UNIVERZITET U BEOGRADU
FAKULTET ORGANIZACIONIH NAUKA

ZAVRŠNI RAD

Razvoj softverskog sistema za praćenje i kontrolu resursa avionskih dijelova u sistemu održavanja vazduhoplovstva korišćenjem .NET tehnologija

Mentor

Prof. Dr Saša Lazarević,

Student

Matija Čolaković 132/19

Beograd, 2023. godina

SADRŽAJ:

1 UVOD

U savremenom vazduhoplovnom sektoru, održavanje avionskih dijelova igra ključnu ulogu u osiguravanju sigurnosti letenja, optimalne efikasnosti i smanjenju neplaniranih zastoja u avio-saobraćaju. Sa rastućim zahtjevima za bezbjednošću, pouzdanošću i efikasnošću vazduhoplovnih sistema, organizacije za održavanje vazduhoplovstva suočavaju se s izazovom upravljanja i praćenja resursa avionskih dijelova.

U ovom kontekstu, kroz pažljivo istraživanje i analizu potreba i zahtjeva industrije vazduhoplovstva, razvijaće se sofisticirani softverski sistem koji će omogućiti korisnicima da u realnom vremenu prate stanje zaliha, predviđaju potrebu za rezervnim dijelovima, te planiraju održavanje i zamjenu dijelova unaprijed, kako bi se smanjili neplanirani zastoji i maksimizirala raspoloživost letjelica.

Kroz ovaj rad, pružiće se detaljan pregled arhitekture, dizajna i implementacije razvijenog softverskog sistema. Takođe će se izvršiti testiranje sistema kako bi se potvrdila njegova funkcionalnost, performanse i bezbjednost, te će se analizirati rezultati kako bi se potvrdila uspješnost sistema u rješavanju identifikovanih problema i izazova.

U ovom radu će se koristiti Microsoft SQL Server Management Studio (SMSS) kao alat koji pruža grafički korisnički interfejs za upravljanje i administraciju Microsoft SQL baze podataka. Microsoft SQL predstavlja centralnu bazu podataka za skladištenje informacija o avionskim dijelovima, stanju zaliha, održavanju i drugim relevantnim podacima. Ovo će omogućiti pouzdanu i sigurnu organizaciju i upravljanje podacima, kao i izvođenje kompleksnih upita i analiza.

Za razvoj softverskog sistema, koristiće se Microsoft Visual Studio, pružajući bogate alate i resurse za implementaciju. Glavni programski jezik koji će biti korišćen za razvoj sistema je C#, omogućavajući objektno-orijentisanu i efikasnu implementaciju funkcionalnosti. Više informacija nalazi se u poglavlju 3.

Kako bismo osigurali integritet i bezbjednost podataka, primijenit ćemo zaključavanje podataka, što će omogućiti sigurno upravljanje podacima i izbjegavanje konflikata prilikom paralelnog pristupa. Više informacija nalazi se u poglavlju 3.

Kao ključan alat za generisanje izvještaja i analiza, koristićemo Crystal Reports, omogućavajući korisnicima vizualizaciju informacija o statusu avionskih dijelova. Više informacija nalazi se u poglavlju 3.

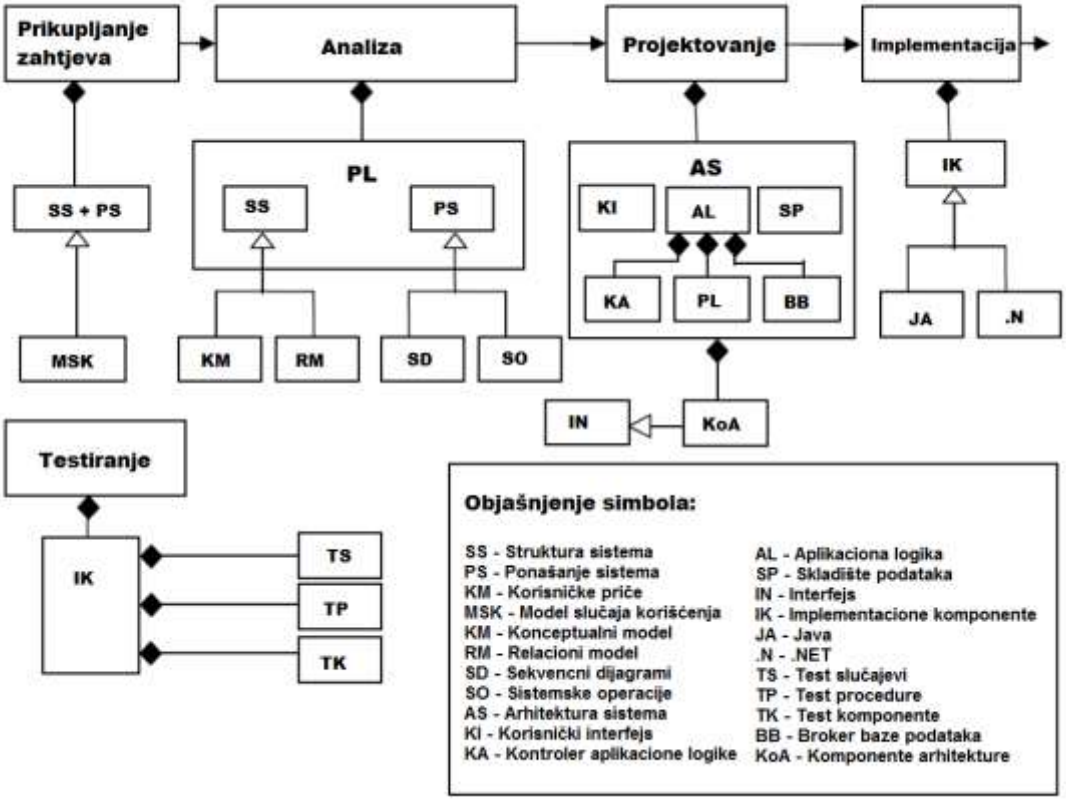
Uz sve ove tehnologije, primijenit ćemo uprošćenu Larmanovu metodu kao agilan pristup razvoju softverskog sistema. Više informacija nalazi se u poglavljima 2, 4 i 5.

Kombinacija ovih tehnologija i alata će omogućiti razvoj sofisticiranog i efikasnog softverskog sistema za praćenje i kontrolu resursa avionskih dijelova u sistemu održavanja vazduhoplovstva. Očekuje se da će ovaj sistem unaprijediti operacije održavanja, smanjiti troškove i poboljšati bezbjednost i pouzdanost u vazduhoplovnom sektoru.

2 UPROŠĆENA LARMANOVA METODA

Prema uprošćenoj Larmanovoj metodi razvoja softvera, životni ciklus softverskog sistema se sastoji iz sledećih pet faza:

- Prikupljanje korisničkih zahtjeva;
- Analiza;
- Projektovanje;
- Implementacija;
- Testiranje.



Slika 1: Razvoj softverskog sistema korišćenjem Larmanove metode[1]

[1] Siniša Vlajić, Projektovanje softvera skripta, Beograd,2020

2.1 PRIKUPLJANJE KORISNIČKIH ZAHTJEVA

U fazi prikupljanja zahtjeva se definišu svojstva i uslovi koje softverski sistem treba da zadovolji. Zahtjevi se mogu podijeliti kao funkcionalni i nefunkcionalni. Funkcionalni zahtjevi definišu zahtijevane funkcije sistema, dok nefunkcionalni zahtjevi definišu sve ostale zahtjeve.

Uopšteno gledano prema uprošćenoj Larmanovoj metodi, zahtjev je svojstvo ili uslov koji neki sistem mora da zadovolji

Strukturu modela slučaja korišćenja čine slučajevi korišćenja (SK), aktori u slučaju korišćenja i veze između aktora i slučajeva korišćenja.

Pravila modela slučaja korišćenja jesu da jedan model može imati više slučajeva korišćenja, više aktora i više veza između slučajeva korišćenja i aktora; jedan slučaj korišćenja može imati više veza sa aktorima; jedan aktor može imati više veza sa slučajem korišćenja, dok veza postoji između tačno jednog para aktora i slučaja korišćenja.

Bilo koji slučaj korišćenja opisan je skupom scenarija, odnosno skupom mogućih korišćenja sistema od strane aktora. Svaki slučaj korišćenja ima jedan glavni scenario i više alternativnih. U svakom scenariju, aktor poziva jednom ili više puta sistemske operacije softverskog sistema.

Zahtjevi se opisuju pomoću UML dijagrama slučaja korišćenja. Tekstualni opis SK ima sledeću strukturu:

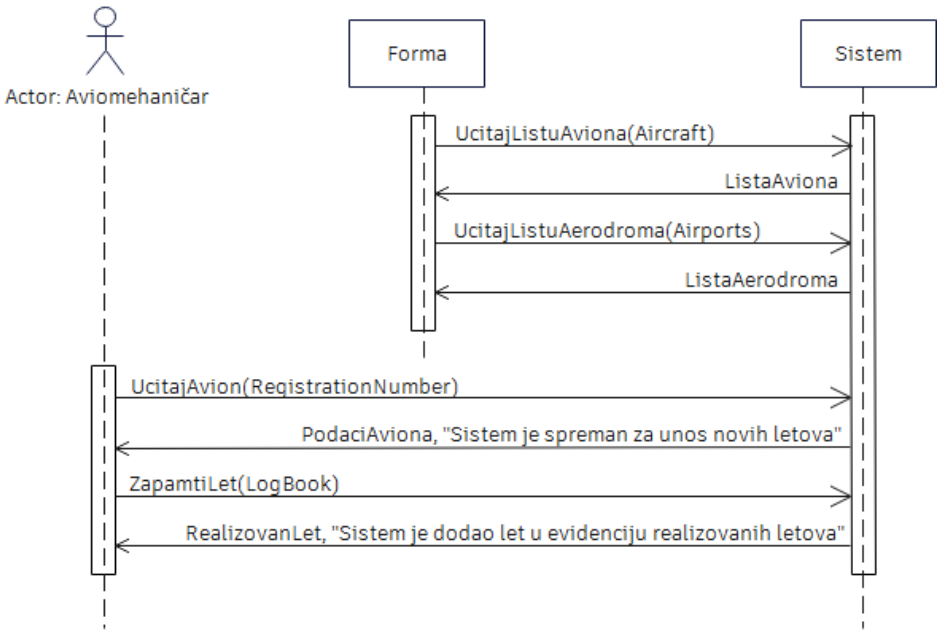
- Naziv SK;
- Aktori SK;
- Učesnici SK;
- Preduslovi neophodni da se SK izvrši;
- Osnovni scenario izvršenja SK;
- Postuslovi neophodni za potvrdu izvršenja SK;
- Alternativni scenariji izvršenja SK;
- Specijalni zahtjevi.

2.2 ANALIZA

Faza analize opisuje poslovnu logiku softverskog sistema, koja se sastoji iz strukture sistema i ponašanja sistema. Ponašanje sistema se opisuje preko sistemskih dijagrama sekvenci i sistemskih operacija, dok se struktura sistema opisuje preko konceptualnih i relacionih modela.

2.2.1 Ponašanje softverskog sistema

Dijagram sekvenci prikazuje događaje u određenom redosledu, koji uspostavljaju interakciju između aktora i softverskog sistema. Za svaki scenario SK, pravi se sistemski dijagram sekvenci.



Slika 2: Primjer dijagrama sekvenci

Sistemska operacija, kao i dijagram sekvenci, opisuje ponašanje sistema. Svaka sistemska operacija ima svoj potpis, koji sadrži ime metode i opciono ulazne i/ili izlazne argumente. Za svaku sistemsku operaciju prave se ugovori koji opisuju šta sama sistemska operacija radi, bez podrobnog objašnjenja kako radi. Ugovor se sastoji iz operacije, veze sa SK, uslovima neophodnim pred izvršenje sistemske operacije kao i uslovima koji moraju biti zadovoljeni nakon izvršenja same operacije.

2.2.2 Struktura softverskog sistema

Struktura sistema opisuje se preko konceptualnog i relacionog modela. Konceptualni model sadrži konceptualne klase i veza - asocijacija između konceptualnih klasa. Konceptualne klase su osobine koje opisuju samu strukturu softverskog sistema, i treba ih razlikovati od softverskih klasa, dok svaki kraj asocijacije ima ulogu konceptualne klase koja učestvuje u asocijaciji.

Slika 3: Primjer konceptualnog modela[2]

Iz konceptualnog modela može se napraviti relacioni model, koji daje osnovu za projektovanje relacione baze podataka. Na primer, relacioni model na osnovu prethodnog konceptualnog modela bi izgledao ovako:

Racun(BrojRacuna, NazivPartnera,UkupnaVrednost,Obradjen,Storniran)
StavkaRacuna(BrojRacuna,RB,SifraProizvoda,Kolicina,ProdajnaCena,ProdajnaVrednost)

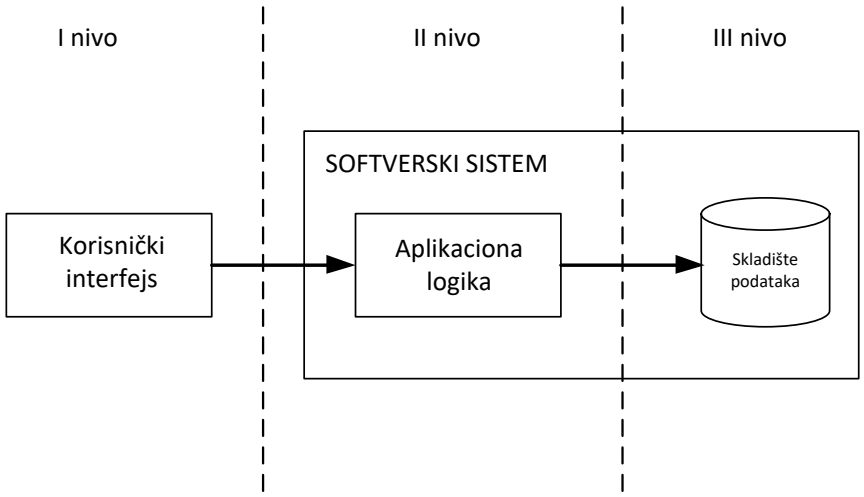
[2] Siniša Vlajić, Projektovanje softvera skripta, Beograd,2020

2.3 PROJEKTOVANJE

Faza projektovanja opisuje fizičku strukturu i ponašanje softverskog sistema. Projektovanje arhitekture softverskog sistema obuhvata projektovanje korisničkog interfejsa (projektovanje kontrolera korisničkog interfejsa i ekranskih formi), aplikacione logike (projektovanje kontrolera aplikacione logike, sistemske operacije) i skladišta podataka (broker baze podataka). Arhitektura sistema je tronivovska i sastoji se od sledećih nivoa:

- Korisnički interfejs;
- Aplikaciona logika;
- Skladište podataka.

Nivo korisničkog interfejsa ja na strani klijenta, dok su aplikaciona logika i skladište na strani servera.



