neki zamjenski dio za određeno vozilo uz pomoć ovog sistema
Alternative/proširenja	

Tok događaja 1: Uspješan završetak (Pronađen zamjenski dio)

Korisnik	Sistem
1. Korisnik otvara stranicu za pregled zamjenskih auto dijelova	
	2. Sistem učitava stranicu i traži odabir modela i traženog zamjenskog djela
3. Korisnik unesi tražene informacije	
	4. Sistem traži odgovarajući zamjenski dio i prikaže ga
	5. Sistem vrši provjeru dostupnosti dijelova te obavijesti korisnika o stanju dostupnosti

Tok događaja 2: Neuspješan završetak (Nije pronađen zamjenski dio)

Korisnik	Sistem
1. Korisnik otvara stranicu za pregled zamjenskih auto dijelova	
	2. Sistem učitava stranicu i traži odabir modela i traženog zamjenskog djela
3. Korisnik unesi tražene informacije	
	4. Sistem obavijesti korisnika da nije pronađen nijedan odgovarajući zamjenski dio

SOLID principi

1. Princip pojedinačne odgovornosti (Single responsibility Principle)

Svaka klasa treba imati samo jednu ulogu. Ovaj princip zahtijeva da svaka klasa ima samo jednu odgovornost, odnosno da klasa vrši samo jedan tip akcija kako ne bih ovisila o prevelikom broju konkretnih implementacija. Kod nas ovaj princip je za sad zadovoljen jer klase u našem sistemu sadrže samo getere i setere, konstruktore i destruktore. Gledajući dalje kroz program, naše klase trebale bi zadovoljiti ovaj princip.

2. Otvoreno-zatvoreni princip (Open/Closed Principle)

Ovaj princip kaže da bi klase trebali biti otvorene za nadogradnju a zatvorene za modifikaciju. Ovaj princip zahtijeva da klasa koja koristi neku drugu klasu ne treba biti modificirana pri uvođenju novih funkcionalnosti, ili pri potrebi za mijenjanjem druge klase. U našem sistemu klasa koja bi možda imala problem sa ovim principom jeste klasa narudžba koja u jednoj narudžbi može primiti samo jedan automobil, te u slučaju da kupac hoće da kupi više od jednog automobila, ova klasa bi se trebala modificirati da može primiti više od jednog automobila. Također ovo može biti riješeno sa pravljenjem više narudžbi u sklopu jedne kupovine.

3. Liskov princip zamjene (Liskov Substitution Principle)

Svaka osnovna klasa treba biti zamjenjiva svim svojim podtipovima, bez da to utječe na ispravnost rada programa. Ovaj princip zahtijeva da nasljeđivanje bude ispravno implementirano, odnosno da je na svim mjestima na kojima se koristi osnovni objekat moguće skoristiti i izvedeni objekat a da takvo nešto ima smisla. U našem sistemu ovaj princip je zadovoljen jer jedine izvedene klase su klasa NoviAutomobil i PolovniAutomobil koje su izvedene iz klase Automobil. Ova klasa koristi u narudžbi i može se zamjeniti sa jednom ili drugom izvedenom klasom.

4. Princip izoliranja interfejsa (Interface Segregation Principle)

Bolje je imati više specifičnih interfejsa, nego jedan generalizovani. Ovaj princip zahtijeva da i svi interfejsi zadovoljavaju princip S, odnosno da svaki interfejs obavlja samo jednu vrstu akcija. U našem sistemu nemamo niti jedan interfejs pa je ovaj princip zadovoljen.

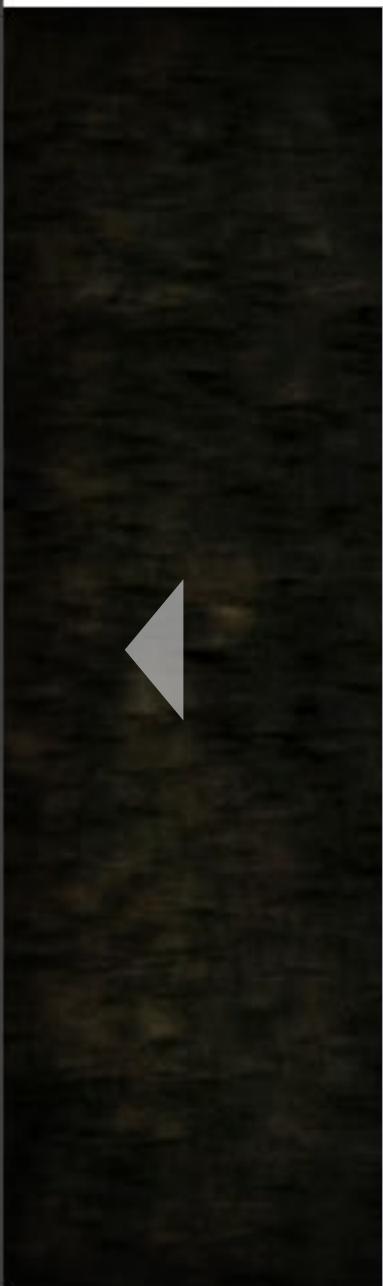
5. Princip inverzije ovisnosti (Dependency Inversion Principle)

Sistem klasa i njegovo funkcionisanje treba ovisiti o apstrakcijama, a ne o konkretnim implementacijama. Ovaj princip zahtijeva da pri nasljeđivanju od strane više klasa bazna klasa uvijek bude apstraktна. Kod nas jedino nasljeđivanje nalazimo kod klase Automobil, koja zadovoljava ovaj princip jer je ona apstraktна.