Slika 4: Arhitektura softverskog sistema

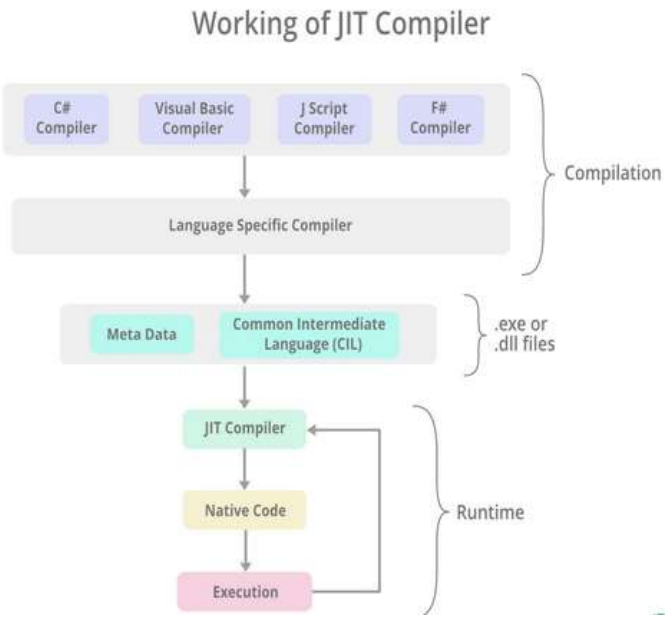
2.4 IMPLEMENTACIJA I TESTIRANJE

Nakon projektovanja, poslednje dve faze razvoja softverskog sistema su implementacija i testiranje. U fazi implementacije, vrši se kodiranje softverskog sistema u određenom programskom jeziku, koji predstavlja implementacionu komponentu. Proces razvoja softverskog sistema objedinjuje se njegovim testiranjem, koje se takođe može podeliti u nekoliko nezavisnih jedinica. Testiranje se obično deli na tri tipa testova: test slučajeva, test procedure i test komponente.

3 .NET TEHNOLOGIJE

Za potrebe implementacije softverskog sistema korišćen je programski jezik C# korišćenjem .NET tehnologija. Prva verzija programskog jezika C# izdata je 2000. godine, kao sastavni dio Majkrosoftovog razvojnog okruženja .NET Framework 1.0. .NET je platforma za razvoj aplikacija koja pruža programerima sveobuhvatan okvir za izradu različitih vrsta aplikacija, uključujući desktop aplikacije, web aplikacije, mobilne aplikacije i servise u cloud-u. Glavne komponente .NET okvira su: Common Language Runtime (CLR), .NET Class Library i Visual Studio IDE.

Svi programi napisani u nekom od .NET jezika prije izvršenja prevode se u zaseban jezik nižeg nivoa. To je CIL (Common Intermediate Language), odnosno zajednički međujezik, koji se često označava i kao IL. CLR (Common Language Runtime) sistem, izvršno okruženje .NET platforme, koristi isključivo IL programski kod. Pošto je dizajn svih .NET jezika zasnovan na IL jeziku, jasno je da između njih postoje krupne sličnosti. Odatle i potiče velika sličnost u opcijama i performansama između jezika VB I C#. Tačnije, stepen kompatibilnosti ovih jezika je takav da web stranica napisana u C# može koristiti VB komponente na isti način kao i one komponente napisane u C#, i obratno.



Slika 5: Just in time compiler

.NET Framework dodatno formalizuje ovu kompatibilnost putem tzv. Common Language Specifikacije (CLS). CLS je u osnovi ugovor koji, ukoliko se poštuje, garantuje da će komponenta napisana u jednom .NET jeziku raditi i sa ostalim jezicima. Deo specifikacije je i tzv. Common Type System (CTS), zajednički sistem tipova, koji definiše pravila za tipove podataka kao što su stringovi, brojevi i nizovi - tipovi koji su zajednički za sve .NET jezike. Pored toga, CLS definše I objektno-orijentisane elemente kao što su klase, metodi, događaji i drugi. .NET programeri u najvećem broju slučajeva uopšte ne moraju voditi računa o tome kako CLS funkcioniše, bez obzira što se u svakodnevnom radu oslanjaju na ovu specifikaciju.

Svaka EXE ili DLL datoteka koju kreirate u nekom od .NET jezika sadrži IL kod. To su upravo datoteke koje postavljate na druge računare. C# kao objektno-orijentisan

programski jezik poštuje pravilo da se jedna klasa nalazi u jednom fajlu. Izvorni kod C# programskog jezika se čuva u fajlovima pod ekstenzijom .cs.

Za razvoj desktop aplikacija s windows formama koristimo:

1. Windows Forms arhitekturu
2. Kreiranje korisničkog interfejsa
3. Manipulaciju podacima
4. Obradu događaja
5. Debagovanje i testiranje aplikacija

3.1 KONCEPTI OBJEKTNO-ORIJENTISANOG PROGRAMIRANJA I C# PROGRAMSKOG JEZIKA

Objektno-orijentisani programi zasnovani su na konceptu objekata iz stvarnog svijeta. Kada pogledamo oko sebe, sve što vidimo predstavlja neki objekat. Objekti iz stvarnog svijeta dijele dvije karakteristike: stanje i ponašanja. Takođe, može se reći da i neki objekat sadrži neki drugi objekat.

Klasa je struktura podataka koju treba posmatrati kao novi tip. Objekat je instanca klase. Objekat u objektno-orijentisanom programiranju se definiše kao entitet koji je sposoban da čuva svoja stanja i koji okolini stavlja na raspolagaje skup operacija preko kojih se tim stanjima pristupa. Objekat karakterišu:

- identitet (omogudava razlikovanje objekata među sobom);
- ponašanje - dinamički aspekt objekta (definiše se metodama koje sadrži objekat);
- stanje - statički aspekt objekta (definiše se podacima objekta).

Stanja se čuvaju u poljima objekta, dok se ponašanja realizuju preko metoda objekta.

3.1.1 Tipovi podataka

Tip promjenljive definiše skup dozvoljenih vrijednosti koje promjenljiva može da uzima i skup operacija za manipulisanje vrijednostima datog tipa i određuje veličinu memorije koja će biti alocirana. Jezici obično uključuju predefinisane tipove (boolean, integer, float, character), ali omogućavaju programeru da definiše i nove tipove – korisničke definisane tipove.

Common Type System (CTS) se može klasifikovati na različite načine. Jedna podjela je na osnovu toga da li su tipovi ugrađeni u jezik ili su korisnički definisani. Korisnički definisani tipovi su tipovi koji nijesu ponuđeni od jezika i dešinišu ih programeri. Drugi način posmatranja tipova je da li su oni vrijednosni ili referentni. Kod vrijednosnog tipa promenljive sadrže podatke, dok kod referentnog tipa promenljive ne sadrže podatke već reference ka tim podacima.

CTS (Common Type System):

- vrijednosni tip
    - built-in (Primitivni tipovi su: cjelobrojni, realni, logički i znakovni tip)
        - Cjelobrojni tip (**byte, sbyte** - 8 bita, **short, ushort** - 16 bita, **int, uint** - 32 bita, **long, ulong** - 64 bita) - Programski jezici podražavaju različite dužine cijelih brojeva i u nekim programskim jezicima se mogu označiti kao **unsigned** ili **signed.** U C#-u **byte** ima opseg vrijednost od 0 do 255, dok **sbyte** ima opseg vrijednost od -128 do 127. **short** ima opseg vrijednosti od -32,768 do 32,767 dok **ushort** ima opseg vrijednosti od 0 do 65,535. **int** ima opseg vrijednost od -2,147,483,648 do 2,147,483,647, dok **uint** ima opseg vrijednost od 0 do 4294967296. **long** ima opseg vrijednosti od - 2^{63} do 2^{63} - 1, dok ulong ima opseg vrijednosti od 0 do 2^{64}. Od verzije

C# 9.0 uvedena su još dva tipa **nint** (native int) i **nuint** (unsigned native int). Ova dva tipa imaju veličinu zavisno od arhitekture računara (ako je 32-bitna arhitektura imaju veličinu 32 bita, dok kod 64-bitne arhitekture imaju 64 bita).

- Realni tip (**float** - 32 bita, **double** - 64 bita, **decimal** - 128 bita)
- Logički tip (**bool** - 1 bajt**,** ima vrijednosti **True** ili **False**, u C ima vrijednosti **0**, **1**, **True**, **False**)
- Znakovni tip (**char** - 16 bita (UTF-16))

- user-defined (**enum**, **struct**)

- referentni tip
  - built-in (**string**, **object, dynamic**)
  - user-defined (**class**, …)
- pointeri (postoje u C# i ne koriste se mnogo)

3.1.2 Koncepti OOP

Postoje dvije strane svakog objekta, ono šta radi(što je obično poznato) i način na koji radi(što je obično nepoznato). Organizovanjem koda u objekte postižu se mnogobrojne pogodnosti. Koncepti objektno-orijentisanog programiranja su:

- učaurenje;
- nasleđivanje;
- polimorfizam.

Učaurenje je koncept OOP kojim se „sakrivaju " detalji implementacije objekta. Postoje dva bitna aspekta učaurenja:

- objedinjavanje podataka i funkcija u jedinstven entitet (klasa);
- kontrola mogućnosti pristupa članovima entiteta (modifikatori pristupa).

Direktan pristup podacima je i nepoželjan i nepotreban. Uvođenjem modifikatora pristupa omogućava se razdvajanje klase na javni dio i privatni dio. Preporuka je da svojstva budu u privatnom dijelu, a metode u javnom dijelu. U C# postoji 6 modifikatora pristupa:

- private – vidljivost samo u sopstvenoj klasi
- private protected – vidljivost u svojoj klasi i svim naslijeđenim klasama koje su unutar istog asemblija
- internal – vidljivost u svojoj klasi i svom asembliju
- protected – vidljivost u svojoj klasi i svim naslijeđenim klasama
- protected internal – vidljivost u svojoj klasi, svim naslijeđenim klasama i unutar svog asemblija
- public – vidljivost unutar čitavog programa

Operaciju izvođenja posebnih klasa iz opšte (generalizovane) klase zovemo specijalizacija. Klase dobijene specijalizacijom, osim što nasljeđuju sve članove (svojstva i metode) polazne klase, definišu i nove, specifične članove. Polaznu klasu zovemo osnovna klasa (roditeljska klasa, nadklasa), a klasu koja je nasljeđuje zovemo izvedena klasa(podklasa). U izvedenoj klasi definišemo samo atribute i metode specifične za tu klasu (eventualno predefinišemo metode osnovne klase) ali njeni objekti nasljeđuju i sve članove osnovne klase. Izvedena klasa proširuje, a u nekim slučajevima i precizira

osnovnu klasu. U definiciji izvedene klase posle navođenja imena klase navodimo ':' , a zatim slijedi ime osnovne klase iz koje je izvedena.

```
class <imeIzvedeneKlase>:<imeOsnovneKlase>
{
opis / definicija članova klase
}
```

U programskom jeziku C# klasa može naslijediti samo jednu osnovnu klasu, odnosno ne može nastati kao specijalizacija dvije ili više klasa. Takođe, izvedena klasa ne može naslijediti klasu koja sadrži modifikator sealed. Ako želimo da definišemo klasu iz koje se ne mogu izvoditi druge klase modifikator sealed navodimo u zaglavlju klase ispred službene riječi **class**.

Važno je napomenuti da u izvedenoj klasi, bez obzira što naslijeđuje sve članove osnovne klase, ne možemo pristupiti privatnim članovima osnovne klase. Da se ne bi narušila enkapsulacija atributi osnovne klase ne treba da budu javni, a ako su privatni onda im se ne može pristupiti iz izvedene klase. Zato postoji i treći nivo pristupa članovima klase - zaštićeni (engl. protected). Zaštićeni članovi klase dostupni su u klasi u kojoj su definisani i u svim klasama izvedenim iz te klase, a izvan njih nisu. Prema tome, iz izvedene klase može se pristupiti javnim i zaštićenim članovima osnovne klase, ali ne i privatnim. Da bi bio kreiran objekat izvedene klase mora se prvo kreirati njegov osnovni, bazni, dio sa atributima definisanim u osnovnoj klasi. Za kreiranje objekta osnovne klase zadužen je konstruktor osnovne klase. Zato se u konstruktoru izvedene klase implicitno (automatski) poziva konstruktor bez argumenata osnovne klase osim ako programer eksplicitno ne navede koji konstruktor osnovne klase poziva prilikom kreiranja objekta. Poziv konstruktora osnovne klase programer realizuje tako što posle potpisa konstruktora izvedene klase navede dvije tačke (':') a zatim službenu riječ base i u zagradama redom parametre konstruktora osnovne klase kojeg pozivamo.

Sposobnost promenljive da referencira objekte različitih tipova i da automatski poziva odgovarajuću metodu objekta koji se referencira se naziva polimorfizam. Polimorfizam se zasniva na slededem konceptu: metoda koja je deklarisana u baznoj klasi može da se implementira na više različitih načina u različitim izvedenim klasama. Ukoliko se u izvedenoj klasi navede identična metoda (metoda sa istim potpisom) kao i u baznoj klasi, prijaviće se upozorenje o preklapanju u nazivima metoda. Ova situacija se može razriješiti na sledede načine:

- promjena naziva metode u izvedenoj klasi ;
- navođenje ključne riječi new ispred metode u izvedenoj klasi (naslijeđena metoda se sakriva novom metodom u izvedenoj klasi);
- navođenje ključne riječi virtual ispred metode u baznoj i ključne riječi override u izvedenoj klasi (naslijeđena metoda se reimplementira u izvedenoj klasi).

Koja metoda de biti pozvana, metoda bazne ili izvedene klase, određuje se na osnovu tipa instance koju promenljiva referencira, a ne na osnovu tipa same promenljive.

3.1.3 Izuzeci

Izuzeci u programskom jeziku C# predstavljaju mehanizam koji omogućava obradu grešaka do kojih može doći prilikom izvršavanja programa, kako bi se obezbijedilo da se, u slučaju da dođe do takvih grešaka, nastavi sa izvršavanjem programa. Izuzetak (exception) je objekat koji nosi informacije o grešci koja je nastala. Da bi se omogućilo da se različite vrste (tj. tipovi) grešaka prepoznaju i različito obrađuju .NET obezbjeđuje niz klasa izuzetaka.

Tabela 1: Hijerarhija klasa izuzetaka

Exception	
SystemException	
FormatException	int broj = Convert.ToInt32("ABC")
ArithmeticException	
DivideByZeroException	int a = 5, b = 0; Console.WriteLine(a/b);
NullReferenceException	BrojIndeksa[] niz = new BrojIndeksa[2]; niz[0].Broj = 1;
IndexOutOfRangeException	int[] niz = new int[2]; Console.WriteLine(niz[5]);
ArgumentException	
ArgumentOutOfRangeException	List <int> lista = new List <int>(); Console.WriteLine(lista[5]);
ApplicationException (klasa iz koje se izvode korisnički-definisani izuzeci)	

Kod se može podijeliti u tri bloka:

```
try
{
// kod koji se odnosi na logiku programa, a čije izvršavanje može
izazvati grešku
}
catch(Exception)
{
// kod kojim se obrađuje određena vrsta greške
// izvršva se samo u slučaju da je došlo do određene vrste greške
}
finally
{
// kod tj. akcija koju uvijek na kraju treba izvršiti (npr. oslobađanje
resursa)
// izvršava se u svakom slučaju bez obzira na to da li je prethodno
došlo do greške ili ne
}
```

Catch ili **finally blok** se mogu izostaviti. DivideByZeroException dobijamo kada dijelimo cijeli broj nulom, a ako dijelimo decimalne brojeve nulom nećemo dobiti izuzetak. Kada su u pitanju decimalni brojevi oni imaju specijalne konstante, tj. Double.PositiveInfinity, Double.NegativeInfinity i Double.NaN (ako dijelimo nulu sa nulom).