LEVRES

PRODAJA | KONFIGURATOR | SERVIS |



LEVRES XC60

PRONAĐITE GA
U KONFIGURATORU



LEVRES

NOVI NAČIN KUPOVINE VAŠEG NOVOG VOZILA

"LOREM IPSUM DOLOR SIT AMET, CONSECTETUR ADIPISCING ELIT, SED DO EIUSMOD TEMPOR INCIDIDUNT
UT LABORE ET DOLORE MAGNA ALIQUA. UT ENIM AD MINIM VENIAM, QUIS NOSTRUD EXERCITATION ULLA-
MCO LABORIS NISI UT ALIQUIP EX EA COMMODO CONSEQUAT. DUIS AUTE IRURE DOLOR IN REPREHEN-



PRODAJA

KUPITE NOVO ILI POLOVNO VOZILO TRENUTNO DOSTUPNO NA LAGERU

FILTERI

TIP VOZILA:

- LIMUZINA
- TERENAC
- MALO VOZILO

CIJENA:

OD: DO:

GODIŠTE:

OD: DO:

KILOMETRAŽA:

OD: DO:

SNAGA:

OD: DO:



LEVRES DY-6

GODIŠTE: 2018
KILOMETRAŽA: 134993KM
SNAGA: 186HP

CIJENA: 64 000KM



LEVRES DY-6

GODIŠTE: 2018
KILOMETRAŽA: 134993KM
SNAGA: 186HP

CIJENA: 64 000KM



LEVRES DY-6

GODIŠTE: 2018
KILOMETRAŽA: 134993KM
SNAGA: 186HP

CIJENA: 64 000KM



LEVRES DY-6

GODIŠTE: 2018
KILOMETRAŽA: 134993KM
SNAGA: 186HP

CIJENA: 64 000KM



LEVRES DY-6

GODIŠTE: 2018
KILOMETRAŽA: 134993KM
SNAGA: 186HP

CIJENA: 64 000KM



LEVRES DY-6

GODIŠTE: 2018
KILOMETRAŽA: 134993KM
SNAGA: 186HP

CIJENA: 64 000KM



SERVIS

ZAKAŽITE SERVIS ZA VAŠE VOZILO

DETALJI

IME I PREZIME:

TERMIN SERVISA:

October 2021 >



SUN MON TUE WED THU FRI SAT

1 2

3 4 5 6 7 8 9

10 11 12 13 14 15 16

17 18 19 20 21 22 23

24 25 26 27 28 29 30

31

ADRESA:

BROJ TELEFONA:

EMAIL:

VIN VOZILA:

ZAHTJEV SERVISA:

ZAKAŽI



LEVRES

LEVRES DY-6

DETALJI

GODIŠTE:	2018
KILOMETRAŽA:	153,013KM
SNAGA:	186HP
GORIVO:	BENZIN
TRANSIMISIJA:	MANUELNI
POGON.	ZADNJI
TIP:	MALO AUTO
KUBIKAŽA:	1998CC
PAKET OPREME:	OPTIMUM
CIJENA:	53,000KM

PRODAJA | KONFIGURATOR | SERVIS |



KUPI



PRIJAVA

EMAIL:

PASSWORD:

PRIJAVA

NEMATE KORISNIČKI RAČUN? [REGISTRUJTE SE OVDJE](#)



LEVRES

PRODAJA | KONFIGURATOR | SERVIS |

DOBRODOŠAO, AMIR



PROCIJENI VRIJEDNOST
VOZILA

OTVORI



PREGLEDAJ ZAKAZANE
SERVISE

OTVORI



PRONAĐI ZAMJENSKI
DIO ZA VOZILO

OTVORI



LEVRES

PRODAJA | KONFIGURATOR | SERVIS |

PROCJENA VRIJEDNOSTI VOZILA

ODREDITE APROKSIMACIJU VRIJEDNOSTI VOZILA

DETALJI

MODEL:

GODIŠTE:

KILOMETRAŽA:

SNAGA:

STANJE:

DODATNA OPREMA:

SLIKE:



VRIJEDNOST:

20,000 - 25,000 KM

VRIJEDNOST VOZILA APROKSIMATIVNO
SE NALAZI IZMEĐU DATIH GRANICA
NE GARANTIRA SE TAČNOST

IZRAČUNAJ

PREGLED SERVISA

SERVISI

MJESEČNI PREGLED

October 2021 < >						
SUN	MON	TUE	WED	THU	FRI	SAT
					1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30
31						

DNEVNI PREGLED ZA 22.10:

09:00H

VIN: 22138148901208123

ZAHTJEV: ZAMJENA KOČNICA

11:00H

VIN: 18398102301231239

ZAHTJEV: MALI SERVIS

12:00H

VIN: 01839812038012093

ZAHTJEV: PUNJENJE KLIME

14:00H

VIN: 91823982123546633

ZAHTJEV: POPRAVAK MJENJAČA



ZAMJENSKI DIJELOVI

MODEL VOZILA

PRIMJENA:

- MOTOR
- OVJES
- MJENJAČ
- KAROSERIJA
- UNUTRAŠNOST
- KOZMETIKA
- POTROŠNI MATERIJAL
- DODATNA OPREMA



PREDNJI DISK

MODEL: DY6, XC60
KOLIČINA: 13

CIJENA (BEZ UGRADNJE):
43.35KM



PREDNJI DISK

MODEL: DY6, XC60
KOLIČINA: 13

CIJENA (BEZ UGRADNJE):
43.35KM



PREDNJI DISK

MODEL: DY6, XC60
KOLIČINA: 13

CIJENA (BEZ UGRADNJE):
43.35KM



PREDNJI DISK

MODEL: DY6, XC60
KOLIČINA: 13

CIJENA (BEZ UGRADNJE):
43.35KM



PREDNJI DISK

MODEL: DY6, XC60
KOLIČINA: 13

CIJENA (BEZ UGRADNJE):
43.35KM



PREDNJI DISK

MODEL: DY6, XC60
KOLIČINA: 13

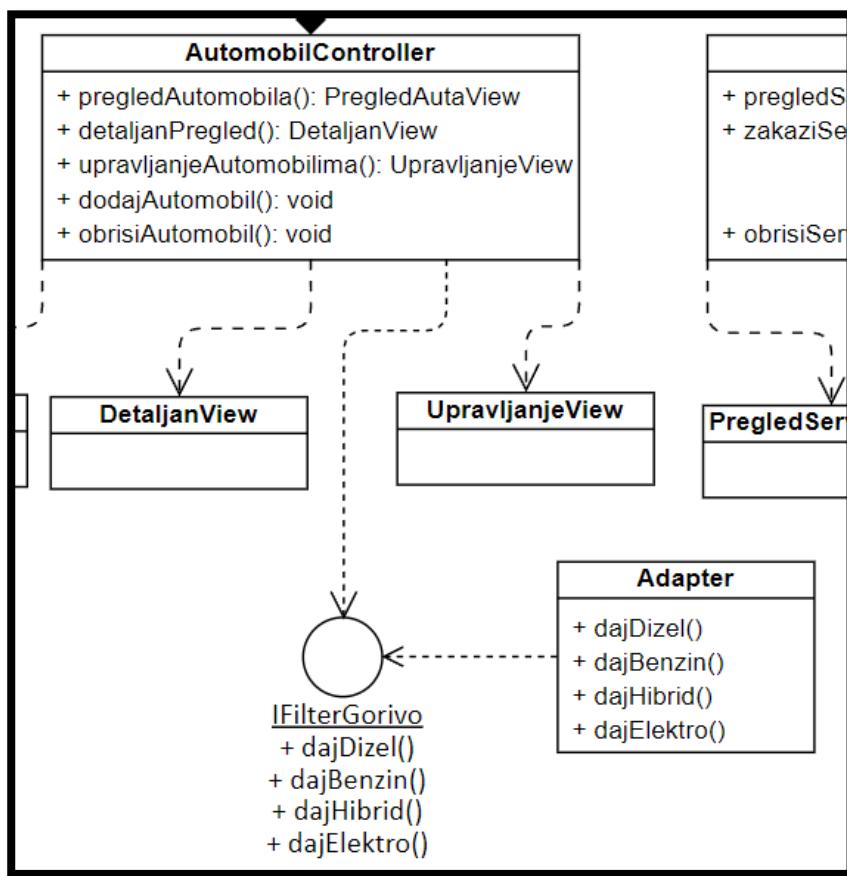
CIJENA (BEZ UGRADNJE):
43.35KM

Strukturni patterni

1. Adapter pattern

Adapter pattern je pattern čija je osnovna namjena da interfejs jedne klase pretvori u neki željeni interfejs kako bi se ona mogla koristiti u situaciji u kojoj bi inače problem predstavljali nekompatibilni interfejsi. Primjenom Adapter paterna dobija se željena funkcionalnost bez izmjena na originalnoj klasi i bez ugrožavanja integriteta cijele aplikacije.

Za našu primjenu, kreirati ćemo novu mogućnost putem koje možemo filtrirati vozila po gorivu koje koristi. Kako i to uradili, kreiramo interfejs **IFilterGorivo**. Taj interfejs će posjedovati metode **dajDizel()**, **dajBenzin()**, **dajHibrid()** i **dajElektro()**. Svaka od metoda vraća odgovarajuća vozila. Te metode će biti implementirane u klasi **Adapter**. Izgled dijagrama nakon implementacije je sljedeći:



2. Facade pattern

Facade pattern se koristi kada sistem ima više identificiranih podsistema pri čemu su apstrakcije i implementacije podistema usko povezane. Osnovna

namjena Facade patterna je da osigura više pogleda visokog nivoa na podsisteme. Ovaj patern sakriva implementaciju podistema od korisnika i pruža pojednostavljeni interfejs putem kojeg korisnik pristupa sistemu.

Ovaj pattern nismo iskoristili jer naš sistem ne sadrži veliki broj podistema čije su implementacije usko vezane.

3. Bridge pattern

Svrha Bridge pattern-a je omogućavanje da se iste operacije primjenjuju nad različitim podklasama. Ovim izbjegavamo nepotrebno duplicitanje koda, odnosno izbjegavamo kreiranje novih metoda za već postojeće funkcionalnosti. Dakle, bridge patern razdvaja pojedinačnu klasu na više zasebnih klasa koje se mogu razvijati odvojeno jedna od druge.