Tok izvršavanja počinje sa **try** blokom. Ukoliko ne dođe ni do jedne greške unutar **try** bloka, tok izvršavanja se na kraju ovog bloka automatski prenosi na **finally** blok ukoliko postoji (ukoliko ne postoji tok izvršavanja se prenosi na prvu sledeću naredbu). Ukoliko dođe do greške unutar **try** bloka njegovo izvršavanje se momentalno prekida. Tok izvršavanja se prnosi na **catch** blok kako bi se obradila greška. Preostali dio koda u **try** bloku nikada neće biti izvršen. Po završetku obrade, tok se prenosi na **finally** blok ili prvu sledeću naredbu.

S obzirom na to da postoji više različitih tipova izuzetaka, moguće je navesti više **catch** blokova. Ako u toku izvršavanja programa dođe do greške, izvršno okruženje CLR:

1. Privremeno prekida normalno izvršavanje programa
2. „Podiže izuzetak“ - odnosno instancira objekat odgovarajuće klase izuzetka i podiže taj objekat (**throw**)
3. Traži najbliži catch blok koji može da obradi dati tip izuzetka
   - Najprije pretražuje catch blokove tekućeg **try** bloka (tj. bloka unutar koga je došlo do greške), po redosledu navođenja
   - Ako pretraga nije uspjela, pretražuje **catch** blokove **try** bloka unutar koga je tekući **try** blok ugnježden ili iz koga je metoda (u kojoj se nalazi tekući **try** blok) pozvana
   - Pretraživanje se nastavlja na ovaj način vraćanjem unazad kroz pozvane metode (call stack) sve dok se ne pronađe **catch** blok koji može da obradi nastali izuzetak. Ukolko se ne pronađe nijedan **catch** blok koji može da obradi dati izuzetak CLR će prekinuti izvršavanje cijelog programa.

Kada CLR pronađe odgovarajući **catch** blok:

- priprema prenos toka izvršavanja na njegovu prvu naredbu;
- prije samog prenosa izvršava sve **finally** blokove pridružene prethodno ispitivanim **catch** blokovima (po redosledu u kojem je vršeno pretraživanje).

3.1.4 Apstrakcija podataka

Apstrakcija, kao još jedan važan koncept u C# programskom jeziku, je proces skrivanja detalja implementacije, čime se korisniku pokazuju samo funkcionalnosti. Time se fokus skreće na to šta objekat radi, a ne kako to radi. Postoje dva načina za postizanje apstrakcije u C#-u, a to su apstraktne klase i interfejsi.

Glavna razlika između ova dva koncepta jeste da se apstraktnom klasom postiže djelimična apstrakcija, dok se interfejsom postiže potpuna apstrakcija. Apstraktna klasa može prosleđivati svoje metode i polja, ali sama ne može biti instancirana (ne može se kreirati njen objekat). Svaka klasa koja sadrži barem jednu apstraktnu metodu mora biti deklarisana kao apstraktna. Ključna reč za apstraktnu klasu jeste abstract. Za razliku od apstraktne klase, gdje se mogu pojaviti i metode koje nisu apstraktne, kod interfejsa sve metode moraju biti apstraktne. Iako je ova mogućnost dostupna i kod apstraktnih klasa, upotrebnost interfejsa je široka. Pored pune apstrakcije, interfejs podržava mogućnost višestrukog nasleđivanja (za razliku od apstraktnih klasa). U C#-u interfejs se implementira tako što se iza naziva klase koja implementira interfejs navedu „:“ i naziv interfejsa.

3.2 GRAFIČKI INTERFEJS U C#

Grafički interfejs(GUI) u C#-u se kreira korišćenjem Windows Forms-a. Potrebno je koristiti .NET Framework. Windows Forms u C# je stariji tehnološki stack za kreiranje grafičkog korisničkog interfejsa aplikacija. On pruža različite kontrole i funkcionalnosti koje omogućavaju dizajniranje i razvijanje interaktivne desktop aplikacije. Neke od ključnih karakteristika i kontrola koje se mogu koristiti u Windows Forms su:

1. Kontrole:
    - Button - dugme koje korisnik može kliknuti
    - Label - tekstualna oznaka ili natpis
    - TextBox - polje za unos teksta od strane korisnika
    - ComboBox - padajuća lista za izbor jedne stavke iz više opcija
    - ListBox - lista za prikazivanje više stavki između kojih korisnik može birati
    - CheckBox - polje za potvrdu koje korisnik može označiti ili poništiti
    - RadioButton - jedna od više opcija koje korisnik može izabrati
    - PictureBox - kontrola za prikazivanje slika
    - DataGridView - tabela za prikazivanje i uređivanje tabelarnih podataka
    - TreeView - kontrola za prikaz hijerarhijskih podataka u obliku stabla
2. Događaji:
    - Click - događaj koji se pokreće kada korisnik klikne na dugme
    - TextChanged - događaj koji se pokreće kada se promeni tekst u TextBox-u
    - SelectedIndexChanged - događaj koji se pokreće kada korisnik izabere drugu stavku u ComboBox-u
    - MouseEnter, MouseLeave - događaji koji se pokreću kada miš uđe ili izađe iznad kontrole
3. Layout upravljanje:
    - Windows Forms pruža različite kontejnere za organizaciju kontrola na formi, kao što su Panel, GroupBox, TableLayoutPanel i FlowLayoutPanel. Ovi kontejneri omogućavaju postaviljanje kontrole u unaprijed definisani raspored.
4. Pomoćne klase i funkcionalnosti:
    - MessageBox - prikazivanje dijaloga sa porukom korisniku.
    - OpenFileDialog, SaveFileDialog - dijalozi za odabir i čuvanje datoteka.
    - ToolTip - prikazivanje dodatnih informacija kada korisnik pređe mišem preko kontrola.
    - Printing - mogućnost štampanja sadržaja iz aplikacije.

3.2.1 Izvještaji u C#

U C#-u, kreiranje izvještaja može se postići korištenjem različitih biblioteka i tehnika za generiranje i manipuliranje dokumentima. Neki načina za stvaranje dokumenata u C#-u su:

1. Microsoft Office interop - Microsoft Office interop omogućuje rad s dokumentima u programima kao što su Word, Excel i PowerPoint.
2. Open XML SDK - Open XML SDK je biblioteka koja omogućuje generiranje, uređivanje i čuvanje dokumenata u formatu Office Open XML (docx, xlsx, pptx).
3. PDF biblioteke - Postoji nekoliko popularnih biblioteka koje vam omogućuju stvaranje PDF dokumenata u C#-u. Neke od tih biblioteka su iTextSharp, PdfSharp i Syncfusion Essential PDF. Ove biblioteke omogućuju generiranje PDF dokumenata, dodavanje teksta, slika, tablica i drugih elemenata.
4. HTML to PDF konverzija – Moguće je koristiti biblioteke kao što su iTextSharp ili SelectPdf za pretvaranje HTML-a u PDF dokumente.
5. Reporting Services - Microsoft SQL Server Reporting Services (SSRS) omogućuje generiranje raznih izvještaja u različitim formatima, uključujući Word, Excel, PDF i druge.

Za kreiranje izvještaja u radu koristićemo Crystal Reports. Crystal Reports spada u peti način za stvaranje dokumenata u C#-u. Crystal Reports je alat koji omogućava kreiranje kompleksnih i bogatih izveštaja iz različitih izvora podataka. Ovaj alat se često koristi za pravljenje poslovnih izvještaja i analiza, i omogućava integraciju sa Microsoft SQL Server Reporting Services (SSRS) kako bi se omogućio pristup raznim formatima izveštaja. Takođe, Crystal Reports može biti integrisan direktno u C# aplikacije kako bi programeri mogli koristiti njegove mogućnosti za generisanje i prikazivanje izvještaja unutar svojih aplikacija.

3.3 NITI

Dijakstra je rekao da se konkurentnost dešava kada postoje dva ili više izvršnih tokova (procesa) koji su sposobni da se izvršavaju simultano (istovremeno). Procesi mogu istovremeno da koriste dijeljene resurse što može da rezultuje nepredviđenim ponašanjem sistema. Uvođenje međusobnog isključenja može da spriječi nepredviđeno ponašanje kod korišćenja dijeljenih resursa ali može da dovede do pojava **mrtvog zaključavanje** i **gladovanja**.

Mrtvo zaključavanje (deadlock) je multitasking problem koji se dešava kada dva procesa zauzmu resurse i međusobno se čekaju da oslobode te resurse.

Gladovanje (starvation) je multitasking problem koji se dešava kada neki proces zauzme neki resurs i ne dozvoljava drugim procesima da ga koriste. To dovodi do toga da drugi procesi ne mogu da se do kraja izvrše.

Ukoliko više procesa međusobno sarađuju u izvršenju nekog zadatka javlja se problem njihove međusobne komunikacije i razmjene podataka jer svaki proces zauzima poseban memorijski prostor. Taj problem je riješen pojavom niti koje dijele isti memorijski prostor.

Nit je osnovna jedinica izvršavanja u programskom jeziku C#. Ona predstavlja putanju izvršavanja koja može istovremeno izvršavati kod sa drugim nitima, čime se postiže višenitno izvršavanje. Niti, kao što je rečeno, dijele isti adresni prostor u okviru jednog procesa i komunikacija između njih je dosta jednostavnija u odnosu na komunikaciju između procesa. Radom sa više procesa upravlja operativni sistem, dok radom sa više niti upravlja C# okruženje.

U C#-u, nitima se upravlja pomoću klase **Thread** iz **System.Threading** namespace-a. Da biste koristili niti, potrebno je kreirati novi objekat klase **Thread** i proslijediti mu metodu koju želitmo da se izvršava asinhrono. U C# programskom jeziku, stanje niti može se provjeriti korištenjem svojstva **Thread.ThreadState**. Neka od stanja niti su: Moguća stanja niti su:

- Unstarted - Nit je stvorena, ali još nije pokrenuta
- Running - Nit se trenutno izvršava
- WaitSleepJoin - Nit je u stanju mirovanja, spavanja ili čekanja na neki događaj
- Stopped - Nit je završila s izvršavanjem
- Aborted - Nit je prekinuta (abortirana)
- Stopped - Nit je zaustavljena
- Background - Nit je označena kao pozadinska nit
- Suspended - Nit je privremeno zaustavljena

3.3.1 Zaključavanje podataka

U C# programiranju, zaključavanje podataka se koristi kako bi se obezbijedilo ispravno i sigurno dijeljenje podataka između više niti u višenitnom okruženju. Kada više niti

istovremeno pristupa i mijenja zajedničke podatke, postoji mogućnost da se podaci oštete ili dođe do neispravnog stanja. Zaključavanje podataka omogućava sinhronizaciju pristupa podacima tako da samo jedna nit može pristupiti podacima u određenom trenutku.

U C#-u, zaključavanje podataka se obično postiže pomoću ključne reči lock i objekta koji se koristi kao zaključavanje. Objekat koji se koristi kao zaključavanje trebao bi biti zajednički za sve niti koje pristupaju podacima koje želimo zaključati. Ovaj objekat se naziva i monitor objekat. Sintaksa za zaključavanje podataka u C# izgleda ovako:

```
lock (lockObject)
{
    // Blok koda u kojem se pristupa i menja zajednički podaci
}
```

Prilikom izvršavanja ovog koda, nit koja prva stigne do bloka lock zaključava lockObject i izvršava kod unutar bloka. Ostale niti koje dođu do istog bloka moraju čekati dok se lockObject ne otključa. Kada prva nit završi s izvršavanjem bloka, ona otključava lockObject i omogućava drugim nitima pristup. Na taj način se osigurava da samo jedna nit može izvršavati kod unutar zaključanog bloka u isto vreme.

Takođe pored lock mehanizma može se koristiti i mutex mehanizam. **Mutex** je mehanizam za zaključavanje podataka koji omogućava sinhronizaciju pristupa dijeljenim resursima između više niti u višenitnom okruženju. Mutex je skraćenica za „mutual exclusion“, što znači međusobno isključivanje.

Mutex obezbjeđuje ekskluzivan pristup resursima tako da samo jedna nit može posjedovati mutex u određenom trenutku. Ovaj mehanizam je koristan u situacijama gde je potrebno zaštiti dijeljene resurse.

```
Mutex mutex = new Mutex();
mutex.WaitOne();   // Blokiranje mutexa
try
{
    // Blok koda u kojem se pristupa i menja deljeni resurs
}
finally
{
    mutex.ReleaseMutex();   // Otključavanje mutexa
}
```

Prilikom izvršavanja ovog koda, nit koja prva stigne do poziva **mutex.WaitOne()** zaključava mutex i nastavlja sa izvršavanjem bloka koda unutar try bloka. Ostale niti koje dođu do **mutex.WaitOne()** moraju čekati dok se mutex ne otključa pozivom **mutex.ReleaseMutex().** Kada nit završi izvršavanje bloka koda, ona otključava mutex, omogućavajući drugim nitima pristup deljenom resursu.

Važno je napomenuti da se mutex može koristiti za sinhronizaciju pristupa podacima unutar jedne aplikacije, ali nije pogodan za sinhronizaciju pristupa između više aplikacija. Za takve scenarije mogu se koristiti drugi mehanizmi, poput semafora (semaphore) ili interprocesnih zaključavanja (interprocess locking).

3.4 RAD U MREŽI

3.4.1 IP adrese

IP adrese (Internet Protocol adrese) su numeričke adrese koje se koriste za identifikaciju uređaja na mreži i omogućavajući međusobnu komunikaciju putem internet protokola. Struktura IP adrese se sadrži od 4 osmobitna bloka, npr. 87.116.181.49. Vrijednosti svakog bloka se kreću od 0 do 255. Pored numeričke vrijednosti adrese, postoji i simbolička adresa, kao što je fon.bg.ac.rs. Jedna simbolička adresa može biti vezana za više numeričkih adresa. Servis za povezivanje simboličkih i numeričkih adresa jeste DNS (Domain Naming Service). Postoje dvije vrste IP adresa: privatne (lokalne) IP adrese i javne IP adrese.

Privatne IP adrese su adrese koje se koriste unutar lokalnih mreža (LAN - Local Area Network) kako bi se identifikovali uređaji unutar iste mreže. Ove adrese nisu vidljive izvan lokalne mreže i ne mogu se koristiti za komunikaciju preko interneta. Dizajnirane su da omoguće internu komunikaciju između uređaja u okviru kućnih mreža, poslovnih mreža, ili drugih organizacijskih mreža. Privatne IP adrese automatski dodjeljuje DHCP(Dynamic Host Configuration Protocol).