Nismo odlučili implementovati ovaj pattern, ali kada bih ga implementovali koristili bi ga za različit način prikazivanja servisa. Radnik i korisnik na isti način dobivaju pregled servisa, ali razlika je u tome što korisnik samo vidi svoje servise, dok radnik ima uvid svih servisa u sistemu. Dodavanjem klase IServisi sa metodom getServisi() i Bridge klase koja vraća listu servisa u istom formatu i za radnika i za korisnika, mi veoma jednostavno to možemo uvesti.

4. Proxy pattern

Proxy pattern je pattern koji pruža objekat koji se ponaša kao substitute objekat za pravi objekat koji pruža neku uslugu. Proxy objekat prihvata korisničke zahtjeve i obrađuje ih u zavisnosti od prava pristupa korisnika tj. onemogućen je slučajni pristup podacima od strane korisnika koji nema odgovarajuće permisije.

Iako nismo odlučili implementovati ovaj pattern, način na koji smo ga mogli iskoristiti je tako što bismo uveli kontrolu pristupa bazi podataka. Kao jedan primjer zašto bi to radili, možemo reći da hoćemo zabraniti unošenje novih automobila u bazu svim drugim akterima osim administratoru.

5. Flyweight pattern

Flyweight pattern koristi se kako bi se onemogućilo bespotrebno stvaranje velikog broja instanci objekata koji u suštini predstavljaju isti objekat.

Ovaj pattern nećemo koristiti u našem sistemu, pošto nemamo potrebu za kreiranjem nekih "defaultnih", tj. dijeljenih resursa. Ali primjer korištenja takvog patterna može biti sljedeći. Recimo da konfiguracija novih vozila se spašava u bazu podataka, te da za novo kreirane korisnike se odmah spasi neka defaultna konfiguracija, i svaka naknadna promjena konfiguracija se

spašava. Kako bi uštedili na kreiranju te nove konfiguracije, napravili bi interfejs **IKonfiguracija**, koja bi imala metodu **dajKonfiguraciju()**. Taj novi interfejs će koristiti dvije nove klase, **Konfiguracija** koja čuva promijenjenu konfiguraciju korisnika i **DefaultKonfiguracija** koja sadrži tu početnu konfiguraciju koja će se vezati za novo kreirane korisnike.

6. Composite patern

Osnovna namjena Composite paterna je da omogući formiranje strukture drveta pomoću klasa, gdje se objekti (listovi) i kompozicije objekata (korijeni) jednako tretiraju. Ovo zapravo znači da je moguće pozivati metodu koja je zajednička na nivou svih tih klasa.

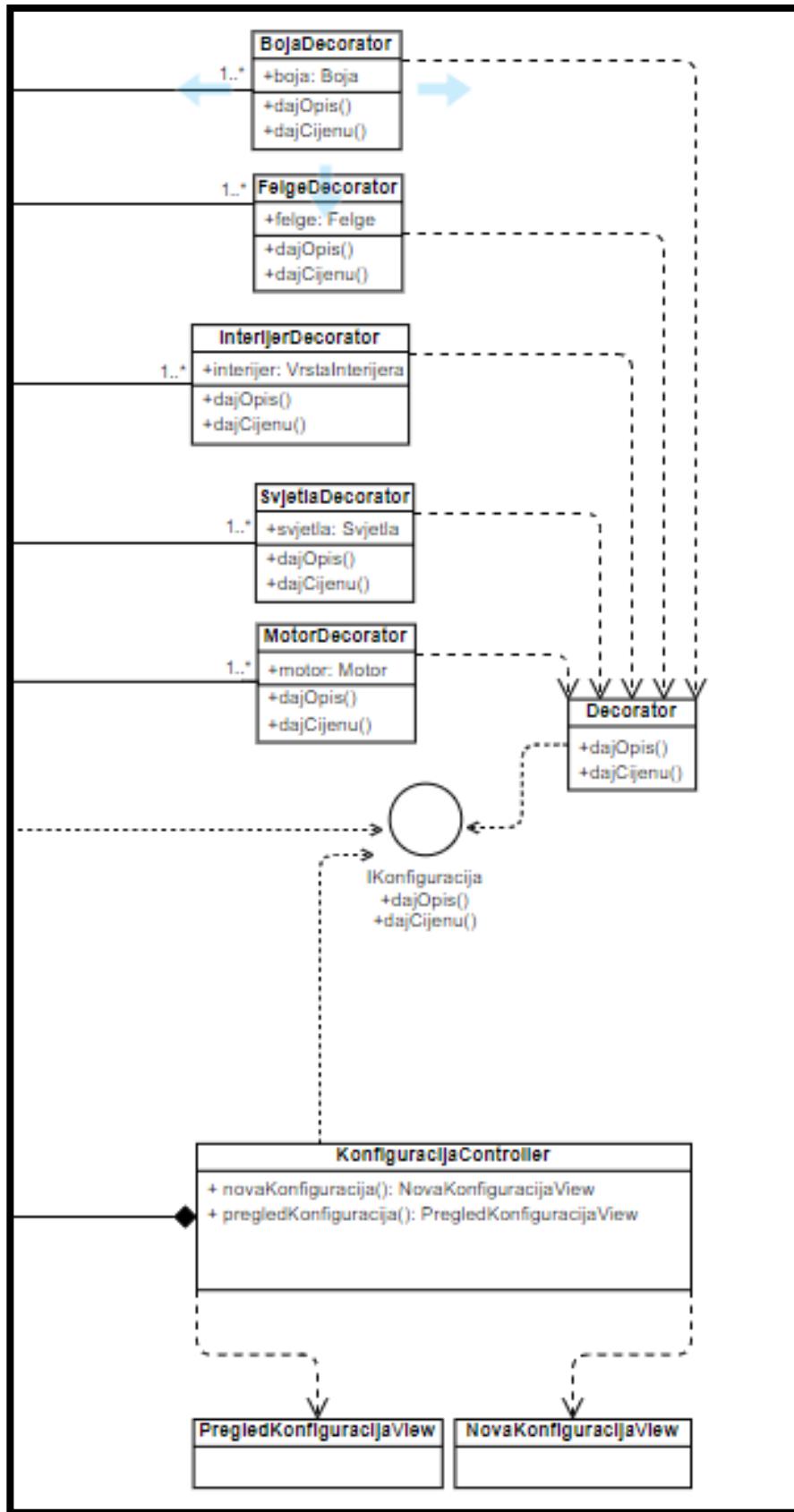
Ovaj obrazac omogućava da se ponašanje izvršava rekursivno preko svih komponenata stabla objekta, bez potrebe da se znaju konkretne klase objekata koji čine stablo.