Najčešće korištene lokalne IP adrese pripadaju IP adresnom rasponu:

- 10.0.0.0 - 10.255.255.255 (10.0.0.0/8)
- 172.16.0.0 - 172.31.255.255 (172.16.0.0/12)
- 192.168.0.0 - 192.168.255.255 (192.168.0.0/16)

Javne IP adrese su adrese koje su vidljive na internetu i koriste se za identifikaciju uređaja na globalnom nivou. Svaki uređaj koji ima pristup internetu ima jedinstvenu javnu IP adresu koja mu omogućuje komunikaciju s drugim uređajima na internetu. Te adrese se dodjeljuju od strane internet provajdera i dodijeljene su globalnoj mrežnoj infrastrukturi.

Kada uređaji u lokalnoj mreži žele komunicirati s uređajima na internetu, koriste NAT (Network Address Translation) kako bi se lokalne IP adrese pretvorile u javne IP adrese kako bi mogli pristupiti globalnoj mreži.

3.4.2 Soketi

Da bi se omogućilo povezivanje između klijenta i servera na nekoj mrežnoj aplikaciji, svaki od ova dva dijela programa otvara soket na kraju svoje konekcije preko koje se čitaju i upisuju podaci. Adresa soketa se sastoji iz dva dijela: adrese računara na kome se nalazi program koji je generisao soket i broj porta koji je generisan pomoću soketa(npr. 87.116.181.49:5432). C# pruža biblioteku klasa za upravljanje soketima kroz namespace System.Net.Sockets. Zadatak serverskog soketa jeste da osluškuje mrežu, držeći otvoren port za konekciju. Kada se određeni klijent preko svoje Socket klase konektuje na serverski soket gađajući adresu i port servera, konekcija je uspostavljena.

Postoje dva osnovna tipa soketa u C#:

- TCP soketi (Transmission Control Protocol) - Koriste se za pouzdanu, redoslijednu i dvosmjernu komunikaciju.
- UDP soketi (User Datagram Protocol) - Koriste se za brzu, nesigurnu i jednosmjernu komunikaciju.

3.5 RAD SA BAZOM PODATAKA

Pri kreiranju aplikacije koristićemo bazu podataka koju kreiramo korišćenjem Majkrosoftovoj alata SQL Server Management Studio (SSMS). SSMS je integrisano razvojno okruženje koje se koristi za upravljanje i održavanje Microsoft SQL baza podataka. To je alat koji pruža širok spektar funkcionalnosti za administraciju, razvoj, optimizaciju i dijagnostiku SQL Server sistema. Napravili smo instancu lokalnog servera i na njemu napravili bazu podataka koja se zove AIRMAC.

3.5.1 Funkcije

SQL (Structured Query Language) je jezik koji se koristi za upravljanje i manipulaciju podacima u relacijskim bazama podataka. Funkcije u SQL-u su posebni elementi koji omogućavaju izvršavanje određenih operacija nad podacima. Ove funkcije mogu biti ugrađene (integrisane u sam SQL sistem) ili korisnički definisane (napisane od strane programera).

Ugrađene funkcije u SQL-u pružaju različite mogućnosti za obradu podataka. Neke od najčešće korišćenih ugrađenih funkcija uključuju:

1. Agregatne funkcije - Ove funkcije izračunavaju agregirane vrijednosti na osnovu skupa podataka. Na primjer, funkcija SUM() se koristi za izračunavanje zbirnih vrednosti, funkcija AVG() za izračunavanje prosjeka, a funkcija COUNT() za brojanje redova ili vrijednosti u koloni.
2. Matematičke funkcije - SQL pruža različite matematičke funkcije kao što su ROUND() za zaokrugljivanje brojeva, ABS() za apsolutnu vrednost, SQRT() za izračunavanje kvadratnog korijena, itd.
3. String funkcije - Ove funkcije se koriste za manipulaciju sa stringovima. Na primer, funkcija CONCAT() se koristi za spajanje stringova, funkcija UPPER() za pretvaranje stringa u velika slova, a funkcija SUBSTRING() za izdvajanje dijela stringa.
4. Datum i vrijeme funkcije - SQL pruža funkcije za rad sa datumima i vremenom. Na primer, funkcija NOW() vraća trenutni datum i vreme, funkcija DATE() izdvaja samo datum iz vremenske vrijednosti, a funkcija DATEADD() dodaje određeni broj vremenskih intervala (dani, mjeseci, godine) datom datumu.
5. Konverzije podataka - SQL ima funkcije za konverziju podataka iz jednog tipa u drugi. Na primer, funkcija CAST() se koristi za eksplicitno konvertovanje jednog tipa podataka u drugi, a funkcija CONVERT() pruža slične mogućnosti konverzije podataka.

Korisnički definisane funkcije su funkcije koje programeri mogu napisati kako bi prilagodili funkcionalnost SQL sistema svojim specifičnim potrebama. Ove funkcije se definišu korišćenjem ključne reči CREATE FUNCTION i mogu izvršavati kompleksne operacije nad podacima.

Funkcije u SQL-u su veoma korisne za manipulaciju podacima, izračunavanje vrednosti i izvršavanje različitih operacija. Kombinovanjem ugrađenih i korisnički definisanih

funkcija, moguće je prilagoditi SQL bazu podataka i izvršiti širok spektar operacija nad podacima.

Primjer korisnički definisane funkcije u bazi AIRMAC:

```
CREATE FUNCTION [dbo].[hours_add](@h1 decimal(18,2),@h2 decimal(18,2))
RETURNS decimal(18,2) AS
BEGIN
DECLARE
@z1 AS int,
@z2 AS int

SELECT @z1 = cast(@h1 * 100 / 100 as int) * 60 + cast(@h1 * 100 as int) % 100
SELECT @z2 = cast(@h2 * 100 / 100 as int) * 60 + cast(@h2 * 100 as int) % 100
return cast((@z1 + @z2) / 60 as int) + cast((@z1 + @z2) % 60 as
decimal(18,2)) /100
END
```

3.5.2 Trigeri

Triggeri u SQL bazi su objekti koji se koriste za automatsko izvršavanje određenih radnji ili reakcija kada se određeni događaji dešavaju u bazi podataka. Ovi događaji mogu biti promene podataka (npr. unos, izmena ili brisanje redova) ili određeni sistemski događaji (npr. pokretanje baze podataka, promena korisnika itd.).

Trigeri se definišu u okviru baze podataka i vezani su za određene tabele ili događaje. Kada se aktivira odgovarajući događaj, triger se pokreće i izvršava zadate akcije ili logiku. Glavne vrste triger-a u SQL-u su:

1. Before trigeri (pre trigeri): Ovi trigeri se pokreću pre izvršenja određene operacije nad podacima. Na primer, BEFORE INSERT triger će se aktivirati pre nego što se novi red unese u tabelu. Ovi trigeri se često koriste za validaciju podataka, postavljanje podrazumevanih vrednosti ili modifikaciju podataka pre nego što budu upisani.
2. After trigeri (posle trigeri): Ovi trigeri se pokreću nakon izvršenja određene operacije nad podacima. Na primer, AFTER UPDATE triger će se aktivirati nakon što se izmene podaci u tabeli. Ovi trigeri se često koriste za ažuriranje drugih tabela, vođenje evidencije promena ili slanje obaveštenja.
3. Instead of trigeri (umesto trigeri): Ovi trigeri se koriste za izvršavanje posebnih akcija umesto standardnih operacija nad podacima. Na primer, INSTEAD OF DELETE triger može biti definisan kako bi izvršio posebne akcije umesto brisanja reda iz tabele. Ovi trigeri se često koriste za implementaciju složenih poslovnih pravila ili za kontrolisano izvršavanje radnji.

Trigeri pružaju fleksibilnost i mogućnost automatizacije određenih operacija u bazi podataka. Oni omogućavaju programerima da definišu složene logike ili radnje koje se automatski izvršavaju pri određenim događajima. Međutim, važno je pažljivo upravljati trigerima, jer nepravilno definisani ili prekomerno korišćeni trigeri mogu uticati na performanse baze podataka.

Primjer instead of trigera za tabelu LogBook u bazi AIRMAC:

```
CREATE TRIGGER [dbo].[Insert_LOGBOOK]
    ON [dbo].[LogBook]
    INSTEAD OF INSERT
AS

DECLARE @ID_LogBook decimal(18,0)
DECLARE @RegistrationNumber varchar(10)
DECLARE @LastACHours decimal(18,2)
DECLARE @LastACCycles decimal(18,0)
DECLARE @ID_Airport decimal(18,0)
DECLARE @ID_Airport_FROM decimal(18,0)
DECLARE @PreviousACHours decimal(18,2)
DECLARE @PreviousACCycles decimal(18,0)
DECLARE @df decimal(18,2)


BEGIN

  BEGIN TRANSACTION

  BEGIN TRY
  SELECT @RegistrationNumber = RegistrationNumber, @ID_Airport_FROM = ID_Airport_FROM,
@PreviousACHours = PreviousACHours,  @PreviousACCycles = PreviousACCycles FROM inserted
  SELECT @LastACHours = LastACHours, @LastACCycles = LastACCycles, @ID_Airport = ID_Airport
FROM
Aircraft WHERE RegistrationNumber =  @RegistrationNumber
  IF not (@ID_Airport_FROM = @ID_Airport AND @PreviousACHours = @LastACHours AND
@PreviousACCycles = @LastACCycles)  RAISERROR ('Origin record is changed!.', 16, 1)
  SELECT * INTO #inserted FROM inserted
  INSERT INTO LogBook
(FlightDate,FlightNumber,ID_Airport_FROM,ID_Airport_TO,RegistrationNumber,FlightTimeStart,
FlightTimeStop,PreviousACHours,PreviousACCycles,NextACHours,NextACCycles) output
inserted.ID_LogBook SELECT
FlightDate,FlightNumber,ID_Airport_FROM,ID_Airport_TO,RegistrationNumber,FlightTimeStart,
FlightTimeStop,PreviousACHours,PreviousACCycles,NextACHours,NextACCycles FROM #inserted

  UPDATE Aircraft SET
           LastACHours = source.NextACHours,
           LastACCycles = source.NextACCycles,
           LastUpdate = source.FlightTimeStop,
           ID_Airport = source.ID_Airport_TO
           FROM
          (select NextACHours , NextACCycles , FlightTimeStop, ID_Airport_TO,
RegistrationNumber
           FROM LogBook t1
           Where
          (SELECT COUNT(*)
           FROM LogBook AS t2
           Where t2.RegistrationNumber = t1.RegistrationNumber And t2.ID_LogBook >
t1.ID_LogBook) = 0
          ) AS source
           WHERE Aircraft.RegistrationNumber = source.RegistrationNumber
           AND source.RegistrationNumber = @RegistrationNumber

   END TRY
   BEGIN CATCH
   IF @@TRANCOUNT > 0
   PRINT ERROR_MESSAGE()
        ROLLBACK TRANSACTION
   END CATCH;

   IF @@TRANCOUNT > 0
        COMMIT TRANSACTION

END
```

3.6 SOFTVERSKI PATERNI

Softverski obrasci (paterni) su generički modeli koji se koriste za riješavanje uobičajenih problema u softverskom dizajnu. Oni predstavljaju provjerene i dobro strukturirane načine za riješavanje određenih problema u razvoju softvera.

Postoji mnogo različitih softverskih obrazaca, ali najpoznatiji su:

- Singleton Pattern - Ovaj obrazac obezbjeđuje da postoji samo jedna instanca klase u cijeloj aplikaciji. Koristi se kada je potrebno da postoji samo jedna globalna tačka pristupa određenim resursima.
- Factory Pattern - Fabrički obrazac koristi posebnu klasu (fabriku) koja proizvodi objekte umesto da se kreira novi objekat direktno. To omogućava da se izbjegne zavisnost od konkretnih klasa i pojednostavi proces kreiranja objekata.
- Strategy Pattern - Ovaj obrazac omogućava definisanje različitih strategija (algoritama) i njihovu zamjenu bez izmena u kodu klijenta koji koristi te strategije. Koristi se kada postoji potreba za dinamičkim izborom između različitih algoritama.
- Observer Pattern - Posmatrač omogućava da objekti (posmatrači) automatski reaguju na promene koje se dešavaju u drugim objektima (subjektima). Koristi se u situacijama gdje postoji potreba za obavještavanjem više objekata o promenama u drugim objektima.
- Decorator Pattern - Dekorater omogućava dodavanje novih funkcionalnosti objektima dinamički, bez potrebe za izmjenom originalnog koda. To omogućava fleksibilno proširivanje funkcionalnosti objekata.
- Builder Pattern - Graditelj se koristi za kreiranje složenih objekata korak po korak. Omogućava da se konstrukcija objekta odvaja od njegove reprezentacije, što olakšava različite načine kreiranja istog objekta.
- Template Method Pattern - Ovaj obrazac se koristi za definisanje osnovne strukture algoritma, ostavljajući specifične korake implementacije podklasama.

Ovo su samo neki od mnogih softverskih obrazaca koji se koriste u programiranju. Svaki obrazac ima svoje prednosti i mane i odabir pravog zavisi od specifičnih zahtjeva problema koji treba riješiti.

3.6.1 Singleton pattern

Singleton je dizajnerski obrazac koji se koristi u razvoju softvera kako bi se osiguralo da postoji samo jedna instanca određene klase, te pruža globalnu tačku pristupa toj instanci. Ovaj obrazac često se primjenjuje kada je potrebno da se jedan objekat koristi u celom sistemskom kontekstu ili kada je neophodno deljenje resursa između različitih delova aplikacije.

Ključna karakteristika singletona je da ima privatni konstruktor koji sprečava direktno instanciranje klase izvan same klase. Umesto toga, singleton klasa pruža statičku metodu koja omogućava pristup jedinstvenoj instanci klase. U pozadini, singleton klasa održava referencu na jednu jedinstvenu instancu, koja se kreira prilikom prvog poziva metode za pristup.

Prednosti upotrebe singletona uključuju:

- Jedinstvena instanca: Singleton garantuje da postoji samo jedna instanca klase u celom sistemu. To je korisno kada je neophodno deljenje resursa i podataka između različitih delova aplikacije.
- Globalni pristup: Singleton obezbeđuje globalnu tačku pristupa instanci, omogućavajući lako korišćenje objekta iz bilo kojeg dela aplikacije.
- Efikasnost: Pošto se instanca kreira samo jednom, singleton može biti efikasan način upravljanja resursima, poput baze podataka ili mrežnih veza, umesto da se nepotrebno ponovo kreira prilikom svakog zahteva.
- Lakše testiranje: Singleton može biti lakše testiran jer omogućava kontrolu nad jedinstvenom instancom, olakšavajući izolaciju i ponovljive testove.

Singleton je dizajnerski obrazac koji omogućava kreiranje samo jedne instance klase i pruža globalnu tačku pristupa toj instanci. Ovaj obrazac je koristan kada je potrebno deljenje resursa između različitih delova aplikacije i omogućava efikasno upravljanje resursima. Međutim, treba pažljivo razmotriti njegovu upotrebu i uzeti u obzir specifične zahteve i kontekst aplikacije.

3.6.2 Template method pattern

Template Method pattern je obrazac ponašanja (behavioral design pattern) koji definiše kostur algoritma u baznoj klasi, ali omogućava podklasama da izmene određene korake algoritma bez promene njegove strukture. Ovaj obrazac promoviše ponovno korišćenje koda i pruža način da se definiše šablon za algoritam, dok istovremeno omogućava podklasama da prilagode određene delove. Template Method obrazac vam omogućava da definišete zajedničku strukturu algoritma u baznoj klasi, dok omogućava podklasama da prilagode određene korake prema svojim potrebama.

3.6.3 Observer pattern

Observer pattern je obrazac ponašanja (behavioral design pattern) koji omogućava objektima da se automatski obaveštavaju o promenama stanja u drugim objektima. Ovaj obrazac uspostavlja jedan-na-više odnos između subjekta (objekta koji obaveštava o promenama) i više posmatrača (objekata koji primaju obaveštenja).

Glavna ideja Observer obrasca je da subjekat održava listu posmatrača i obaveštava ih o bilo kakvim promenama u svom stanju. Kada se stanje subjekta promeni, svi registrovani posmatrači dobijaju obaveštenje i mogu preuzeti potrebne akcije.

Observer obrazac omogućava slabu povezanost između subjekta i posmatrača, što omogućava fleksibilnost i olakšava dodavanje novih posmatrača ili izmene postojećih posmatrača bez uticaja na subjekt.

4 STUDIJSKI PRIMJER

Studijski primjer napravljen u C# programskom jeziku, služeći se opisanim .NET tehnologijama i poštujući pet faza uprošćene Larmanove metode.

4.1 FAZA PRIKUPLJANJA KORISNIČKIH ZAHTJEVA

4.1.1 Verbalni opis

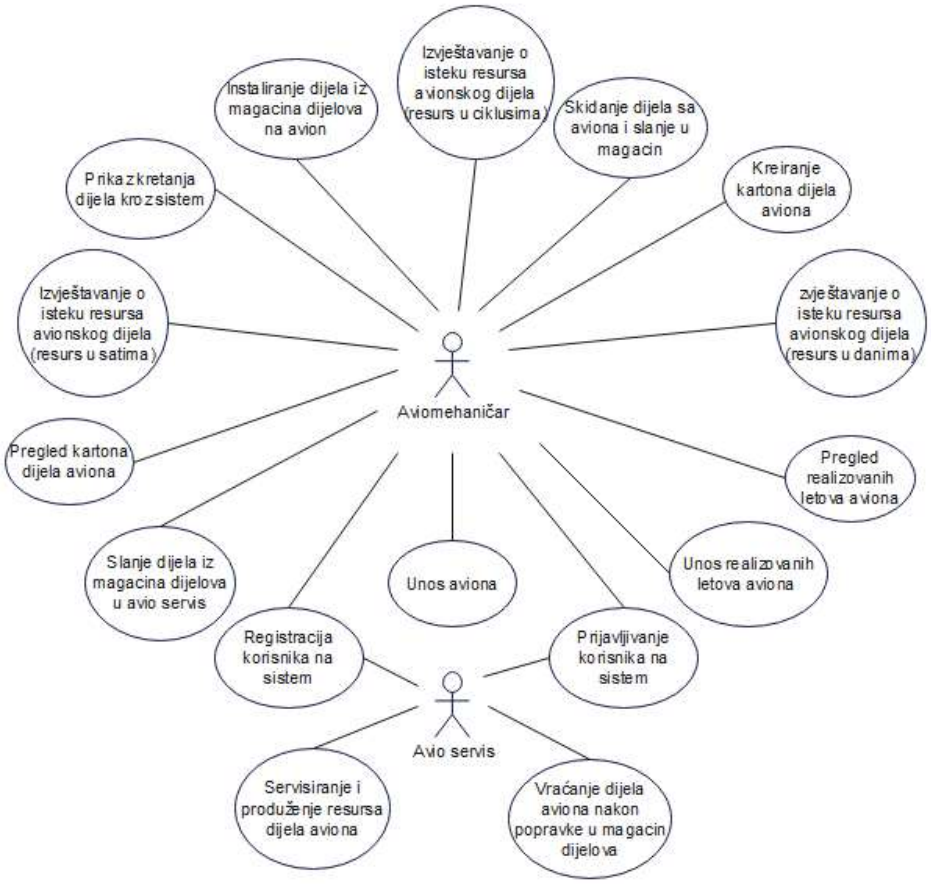
Aplikacija omogućava vođenje evidencije avionskih dijelova u jednom sistemu održavanja vazduhoplova. Sistem prati i kontroliše trošenje resursa avionskog dijela u cilju preventivnog izvještavanja o preostalim resursima avionskog dijela, a u cilju planiranja njegovog servisiranja radi zanavljanja trajanja resursa. Resursi avionskog dijela se vode po više kriterijuma zavisno da li je avionski dio lociran na avionu (maksimalan broj sati i/ili maksimalan broj ciklusa naleta aviona, kao i maksimalan broj dana provedenih na avionu) ili u magacinu (maksimalan broj dana provedenih u magacinu). U slučaju dostizanja limita bilo kog od navedenih kriterijuma sistem mora alarmirati aviomehaničara radi slanja avionskog dijela u avio servis. U avio servisu se avionski dio testira, servisira i izdaje se sertifikat o produžetku limita po svim kriterijumima.

Aviomehaničar ima mogućnost unosa, pregledanja i ažuriranja realizovanih letova aviona. Takođe aviomehaničar ima mogućnost kreiranja kartona dijela aviona o vođenju evidencije o njegovoj lokaciji (avion, magacin, servis), kao i trošenju njegovih resursa. Dio iz magacina može biti instaliran na avion, skinut sa njega i poslat u magacin ili servis. Servis preuzima avionski dio i nakon predviđenih procedura izdaje sertifikat o zanavljanju resursa i dio vraće u magacin pošiljaoca.

4.1.2 Slučajevi korišćenja

U ovom softverskom sistemu, identifikovano je 17 slučajeva korišćenja:

1. Registracija korisnika na sistem
2. Prijavljivanje korisnika na sistem
3. Unos aviona
4. Unos realizovanih letova aviona
5. Pregled realizovanih letova aviona
6. Kreiranje kartona dijela aviona
7. Pregled kartona dijela aviona
8. Instaliranje dijela iz magacina dijelova na avion
9. Skidanje dijela sa aviona i slanje u magacin
10. Slanje dijela iz magacina dijelova u avio servis
11. Izvještavanje o preostalim resursima dijela aviona
12. Izvještavanje o isteku resursa avionskog dijela (resurs u satima)
13. Izvještavanje o isteku resursa avionskog dijela (resurs u danima)
14. Izvještavanje o isteku resursa avionskog dijela (resurs u ciklusima)
15. Servisiranje i produženje resursa dijela aviona
16. Vraćanje dijela aviona nakon popravke u magacin dijelova
17. Prikaz kretanja dijela kroz sistem



Slika 6: Slučajevi korišćenja

4.1.2.1 SK1: Slučaj korišćenja – Registracija korisnika na sistem

Naziv SK
Registracija korisnika na sistem

Aktori SK
Aviomehaničar/avio servis

Učesnici SK
Aviomehaničar/avio servis i sistem

Preduslov
Sistem je funkcionalan, sistem prikazuje formu za unos podataka.

Osnovni scenario SK

1. Aviomehaničar/avio servis unosi informacije o korisniku. (APUSO)
2. Aviomehaničar/avio servis poziva sistem da na osnovu unijetih podataka kreira korisnički nalog. (APSO)
3. Sistem kreira korisnički nalog na osnovu unijetih podataka. (SO)
4. Sistem prikazuje poruku „Sistem je dodao korisnika u bazu korisnika! “ (IA)

Alternativna scenarija

4.1 Ukoliko sistem ne može da kreira korisnički nalog on aviomehaničaru/avio servisu prikazuje poruku „Sistem ne može da sačuva podatke o novoj registraciji! “. (IA)

4.1.2.2 SK2: Slučaj korišćenja – Prijavljivanje korisnika na sistem

Naziv SK
Prijavljivanje korisnika na sistem

Aktori SK
Aviomehaničar/avio servis

Učesnici SK
Aviomehaničar/avio servis i sistem

Preduslov
Sistem je funkcionalan, korisnički nalog postoji.

Osnovni scenario SK

1. Aviomehaničar/avio servis unosi korisničko ime i lozinku. (APUSO)
2. Aviomehaničar/avio servis poziva sistem da na osnovu unijetih podataka pronađe korisnički nalog. (APSO)
3. Sistem pronalazi korisnički nalog i prijavljuje korisnika na sistem. (SO)
4. Sistem prikazuje poruku „Korisnik je prijavljen. “. (IA)

Alternativna scenarija

4.1 Ukoliko sistem ne može da pronađe korisnički nalog on aviomehaničaru/avio servisu prikazuje poruku „Sistem ne može da pronađe podatke o korisniku. “. (IA)

4.1.2.3 SK3: Slučaj korišćenja – Unos aviona

Naziv SK
Unos aviona

Aktori SK
Aviomehaničar

Učesnici SK
Aviomehaničar i sistem

Preduslov
Sistem je funkcionalan, aviomehaničar je prijavljen na sistem pod svojim nalogom, sistem prikazuje formu za unos podataka. Šifarnici aerodroma su inicijalizovani. Učitana je lista aerodroma.

Osnovni scenario SK

1. Aviomehaničar unosi podatke koji identifikuju avion. (APUSO)
2. Aviomehaničar poziva sistem da na osnovu unijetih podataka sačuva avion. (APSO)
3. Sistem kreira avion na osnovu unijetih podataka. (SO)
4. Sistem prikazuje poruku „Sistem je dodao avion u bazu aviona! “. (IA)

Alternativna scenarija

4.1 Ukoliko sistem ne može da kreira avion on aviomehaničaru prikazuje poruku „Sistem ne može da sačuva podatke o avionu! “. (IA)

4.1.2.4 SK4: Slučaj korišćenja – Unos realizovanih letova aviona

Naziv SK
Unos realizovanih letova aviona

Aktori SK
Aviomehaničar

Učesnici SK
Aviomehaničar i sistem

Preduslov
Sistem je funkcionalan, aviomehaničar je prijavljen na sistem pod svojim nalogom, sistem prikazuje formu za unos podataka. Šifarnici aviona i aerodroma su inicijalizovani. Učitane su liste aviona i aerodroma.

Osnovni scenario SK

1. Aviomehaničar bira avion za koji želi unos realizovanih letova. (APUSO)
2. Aviomehaničar poziva sistem da na osnovu izabranog aviona vrati relevantne podatke bitne za unos letova (poziciju aviona nakon zadnjeg realizovanog leta, kao i podatke o naletu aviona). (APSO)
3. Sistem traži podatke o avionu na osnovu zadatog kriterijuma. (SO)
4. Sistem prikazuje tražene podatke o avionu uz poruku „Sistem je spreman za unos novih letova“. (IA)
5. Aviomehaničar unosi podatke koji jednoznačno identifikuju realizovani let. (APUSO)
6. Aviomehaničar kontroliše da li je korektno unio podatke o realizovanom letu. (ANSO)
7. Aviomehaničar poziva sistem da zapamti podatke o realizovanom letu. (APSO)
8. Sistem pamti podatke o realizovanom letu. (SO)
9. Sistem prikazuje podatke o realizovanom letu uz poruku „Sistem je dodao let u evidenciju realizovanih letova“. (IA)

Alternativna scenarija

4.1 Ukoliko sistem ne može da nađe tražene podatke o avionu on aviomehaničaru prikazuje poruku „Sistem ne može da pronađe podatke o avionu“. Prekida se izvršavanje scenarija. (IA)
9.1 Ukoliko sistem ne može da zapamti podatke o realizovanom letu on aviomehaničaru prikazuje poruku „Sistem ne može da sačuva podatke o realizovanom letu“. (IA)

4.1.2.5 SK5: Slučaj korišćenja – Pregled realizovanih letova aviona

Naziv SK
Pregled realizovanih letova aviona

Aktori SK
Aviomehaničar

Učesnici SK
Aviomehaničar i sistem

Preduslov
Sistem je funkcionalan, aviomehaničar je prijavljen na sistem pod svojim nalogom, sistem prikazuje formu za unos podataka. Šifarnici aviona i aerodroma su inicijalizovani. Učitane su liste aviona i aerodroma.

Osnovni scenario SK

1. Aviomehaničar unosi vremenski raspon i bira avion za koji želi pregled realizovanih letova. (APUSO)
2. Aviomehaničar poziva sistem da na osnovu izabranog aviona i vremenskog raspona vrati realizovane letove. (APSO)
3. Sistem traži podatke o realizovanim letovima na osnovu definisanog kriterijuma. (SO)
4. Sistem vraća tražene podatke o realizovanim letovima uz poruku „Sistem je našao sledeće realizovane letove“. (IA)
5. Aviomehaničar bira konkretan let. (APUSO)
6. Aviomehaničar zahtijeva od sistema podatke o konkretnom letu. (APSO)
7. Sistem traži podatke o konkretnom letu. (SO)
8. Sistem vraća podatke o konkretnom letu uz poruku „Sistem je našao podatke o konkretnom letu“. (IA)
9. Aviomehaničar ažurira podatke za korektan let. (APUSO)
10. Aviomehaničar poziva sistem da zapamti podatke ažuriranog leta. (APSO)
11. Sistem pamti podatke o ažuriranom letu. (SO)
12. Sistem prikazuje podatke o ažuriranom letu uz poruku „Sistem je ažurirao let u evidenciji realizovanih letova“. (IA)

Alternativna scenarija

4.1 Ukoliko sistem ne može da nađe tražene podatke o realizovanim letovima on aviomehaničaru prikazuje poruku „Sistem ne može da pronađe podatke o realizovanim letovima“. Prekida se izvršavanje scenarija. (IA)

8.1 Ukoliko sistem ne može da nađe tražene podatke o konkretnom letu on aviomehaničaru prikazuje poruku „Sistem ne može da nađe podatke o konkretnom letu“. Prekida se izvršavanje scenarija. (IA)

12.1 Ukoliko sistem ne može da zapamti podatke nakon ažuriranja leta on aviomehaničaru prikazuje poruku „Sistem ne može da sačuva ažurirane podatke o realizovanom letu“. (IA)

4.1.2.6 SK6: Slučaj korišćenja – Kreiranje kartona dijela aviona

Naziv SK
Kreiranje kartona dijela aviona

Aktori SK
Aviomehaničar

Učesnici SK
Aviomehaničar i sistem

Preduslov
Sistem je funkcionalan, aviomehaničar je prijavljen na sistem pod svojim nalogom, sistem prikazuje formu za unos podataka. Šifarnici aviona i aerodroma su inicijalizovani.