U našem sistemu composite patern možemo uvesti kada hoćemo rukovati sa korisnicima. Pomoću njega možemo dopustiti administratoru pregled informacija svih korisnika. Na ovaj način bi se stvorila zahtijevana struktura stabla, gdje bi korijen stabla bila klasa **Korisnik**, a listovi bi nam mogli biti **Zaposlenik** i **Kupac**.

7. Decorator patern

Svrha Decorator patern-a je da omogući dinamičko dodavanje novih elemenata i ponašanja (funkcionalnosti) postojećim objektima. Objekat pri tome ne zna da je uradena dekoracija što je veoma korisno za ponovnu upotrebu komponenti softverskog sistema.

U našem slučaju dodali smo decorator patern za konfiguraciju automobila. Uz pomoć dvije opcije koje se nalaze u sklopu interfejsa **dajOpis()** i **dajCijenu()**, naš program može da dinamički mijenja opis i cijenu konfiguracije. Opis i cijena se mijenja pomoću **Decoratora** i njegovih **ConcreteDecorator** klase, kao što su **BojaDecorator**, **FelgeDecorator**, **InterijerDecorator**, **SvetlaDecorator** i **MotorDecorator**, gdje je svaka od njih vezana za enumeratorske klase, gdje odabirom neke vrijednosti iz istoimene enumeratorske klase dobije određenu cijenu za proizvod.



KREACIJSKI PATERNI

1. Singleton pattern

Uloga Singleton patterna je da osigura da se klasa može instancirati samo jednom i da osigura globalni pristup kreiranoj instanci klase. Postoji više objekata koje je potrebno samo jednom instancirati i nad kojim je potrebna jedinstvena kontrola pristupa.

Instanciranje više nego jednom može prouzrokovati probleme kao što su nekorektno ponašanje programa, neadekvatno korištenje resursa ili nekonzistentan rezultat.

U našem sistemu mi smo napravili Inventar klasu. Ona omogućava administratoru da doda i ukloni automobile po volji. Ova klasa se instancira samo jednom i sve promjene koje se naprave uz pomoć ove klase, odrazit će se na cijeli sistem.

Inventar
- <code>inventarIstanca: Inventar</code>
<code><<create>> + Inventar()</code>
<code>+ dajInstancu(): Inventar</code>
<code>+ dodajAuto(auto: Automobil) : void</code>
<code>+ obrisiAuto(auto: Automobil) : void</code>