Osnovni scenario SK

1. Aviomehaničar unosi PN (Part number) i SN (Serial number) avionskog dijela radi provjere da li je karton avionskog dijela već kreiran. (APUSO)
2. Aviomehaničar poziva sistem da na osnovu PN (Part number) i SN (Serial number) kao parametara provjeri da li je karton avoinskog dijela već unešen u evidenciju avionskih dijelova. (APSO)
3. Sistem traži karton avionskog dijela po zadatom kriterijumu. (SO)
4. Sistem vraća informaciju da avionski dio ne postoji u evidenciji avionskih dijelova i vraća poruku „Sistem je spreman za unos kartona avionskog dijela“. (IA)
5. Aviomehaničar unosi podatke relevantne za opis avionskog dijela kao i podatke o njegovim resursima. (APUSO)
6. Aviomehaničar kontroliše da li je korektno unio podatke u karton avionskog dijela. (ANSO)
7. Aviomehaničar poziva sistem da zapamti podatke iz kartona avionskog dijela. (APSO)
8. Sistem pamti podatke iz kartona avionskog dijela. (SO)
9. Sistem prikazuje podatke iz kartona avionskog dijela uz poruku „Sistem je dodao karton dijela u evidenciju avio dijelova“. (IA)

Alternativna scenarija

4.1 Ukoliko sistem nađe da avionski dio postoji u evidenciji avionskih dijelova sistem šalje aviomehaničaru poruku „Sistem je našao karton dijela u evidenciji avio dijelova“. Prekida se izvršavanje scenarija. (IA)

9.1 Ukoliko sistem ne može da zapamti podatke iz kartona avionskog dijela on aviomehaničaru šalje poruku „Sistem ne može da sačuva karton dijela u evidenciju avio dijelova“. (IA)

4.1.2.7 SK7: Slučaj korišćenja – Pregled kartona dijela aviona

Naziv SK
Pregled kartona dijela aviona

Aktori SK
Aviomehaničar

Učesnici SK
Aviomehaničar i sistem

Preduslov
Sistem je funkcionalan, aviomehaničar je prijavljen na sistem pod svojim nalogom, sistem prikazuje formu za unos podataka. Šifarnici aviona i aerodroma su inicijalizovani.

Osnovni scenario SK

1. Aviomehaničar unosi PN (Part number) i SN (Serial number) avionskog dijela radi pristupa kartonu avionskog dijela. (APUSO)
2. Aviomehaničar poziva sistem da na osnovu PN (Part number) i SN (Serial number) kao parametara nađe karton avionskog dijela iz evidencije avionskih dijelova. (APSO)
3. Sistem traži karton avionskog dijela po zadatom kriterijumu. (SO)
4. Sistem vraća podatke iz kartona avionskog dijela kao i poruku „Sistem je spreman za ažuriranje kartona avio dijela“. (IA)
5. Aviomehaničar ažurira podatke iz kartona avionskog dijela. (APUSO)
6. Aviomehaničar kontroliše da li je korektno unio podatke u karton avionskog dijela. (ANSO)
7. Aviomehaničar poziva sistem da zapamti podatke iz kartona avionskog dijela. (APSO)
8. Sistem pamti podatke iz kartona avionskog dijela. (SO)
9. Sistem prikazuje podatke iz kartona avionskog dijela uz poruku „Sistem je ažurirao karton dijela u evidenciji avio dijelova“. (IA)

Alternativna scenarija

4.1 Ukoliko sistem ne može da nađe tražene podatke o avionskom dijelu sistem šalje aviomehaničaru poruku „Sistem ne može da pronađe karton dijela u evidenciji avio dijelova“. Prekida se izvršavanje scenarija. (IA)

9.1 Ukoliko sistem ne može da zapamti podatke iz kartona avionskog dijela on aviomehaničaru šalje poruku „Sistem ne može da sačuva kartona dijela u evidenciju avio dijelova“. (IA)

4.1.2.8 SK8: Slučaj korišćenja – Instaliranje dijela iz magacina dijelova na avion

Naziv SK
Instaliranje dijela iz magacina dijelova na avion

Aktori SK
Aviomehaničar

Učesnici SK
Aviomehaničar i sistem

Preduslov
Sistem je funkcionalan, aviomehaničar je prijavljen na sistem pod svojim nalogom, sistem prikazuje formu za unos podataka. Šifarnici aviona i aerodroma su inicijalizovani. Učitana je lista aviona.

Osnovni scenario SK

1. Aviomehaničar unosi PN (Part number) i SN (Serial number) avionskog dijela radi pristupa kartonu avionskog dijela koji je lociran u magacinu. (APUSO)
2. Aviomehaničar poziva sistem da na osnovu PN (Part number) i SN (Serial number) kao parametara nađe avionski dio u magacinu avionskih dijelova. (APSO)
3. Sistem traži karton avionskog dijela po zadatom kriterijumu. (SO)
4. Sistem vraća podatke iz kartona avionskog dijela kao i poruku „Sistem je našao avionski dio u magacinu avionskih dijelova“. (IA)
5. Aviomehaničar bira avion na koji želi instalirati avionski dio. (APUSO)
6. Aviomehaničar poziva sistem da na osnovu izabranog aviona vrati relevantne podatke bitne za instalaciju avionskog dijela (nalet aviona). (APSO)
7. Sistem traži informacije o naletu aviona. (SO)
8. Sistem vraća informacije o naletu aviona korisniku uz poruku „Sistem je spreman za instaliranje avio dijela na avion“. (IA)
9. Aviomehaničar unosi podatke relevantne za instalaciju avionskog dijela. (APUSO)
10. Aviomehaničar kontroliše da li je korektno unio podatke o instalaciji avionskog dijela. (ANSO)
11. Aviomehaničar poziva sistem da zapamti podatke vezane za instalaciju avionskog dijela. (APSO)
12. Sistem pamti podatke vezane za instalaciju avionskog dijela. (SO)
13. Sistem prikazuje podatke vezane za instalaciju avionskog dijela uz poruku „Sistem je instalirao avio dio na avionu“. (IA)

Alternativna scenarija

4.1 Ukoliko sistem ne može da nađe tražene podatke iz kartona avionskog dijela koji je lociran u magacinu on aviomehaničaru prikazuje poruku „Sistem ne može da pronađe avionski dio u magacinu avionskih dijelova“. Prekida se izvršavanje scenarija. (IA)

8.1 Ukoliko sistem ne može da nađe tražene podatke o avionu on aviomehaničaru prikazuje poruku „Sistem ne može da pronađe podatke o avionu“. Prekida se izvršavanje scenarija. (IA)

13.1 U slučaju da sistem ne može da završi instalaciju avionskog dijela na avion on aviomehaničaru šalje poruku „Sistem nije u mogućnosti da završi instalaciju avionskog dijela na avionu“. (IA)

4.1.2.9 SK9: Slučaj korišćenja – Skidanje dijela sa aviona i slanje u magacin

Naziv SK
Skidanje dijela sa aviona i slanje u magacin

Aktori SK
Aviomehaničar

Učesnici SK
Aviomehaničar i sistem

Preduslov
Sistem je funkcionalan, aviomehaničar je prijavljen na sistem pod svojim nalogom, sistem prikazuje formu za unos podataka. Šifarnici aviona i aerodroma su inicijalizovani. Učitana je lista aviona.

Osnovni scenario SK

1. Aviomehaničar bira avion sa koga želi skinuti avionski dio. (APUSO)
2. Aviomehaničar poziva sistem da na osnovu izabranog aviona vrati sve dijelove instalirane na njega. (APSO)
3. Sistem traži dijelove instalirane na njega po zadatom kriterijumu. (SO)
4. Sistem vraća tražene podatke korisniku uz poruku „Spisak avionskih djelova sa preostalim resursima.“. (IA)
5. Aviomehaničar bira PN (Part number) i SN (Serial number) avionskog dijela radi skidanja istog sa aviona i smještanja u magacin avionskih dijelova. (APUSO)
6. Aviomehaničar poziva sistem da na osnovu PN (Part number) i SN (Serial number) vrati podatke o dijelu instaliranom na avionu. (APSO)
7. Sistem pronalazi podatke relevantne za prebacivanje dijela sa aviona u magacin avionskih dijelova. (SO)
8. Sistem vraća podatke o dijelu instaliranom na avionu uz poruku „Sistem je spreman za prebacanje dijela sa aviona u magacin.“. (IA)
9. Aviomehaničar unosi relevantne podatke bitne za skidanje dijela sa aviona. (APUSO)
10. Aviomehaničar poziva sistem da prebaci dio sa aviona u magacin avio dijelova. (APSO)
11. Sistem prebaca dio sa aviona u magacin avionskih dijelova. (SO)
12. Sistem prikazuje podatke dijela u magacinu avionskih dijelova uz poruku „Sistem je izvršio prebacanje dijela sa aviona u magacin.“. (IA)

Alternativna scenarija

4.1 Ukoliko sistem ne može da nađe tražene podatke o instaliranim dijelovima na avionu on aviomehaničaru prikazuje poruku „Sistem ne može da pronađe podatke o preostalim resursima“. Prekida se izvršavanje scenarija. (IA)

8.1 Ukoliko sistem ne može da nađe tražene podatke o instaliranom dijelu na avionu on aviomehaničaru prikazuje poruku „Sistem ne može da pronađe podatke o instaliranom dijelu“. Prekida se izvršavanje scenarija. (IA)

12.1 Ukoliko sistem ne može da izvrši prebacivanje dijela sa aviona u magacin avionskih dijelova on aviomehaničaru prikazuje poruku „Sistem ne može da izvrši prebacivanje dijela sa aviona u magacin“. (IA)

4.1.2.10 SK10: Slučaj korišćenja – Slanje dijela iz magacina u avio servis

Naziv SK
Slanje dijela iz magacina u avio servis

Aktori SK
Aviomehaničar

Učesnici SK
Aviomehaničar i sistem

Preduslov
Sistem je funkcionalan, aviomehaničar je prijavljen na sistem pod svojim nalogom, sistem prikazuje formu za unos podataka. Šifarnici aviona i aerodroma su inicijalizovani.

Osnovni scenario SK

1. Aviomehaničar unosi PN (Part number) i SN (Serial number) avionskog dijela radi pristupa kartonu avionskog dijela koji je lociran u magacinu. (APUSO)
2. Aviomehaničar poziva sistem da na osnovu PN (Part number) i SN (Serial number) kao parametara nađe avionski dio u magacinu avionskih dijelova. (APSO)
3. Sistem traži avionski dio u magacinu avionskih dijelova. (SO)
4. Sistem vraća podatke iz kartona avionskog dijela kao i poruku „Sistem je našao avionski dio u magacinu avionskih dijelova“. (IA)
5. Aviomehaničar unosi podatke relevantne za prebacanje avionskog dijela iz magacina avionskih dijelova u magacin avio servisa. (APUSO)
6. Aviomehaničar poziva sistem da izabrani avionski dio prebaci iz magacina avionskih dijelova u magacin avio servisa. (APSO)
7. Sistem prebaca avionski dio iz magacina avionskih dijelova u magacin avio servisa. (SO)
8. Sistem prikazuje podatke o prebačenom dijelu u avio servis uz poruku „Sistem je izvršio prebacanje dijela u magacin avio servisa“. (IA)

Alternativna scenarija

4.1 Ukoliko sistem ne može da nađe tražene podatke iz kartona avionskog dijela koji je lociran u magacinu on aviomehaničaru prikazuje poruku „Sistem ne može da pronađe avionski dio u magacinu avionskih dijelova“. Prekida se izvršavanje scenarija. (IA)

8.1 Ukoliko sistem ne može da izvršio prebacanje dijela iz magacina avionskih dijelova u magacin avio servisa on aviomehaničaru prikazuje poruku „Sistem ne može da prebaci avionski dio u magacin avio servisa“. (IA)

4.1.2.11 SK11: Slučaj korišćenja – Izvještavanje o preostalim resursima dijela aviona

Naziv SK
Izvještavanje o preostalim resursima dijela aviona

Aktori SK
Aviomehaničar

Učesnici SK
Aviomehaničar i sistem

Preduslov
Sistem je funkcionalan, aviomehaničar je prijavljen na sistem pod svojim nalogom, sistem prikazuje formu za unos podataka. Šifarnici aviona i aerodroma su inicijalizovani. Učitana je lista aviona.

Osnovni scenario SK

1. Aviomehaničar bira avion radi izvještavanja o preostalim resursima avionskih dijelova instaliranih na njega. (APUSO)
2. Aviomehaničar poziva sistem da na osnovu izabranog aviona vrati spisak svih dijelova sa informacijom o preostaloj količini resursa. (APSO)
3. Sistem traži informacije o preostaloj količini resursa avionskih dijelova instaliranih na avionu. (SO)
4. Sistem vraća tražene podatke korisnika uz poruku „Spisak avionskih dijelova sa preostalim resursima“. (IA)

Alternativna scenarija

4.1 Ukoliko sistem ne može da nađe tražene podatke o preostalim resursima on aviomehaničaru prikazuje poruku „Sistem ne može da pronađe podatke o preostalim resursima“.

4.1.2.12 SK12: Slučaj korišćenja – Izvještavanje o isteku resursa avionskog dijela (resurs u satima)

Naziv SK
Izvještavanje o isteku resursa avionskog dijela (resurs u satima)

Aktori SK
Aviomehaničar

Učesnici SK
Aviomehaničar i sistem

Preduslov
Sistem je funkcionalan, aviomehaničar je prijavljen na sistem pod svojim nalogom, sistem prikazuje formu za unos podataka. Šifarnici aviona i aerodroma su inicijalizovani. Učitana je lista aviona.

Osnovni scenario SK

1. Aviomehaničar bira avion radi izvještavanja o isteku resursa avionskih dijelova instaliranih na njemu. (APUSO)
2. Aviomehaničar poziva sistem da na osnovu izabranog aviona vrati sve informacije o izabranom avionu. (APSO)
3. Sistem traži informacije o avionu. (SO)
4. Sistem vraća tražene podatke korisniku. (IA)
5. Aviomehaničar unosi informacije o dnevnom broju sati leta aviona, kao i broju sati za servis. (APUSO)
6. Aviomehaničar poziva sistem da na osnovu unešenih vrijednosti vrati sve dijelove aviona koji zadovoljavaju navedeni uslov. (APSO)
7. Sistem traži informacije o preostaloj količini resursa avionskih dijelova instaliranih na avionu. (SO)
8. Sistem vraća tražene podatke korisnika uz poruku „Sistem je našao sledeće dijelove!“. (IA)

Alternativna scenarija

4.1 Ukoliko sistem ne može da nađe tražene podatke o avionu on aviomehaničaru prikazuje poruku „Sistem ne može da pronađe podatke o avionu.“. Prekida se izvršavanje scenarija. (IA)

8.1 Ukoliko sistem ne može da nađe tražene dijelove aviona on aviomehaničaru prikazuje poruku „Sistem ne može da pronađe podatke o preostalim resursima“. (IA)

4.1.2.13 SK13: Slučaj korišćenja – Izvještavanje o isteku resursa avionskog dijela (resurs u ciklusima)

Naziv SK
Izvještavanje o isteku resursa avionskog dijela (resurs u ciklusima)

Aktori SK
Aviomehaničar

Učesnici SK
Aviomehaničar i sistem

Preduslov
Sistem je funkcionalan, aviomehaničar je prijavljen na sistem pod svojim nalogom, sistem prikazuje formu za unos podataka. Šifarnici aviona i aerodroma su inicijalizovani. Učitana je lista aviona.