2. Prototype pattern

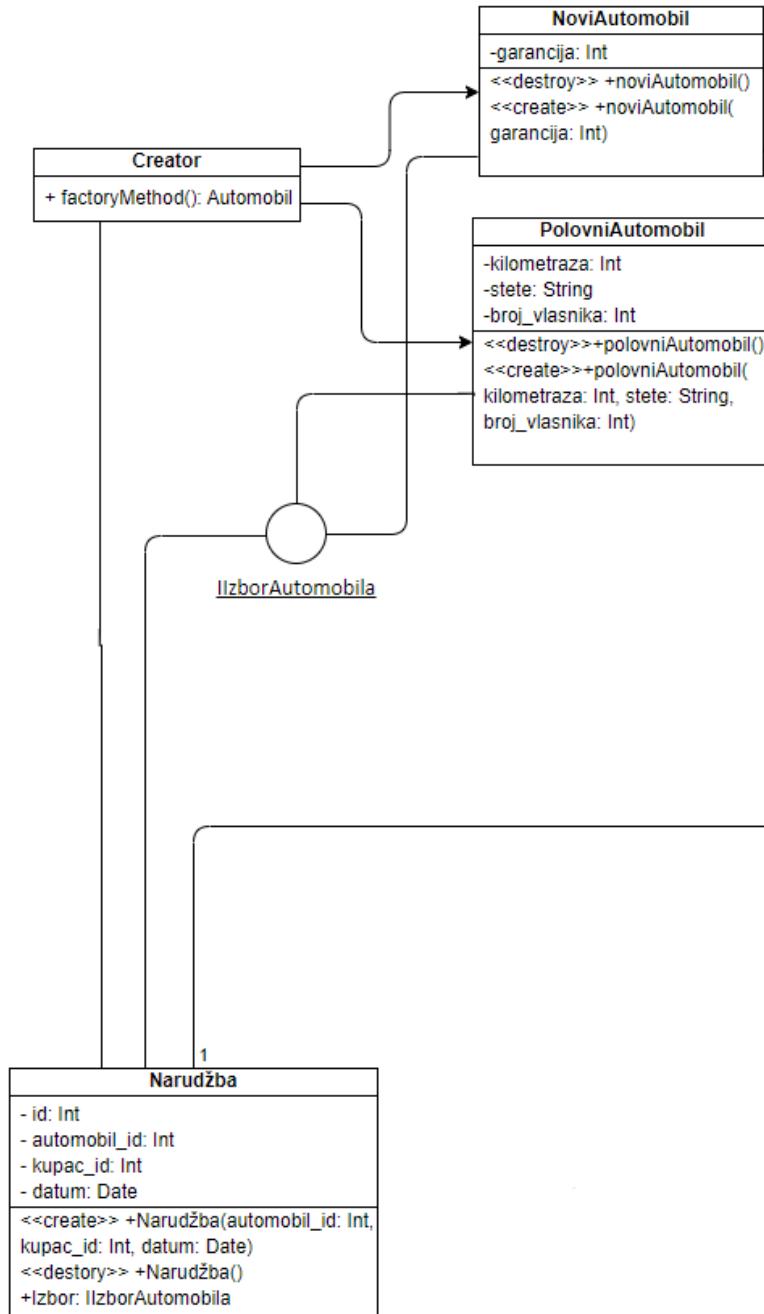
Uloga Prototype patterna je da kreira nove objekte klonirajući jednu od postojećih prototip instanci (postojeći objekat). Ako je trošak kreiranja novog objekta velik i kreiranje objekta je resursno zahtjevno tada se vrši kloniranje već postojećeg objekata. Dakle ovaj šablon omogućava jednostavnije kreiranje novih instanci klasa kod kojih je veliki broj atributa identičan za većinu instanci.

Naš sistem se već sastoji od klasa kod kojih je većina atributa za većinu instanci različita, čak u nekim slučajevima i svi su različiti, jer su u pitanju klase koje modeliraju specifične objekte, korisnike, narudžbe i zahtjeve, pa nema potrebe za kloniranjem.

3. Factory Method pattern

Uloga Factory Method patterna je da omogući kreiranje objekata na način da podklase odluče koju klasu instancirati. Različite podklase mogu na različite načine implementirati interfejs. Factory Method instancira odgovarajuću podklasu (izvedenu klasu) preko posebne metode na osnovu informacije od strane klijenta ili na osnovu tekućeg stanja.

U našem sistemu ovaj patern možemo implementirati kod naše klase Narudžba. Pomoću ovog paterna možemo kontrolisati mjenjanje stanja baze s obzirom da li naš kupac kupuje polovno ili novo auto.



4. Abstract Factory pattern

Abstract Factory pattern omogućava da se kreiraju familije povezanih objekata/produkata. Na osnovu apstraktne familije produkata kreiraju se konkretne fabrike (factories) produkata različitih tipova i različitih kombinacija.

U našem sistemu ovaj patern nije moguće promjeniti jer naš sistem ne posjeduje više različitih objekata od kojih se može kreirati familija.

5. Builder pattern

Uloga Builder patterna je odvajanje specifikacije kompleksnih objekata od njihove stvarne konstrukcije. Isti konstrukcijski proces može kreirati različite reprezentacije.

U našem sistemu već imamo implementiran ovaj patern kod naše klase Konfiguracija. Kako bi upotpunili ovaj patern mogli bismo nadodati interfejs IBuilder i Director klasu, ali to nam se ne čini nešto veoma potrebno našem sistemu.

PATERNI PONAŠANJA

1. Strategy pattern

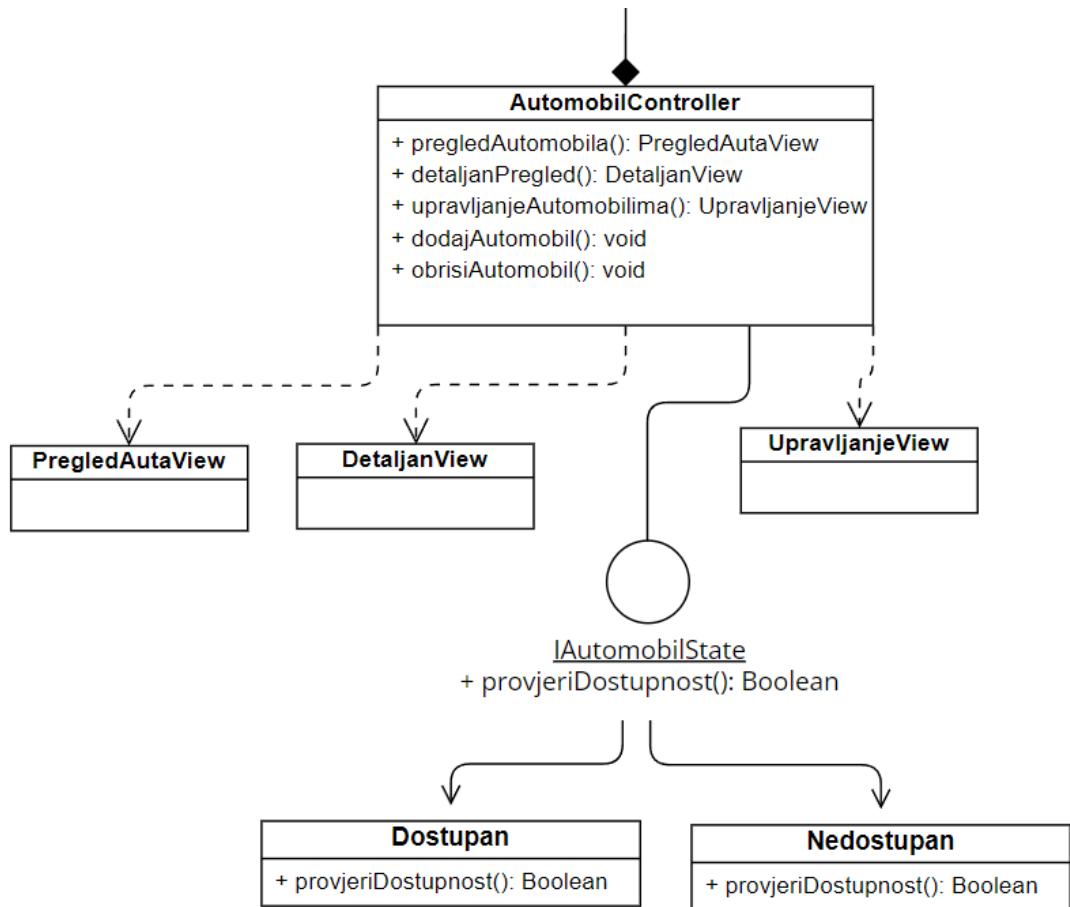
Strategy pattern definiše porodicu algoritama, enkapsulira svaki od njih i omogućava njihovu zamjenu. Klijenti koji koriste algoritme mogu to raditi nezavisno od konkretne implementacije algoritma. Ovo omogućava fleksibilnost i proširivost, jer se novi algoritmi mogu lako dodati bez menjanja postojećeg koda.

Strategy pattern bi mogli iskoristiti kod klase Konfiguracija gdje možemo imati različite strategije za izračunavanje konačne cijene konfiguracije, poput standardne cijene, cijene bez PDV-a, cijene sa popustom i sl. Kako bi to odradili napravili bi novu klasu Popust koja bi bila naša Context klasa, i također bi imali interfejs Strategy iz kojeg bi naslijedili te različite cijene.

2. State pattern

State pattern omogućava objektu da promjeni svoje ponašanje kada mu se interno stanje promeni. Svako stanje je enkapsulirano kao posebna klasa koja definiše ponašanje objekta u tom stanju. Na ovaj način, objekti mogu dinamički mjenjati klase kako bi reflektovali promjenu stanja.

Ovaj pattern ćemo implementirati za klasu AutomobilController tako što ćemo dodati interface IAutomobilState koji će govoriti o dostupnosti vozila. Iz interfacea ćemo izvesti klase Dostupan i Nedostupan. Izgled u dijagramu klasa je na slici ispod.



3. Template method pattern

Template method pattern definiše skelet algoritma u okviru metode, ostavljajući određene korake podklasama da ih implementiraju. Ovo omogućava ponovnu upotrebu osnovne strukture algoritma dok podklase mogu prilagoditi specifične korake. Na ovaj način se postiže kontrola toka algoritma uz mogućnost prilagođavanja.

Klasa Servis može definisati šablon metode za proces servisiranja. Konkretnе klase kao što su MaliServis i VelikiServis mogu implementirati specifične korake unutar tog šablona. Na primer, koraci kao što su proveravanje ulja, zamena filtera, itd. mogu biti zajednički, dok konkretne akcije mogu varirati zavisno od tipa servisa.

4. Observer pattern

Observer pattern definiše zavisnost jedan-na-više između objekata, tako da kada se stanje jednog objekta promjeni, svi zavisni objekti budu obavješteni i automatski ažurirani. Ovo omogućava reaktivno