Osnovni scenario SK

1. Aviomehaničar bira avion radi izvještavanja o isteku resursa avionskih dijelova instaliranih na njemu. (APUSO)
2. Aviomehaničar poziva sistem da na osnovu izabranog aviona vrati sve informacije o izabranom avionu. (APSO)
3. Sistem traži informacije o avionu. (SO)
4. Sistem vraća tražene podatke korisniku. (IA)
5. Aviomehaničar unosi informacije o dnevnom broju ciklusa leta aviona, kao i broju ciklusa za servis. (APUSO)
6. Aviomehaničar poziva sistem da na osnovu unešenih vrijednosti vrati sve dijelove aviona koji zadovoljavaju navedeni uslov. (APSO)
7. Sistem traži informacije o preostaloj količini resursa avionskih dijelova instaliranih na avionu. (SO)
8. Sistem vraća tražene podatke korisnika uz poruku „Sistem je našao sledeće dijelove!“. (IA)

Alternativna scenarija

4.1 Ukoliko sistem ne može da nađe tražene podatke o avionu on aviomehaničaru prikazuje poruku „Sistem ne može da pronađe podatke o avionu.“. Prekida se izvršavanje scenarija. (IA)

8.1 Ukoliko sistem ne može da nađe tražene dijelove aviona on aviomehaničaru prikazuje poruku „Sistem ne može da pronađe podatke o preostalim resursima“. (IA)

4.1.2.14 SK14: Slučaj korišćenja – Izvještavanje o isteku resursa avionskog dijela (resurs u danima)

Naziv SK
Izvještavanje o isteku resursa avionskog dijela (resurs u danima)

Aktori SK
Aviomehaničar

Učesnici SK
Aviomehaničar i sistem

Preduslov
Sistem je funkcionalan, aviomehaničar je prijavljen na sistem pod svojim nalogom, sistem prikazuje formu za unos podataka. Šifarnici aviona i aerodroma su inicijalizovani. Učitana je lista aviona.

Osnovni scenario SK

1. Aviomehaničar bira avion radi izvještavanja o isteku resursa avionskih dijelova instaliranih na njemu. (APUSO)
2. Aviomehaničar poziva sistem da na osnovu izabranog aviona vrati sve informacije o izabranom avionu. (APSO)
3. Sistem traži informacije o avionu. (SO)
4. Sistem vraća tražene podatke korisniku. (IA)
5. Aviomehaničar unosi informacije o broju dana za servis. (APUSO)
6. Aviomehaničar poziva sistem da na osnovu unešenih vrijednosti vrati sve dijelove aviona koji zadovoljavaju navedeni uslov. (APSO)
7. Sistem traži informacije o preostaloj količini resursa avionskih dijelova instaliranih na avionu. (SO)
8. Sistem vraća tražene podatke korisnika uz poruku „Sistem je našao sledeće dijelove! “. (IA)

Alternativna scenarija

4.1 Ukoliko sistem ne može da nađe tražene podatke o avionu on aviomehaničaru prikazuje poruku „Sistem ne može da pronađe podatke o avionu.“. Prekida se izvršavanje scenarija. (IA)

8.1 Ukoliko sistem ne može da nađe tražene dijelove aviona on aviomehaničaru prikazuje poruku „Sistem ne može da pronađe podatke o preostalim resursima“. (IA)

4.1.2.15 SK15: Slučaj korišćenja – Servisiranje i produženje resursa dijela aviona

Naziv SK
Servisiranje i produženje resursa dijela aviona

Aktori SK
Avio servis

Učesnici SK
Avio servis i sistem

Preduslov
Sistem je funkcionalan, Avio servis je prijavljen na sistem pod svojim nalogom, sistem prikazuje formu za unos podataka.

Osnovni scenario SK

1. Avio servis bira magacin neservisiranih dijelova radi testiranja i reparacije avionskog dijela. (APUSO)
2. Avio servis poziva sistem da iz magacin neservisiranih dijelova vrati sve neservisirane dijelove radi testiranja i reparacije avionskog dijela. (APSO)
3. Sistem pravi spisak svih neservisiranih dijelova aviona. (SO)
4. Sistem vraća spisak neservisiranih dijelova aviona uz poruku „Sistem je našao sledeće neservisirane avio dijelove“. (IA)
5. Avio servis bira PN (Part number) i SN (Serial number) avionskog dijela radi testiranja i reparacije istog. (APUSO)
6. Avio servis poziva sistem da vrati informacije o selektovanom dijelu iz liste neservisiranih djelova. (APSO)
7. Sistem izvršava akciju selektovanja dijelu iz liste neservisiranih djelova. (SO)
8. Sistem vraća avio servisu poruku „Sistem je spreman da sačuva rezultat inspekcije dijela“. (IA)
9. Nakon izvršenih testiranja avio servis izdaje sertifikat o produženju resursa avionskom dijelu ili potvrdu da se dio ne može servisirati. (APUSO)
10. Avio servis poziva sistem da podatke vezane za inspekciju avionskog dijela sačuva i prebaci dio u magacin servisiranih dijelova. (APSO)
11. Sistem izvršava akciju prebacanja dijela u magacin servisiranih dijelova. (SO)
12. Sistem prikazuje podatke o servisiranom dijelu uz „Sistem je prebacio dio u magacin servisiranih avio dijelova“. (IA)

Alternativna scenarija

4.1 Ukoliko sistem ne može da nađe neservisirane dijelove aviona on avio servisu prikazuje poruku „Sistem ne može da pronađe neservisirane dijelove aviona“. Prekida se izvršavanje scenarija. (IA)
8.1 Ukoliko sistem ne može da nađe neservisirani dio aviona on avio servisu prikazuje poruku „Sistem ne može da pronađe neservisirani dio aviona“. Prekida se izvršavanje scenarija. (IA)
12.1 Ukoliko sistem ne može da zapamti podatke o servisiranju i prebacanju dijela u magacin servisiranih dijelova on avio servisu prikazuje poruku „Sistem ne može da prebaci dio u magacin servisiranih avionskih dijelova“.

4.1.2.16 SK16: Slučaj korišćenja – Vraćanje dijela nakon popravke u magacin dijelova

Naziv SK
Vraćanje dijela aviona nakon reparacije u magacin dijelova

Aktori SK
Avio servis

Učesnici SK
Avio servis i sistem

Preduslov
Sistem je funkcionalan, avio servis je prijavljen na sistem pod svojim nalogom, sistem prikazuje formu za unos podataka.

Osnovni scenario SK

1. Avio servis bira magacin servisiranih dijelova radi vraćanja avionskog dijela u magacin pošiljaoca. (APUSO)
2. Avio servis poziva sistem da iz magacin servisiranih dijelova vrati sve servisirane dijelove radi vraćanja avionskog dijela u magacin pošiljaoca. (APSO)
3. Sistem pravi spisak svih servisiranih dijelova aviona. (SO)
4. Sistem vraća spisak servisiranih dijelova aviona uz poruku „Sistem je našao sledeće servisirane avio dijelove“. (IA)
5. Avio servis bira PN (Part number) i SN (Serial number) avionskog dijela radi slanja istog u magacin avionskih dijelova. (APUSO)
6. Avio servis poziva sistem da na osnovu PN (Part number) i SN (Serial number) kao parametara vrati informacije o selektovanom dijelu iz liste servisiranih djelova. (APSO)
7. Sistem izvršava akciju selektovanja dijelu iz liste servisiranih djelova. (SO)
8. Sistem vraća Avio servisu poruku „Sistem je spreman da prebaci dio u magacin avio dijelova“. (IA)
9. Avio servis unosi relevantne podatke bitne za prebacanje avionskog dijela iz magacina avio servisa u magacin avio djelova. (APUSO)
10. Avio servis poziva sistem da prebaci dio u magacin avionskih dijelova. (APSO)
11. Sistem izvršava akciju prebacanja dijela u magacin avionskih dijelova. (SO)
12. Sistem prikazuje podatke o dijelu koji se vraća u magacin avio dijelova uz poruku „Sistem je izvršio prebacanje dijela u magacin avio dijelova“. (IA)

Alternativna scenarija

4.1 Ukoliko sistem ne može da nađe servisirane dijelove avionu on avio servisu prikazuje poruku „Sistem ne može da pronađe servisirane dijelove aviona“. Prekida se izvršavanje scenarija. (IA)
8.1 Ukoliko sistem ne može da nađe servisirani dio avionu on avio servisu prikazuje poruku „Sistem ne može da pronađe servisirani dio aviona“. Prekida se izvršavanje scenarija. (IA)
12.1 Ukoliko sistem ne može da izvrši prebacivanje dijela iz avio servisa u magacin avionskih dijelova on avio servisu prikazuje poruku „Sistem ne može da prebaci dio iz avio servisa u magacin avio dijelova“.

4.1.2.17 SK17: Slučaj korišćenja – Prikaz kretanja dijela kroz sistem

Naziv SK
Prikaz kretanja dijela kroz sistem

Aktori SK
Aviomehaničar

Učesnici SK
Aviomehaničar i sistem

Preduslov
Sistem je funkcionalan, aviomehaničar je prijavljen na sistem pod svojim nalogom, sistem prikazuje formu za unos podataka. Šifarnici aviona i aerodroma su inicijalizovani.

Osnovni scenario SK

1. Aviomehaničar unosi PN (Part number) i SN (Serial number) avionskog dijela radi pristupa kartonu avionskog dijela. (APUSO)
2. Aviomehaničar poziva sistem da na osnovu PN (Part number) i SN (Serial number) kao parametara nađe karton kretanja avionskog dijela iz evidencije avionskih dijelova. (APSO)
3. Sistem traži karton kretanja avionskog dijela po zadatom kriterijumu. (SO)
4. Sistem vraća podatke kretanja avionskog dijela kao i poruku „Sistem je našao istoriju dijela!“. (IA)

Alternativna scenarija

4.1 Ukoliko sistem ne može da nađe tražene podatke o avionskom dijelu sistem šalje aviomehaničaru poruku „Sistem ne može da pronađe karton dijela u evidenciji avio dijelova“. (IA)

4.2 FAZA ANALIZE

U fazi analize opisujemo logičku strukturu i ponašanje softvera. Zapravo, rezultat faze analize jeste poslovna logika softverskog sistema. Ponašanje softverskog sistema se opisuje pomoću dijagrama sekvenci i sistemskih operacija, dok se struktura sistema opisuje pomoću konceptualnog i relacionog modela.

4.2.1 Sistemski dijegram sekvenci

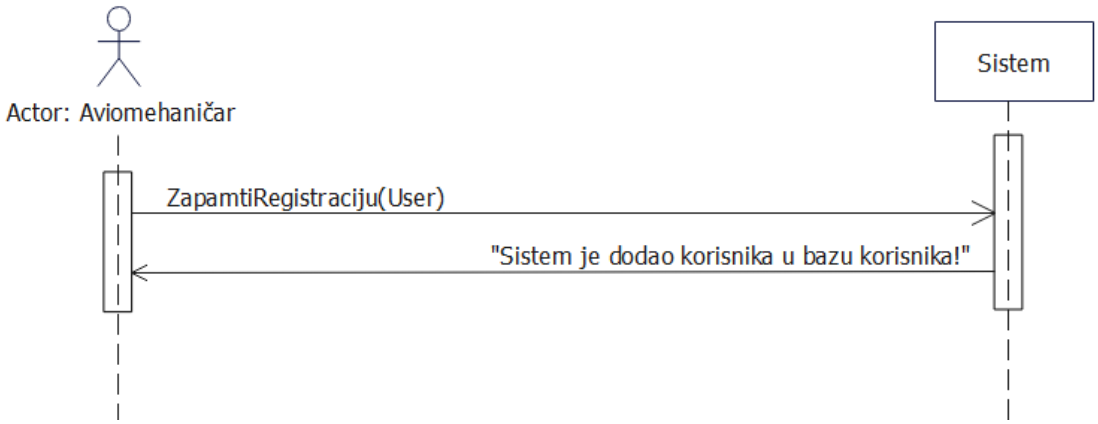
Ponašanje softverskog sistema prikazujemo putem sistemskih dijagrama sekvenci, gdje ćemo za svaki slučaj korišćenja, uočen u fazi prikupljanja zahtjeva, dati dijagram sekvenci. Sistemski dijagram sekvenci treba da prikaže interakciju između aktora i sistema, putem aktivnosti u određenom redosledu.

Na dijagramu sekvenci, aktor ne komunicira sa sistemom direkto, već preko posrednika (forma). Prilikom crtanja sekvencnih dijagrama, izostavićemo prikaz forme, ali se njeno prisustvo u komunikaciji podrazumeva.

4.2.1.1 SK1: Slučaj korišćenja – Registracija korisnika na sistem

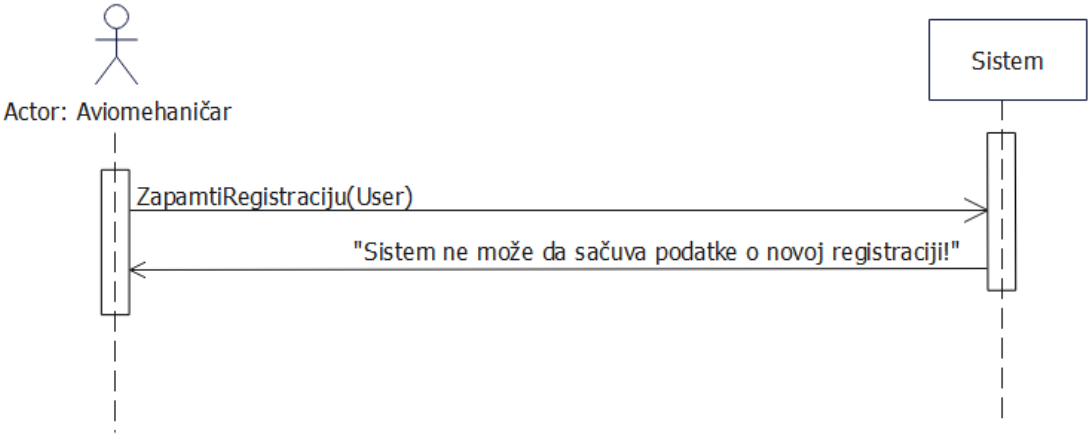
Osnovni scenario SK

1. Aviomehaničar poziva sistem da zapamti podatke korisničkog naloga. (APSO)
2. Sistem prikazuje poruku „Sistem je dodao korisnika u bazu korisnika!“. (IA)



Alternativna scenarija

2.1 Ukoliko sistem ne uspije da kreira korisnički nalog, sistem šalje aviomehaničaru poruku „Sistem ne može da sačuva podatke o novoj registraciji!“. (IA)