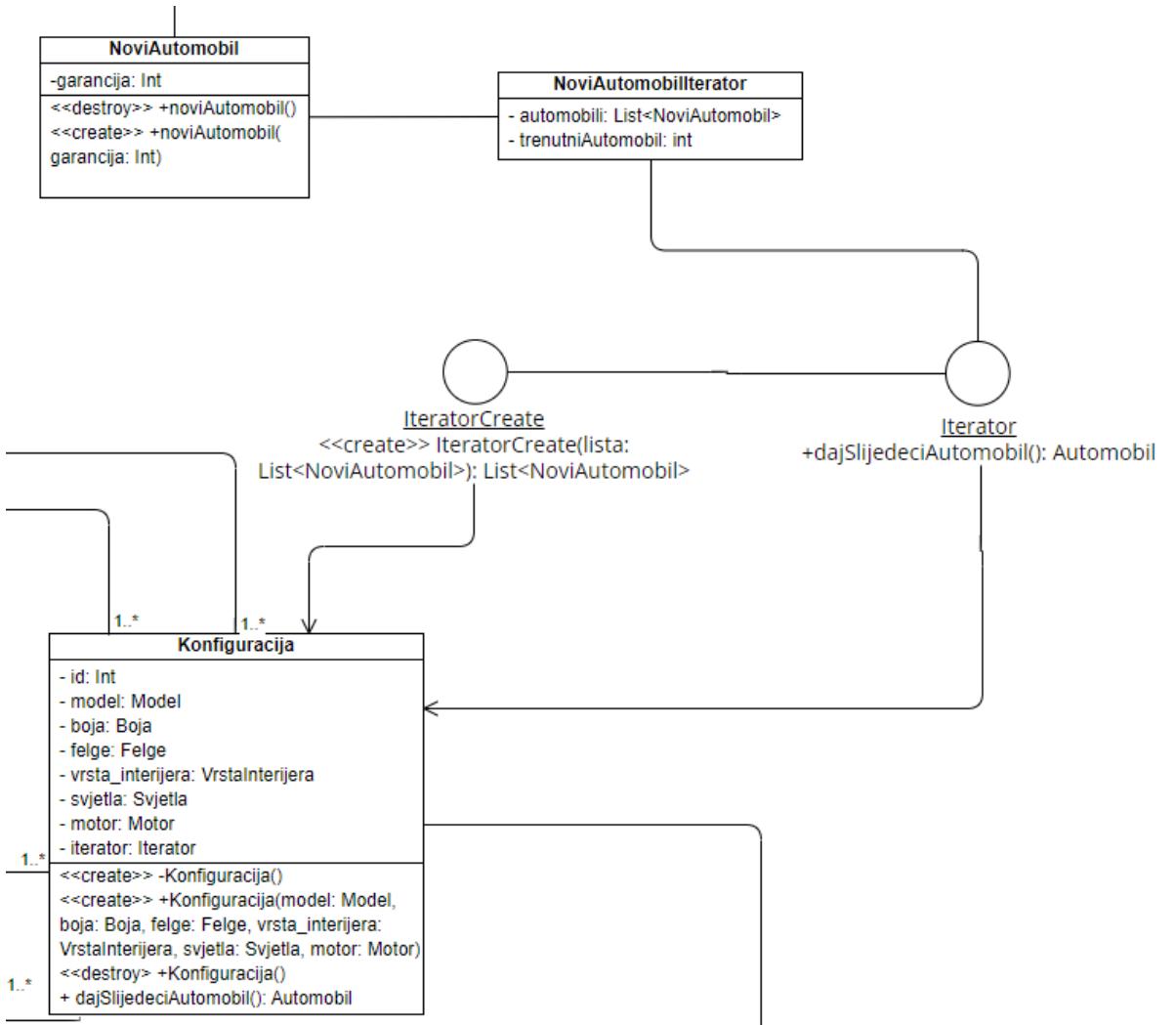
programiranje gdje promjene u jednom dijelu sistema propagiraju kroz sistem.

U sistemu za upravljanje narudžbama automobila, Observer Pattern se koristi za obaveštavanje korisnika o promenama statusa njihovih narudžbi. Kada korisnik kreira narudžbu, ta narudžba postaje subjekt sa listom posmatrača (korisnika). Kada se status narudžbe promeni, narudžba obaveštava sve posmatrače o novom statusu. Na primer, ako se status narudžbe promeni iz "u obradi" u "poslano", korisnici će automatski biti informisani o toj promeni. Ovaj pattern omogućava korisnicima da uvek budu ažurirani o stanju svojih narudžbi bez potrebe za ručnim proveravanjem, čime se poboljšava korisničko iskustvo i komunikacija u sistemu.

5. Iterator pattern

Iterator pattern pruža način za sekvenčijalno pristupanje elementima agregatnog objekta bez izlaganja njegove osnovne reprezentacije. Klijent može da koristi iterator za traversiranje kolekcije objekata uniformno.

Ovaj pattern ćemo implementirati na sljedeći način, tako što ćemo imati iterator za klasu NoviAutomobil, i tim iteratorom će moći iterirati klasa Konfiguracija. Kako bi ovo implementirali, potrebna nam je klasa NoviAutomobilIterator koja je izvedena, treba nam interface IteratorCreate i Iterator, i trebamo dodati atribut iterator u klasi Konfiguracija. U našem dijagramu to izgleda ovako.



6. Chain of responsibility pattern

Chain of responsibility pattern omogućava niz objekata koji mogu sukcesivno pregledati i obraditi zahtjev. Svaki objekat u lancu ima mogućnost da obradi zahtjev ili da ga proslijedi sljedećem objektu u nizu. Na taj način se izbjegava čvrsta povezanost pošiljaoca i primaoca zahtjeva.

U kontekstu obrade narudžbi, različite faze obrade (npr. validacija, provjera dostupnosti, finalizacija) mogu biti predstavljene kao lanci odgovornosti. Kada se narudžba kreira, ona prolazi kroz različite obradivače. Prvi obradivač može biti zadužen za validaciju podataka, drugi za provjeru dostupnosti automobila, a treći za finalizaciju narudžbe.

7. Mediator pattern

Mediator pattern obezbjeđuje centralizovani objekat koji upravlja komunikacijom između skupa objekata, smanjujući njihovu međusobnu zavisnost. Na ovaj način se pojednostavljuje komunikacija i smanjuje kompleksnost sistema.

U kontekstu narudžbi i servisa, posrednik može koordinirati komunikaciju između različitih komponenti sistema. Posrednik može koordinirati interakcije između narudžbi i servisa. Na primer, kada se narudžba kreira, posrednik može obavestiti servis da je novi automobil naručen i da treba zakazati inicialni servis.