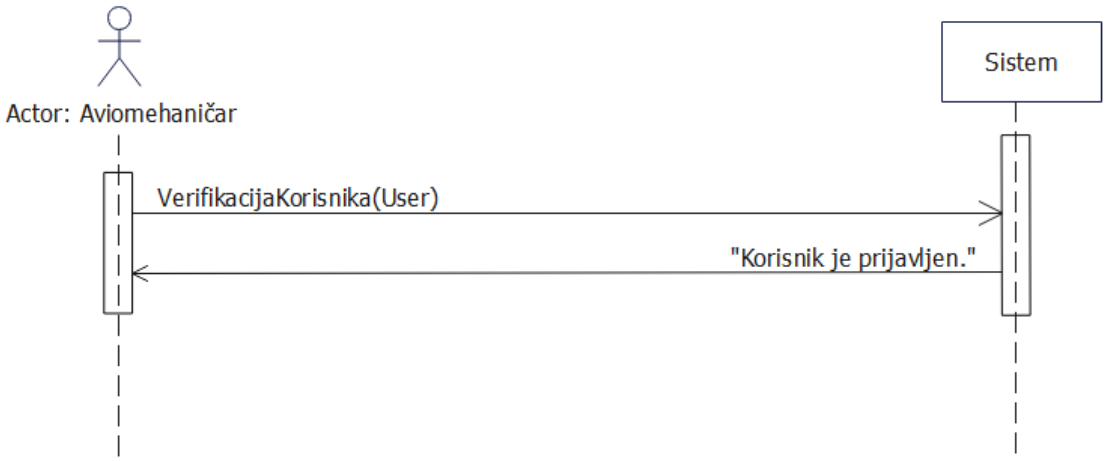
Sa navedenih sekvencijskih dijagrama uočavaju se sledeće sistemske operacije:

- Signal ZapamtiRegistraciju(User)

4.2.1.2 SK2: Slučaj korišćenja – Prijavljivanje korisnika na sistem

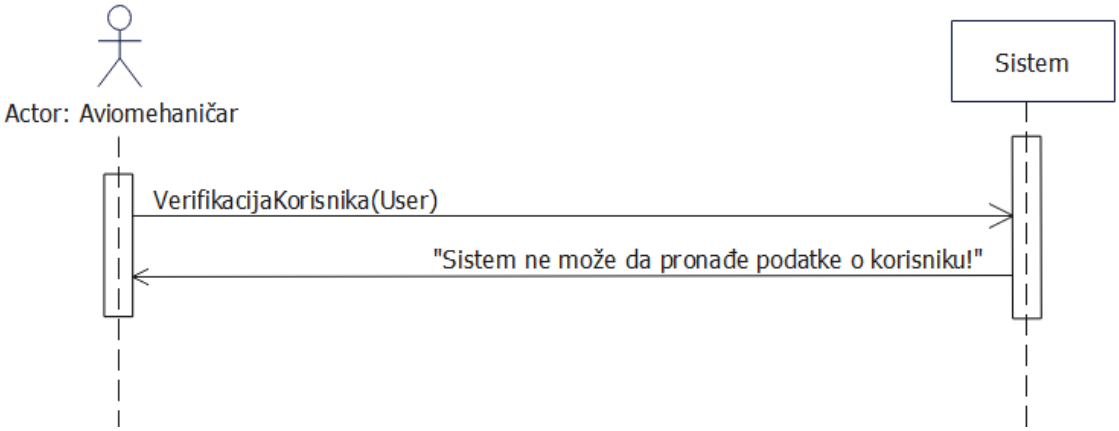
Osnovni scenario SK

1. Aviomehaničar poziva sistem da na osnovu korisničkog imena i lozinke kao parametara provjeri da li korisnik postoji u sistemu i ako postoji da ga prijavi na sistem. (APSO)
2. Sistem prijavljuje korisnika na system uz poruku „Korisnik je prijavljen.“. (IA)



Alternativna scenarija

2.1 Ukoliko sistem ne uspije da prijavi korisnika, sistem šalje aviomehaničaru poruku „Sistem ne može da pronađe podatke o korisniku“. (IA)



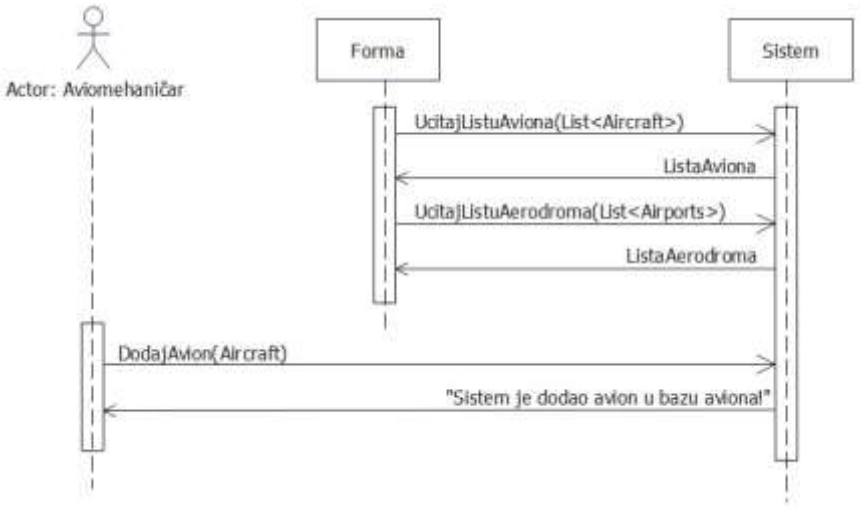
Sa navedenih sekvencijskih dijagrama uočavaju se sledeće sistemske operacije:

- Signal VerifikacijaKorisnika(User)

4.2.1.3 SK3: Slučaj korišćenja – Unos aviona

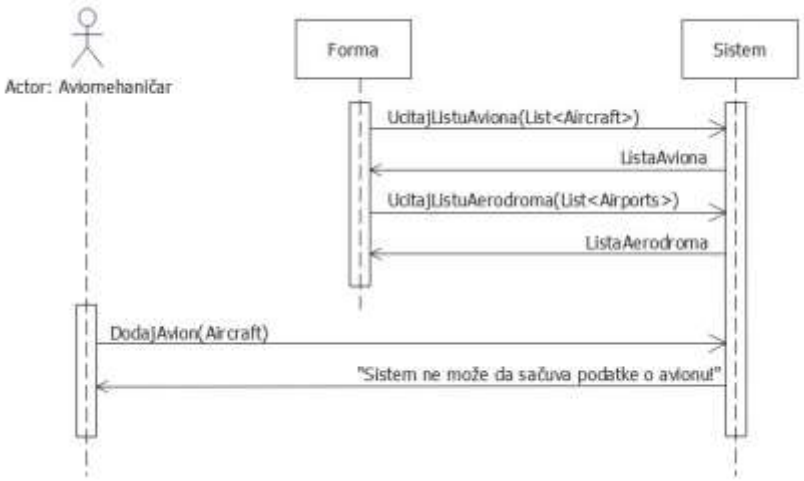
Osnovni scenario SK

1. Forma poziva sistem da učita listu aviona. (APSO)
2. Sistem vraća formi listu aviona. (IA)
3. Forma poziva sistem da učita listu aerodroma. (APSO)
4. Sistem vraća formi listu aerodroma. (IA)
5. Aviomehaničar poziva sistem da na osnovu unijetih podataka sačuva avion. (APSO)
6. Sistem prikazuje poruku „Sistem je dodao avion u bazu aviona!“. (IA)



Alternativna scenarija

6.1 Ukoliko sistem ne može da sačuva podatke o avionu on aviomehaničaru prikazuje poruku „Sistem ne može da sačuva podatke o avionu“. (IA)



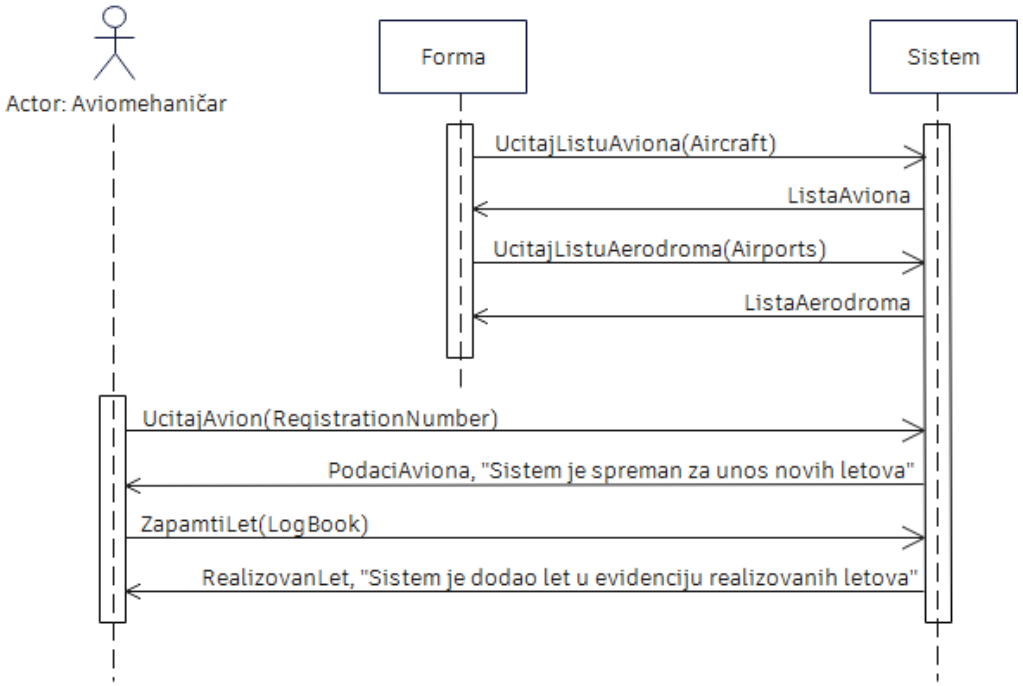
Sa navedenih sekvencijskih dijagrama uočavaju se sledeće sistemske operacije:

- Signal UcitajListuAviona(Aircraft)
- Signal UcitajListuAerodroma(Airports)
- Signal DodajAvion(Aircraft)

4.2.1.4 SK4: Slučaj korišćenja – Unos realizovanih letova aviona

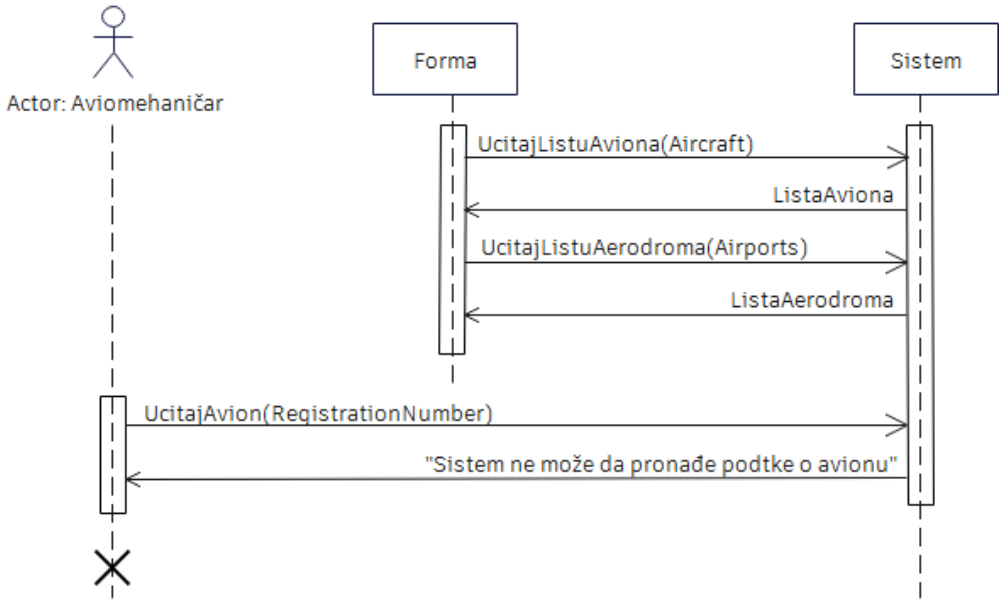
Osnovni scenario SK

1. Forma poziva sistem da učita listu aviona. (APSO)
2. Sistem vraća formi listu aviona. (IA)
3. Forma poziva sistem da učita listu aerodroma. (APSO)
4. Sistem vraća formi listu aerodroma. (IA)
5. Aviomehaničar poziva sistem da na osnovu izabranog aviona vrati relevantne podatke bitne za unos letova (poziciju aviona nakon zadnjeg realizovanog leta, kao i podatke o naletu aviona). (APSO)
6. Sistem prikazuje tražene podatke o avionu uz poruku „Sistem je spreman za unos novih letova“. (IA)
7. Aviomehaničar poziva sistem da zapamti podatke o realizovanom letu. (APSO)
8. Sistem prikazuje podatke o realizovanom letu uz poruku „Sistem je dodao let u evidenciju realizovanih letova“. (IA)

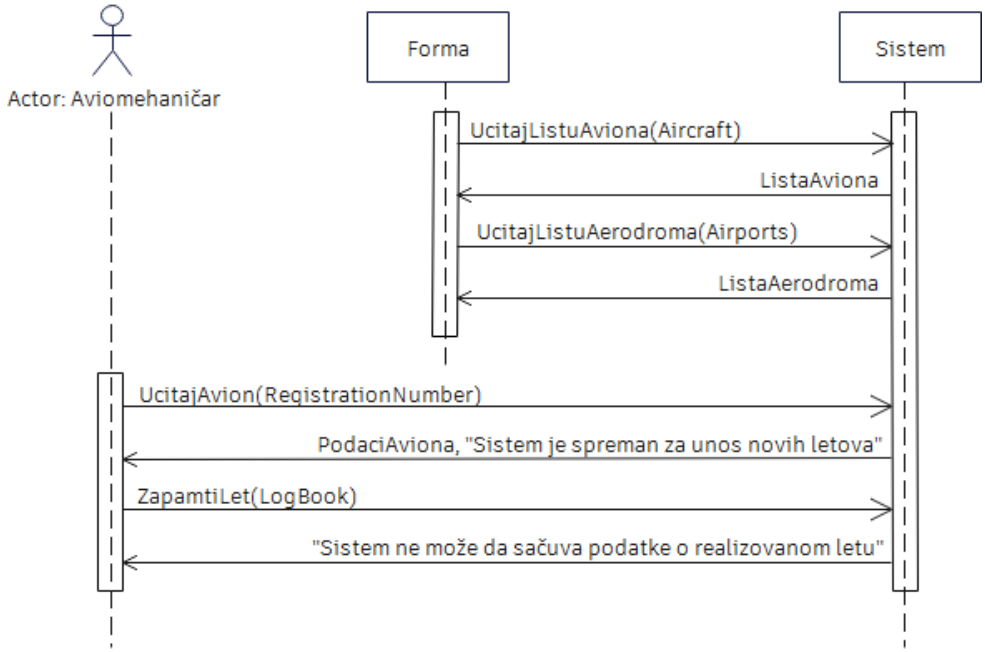


Alternativna scenarija

6.1 Ukoliko sistem ne može da nađe tražene podatke o avionu on aviomehaničaru prikazuje poruku „Sistem ne može da pronađe podatke o avionu“. Prekida se izvršavanje scenarija. (IA)



8.1 Ukoliko sistem ne može da zapamti podatke o realizovanom letu on aviomehaničaru prikazuje poruku „Sistem ne može da sačuva podatke o realizovanom letu“. (IA)



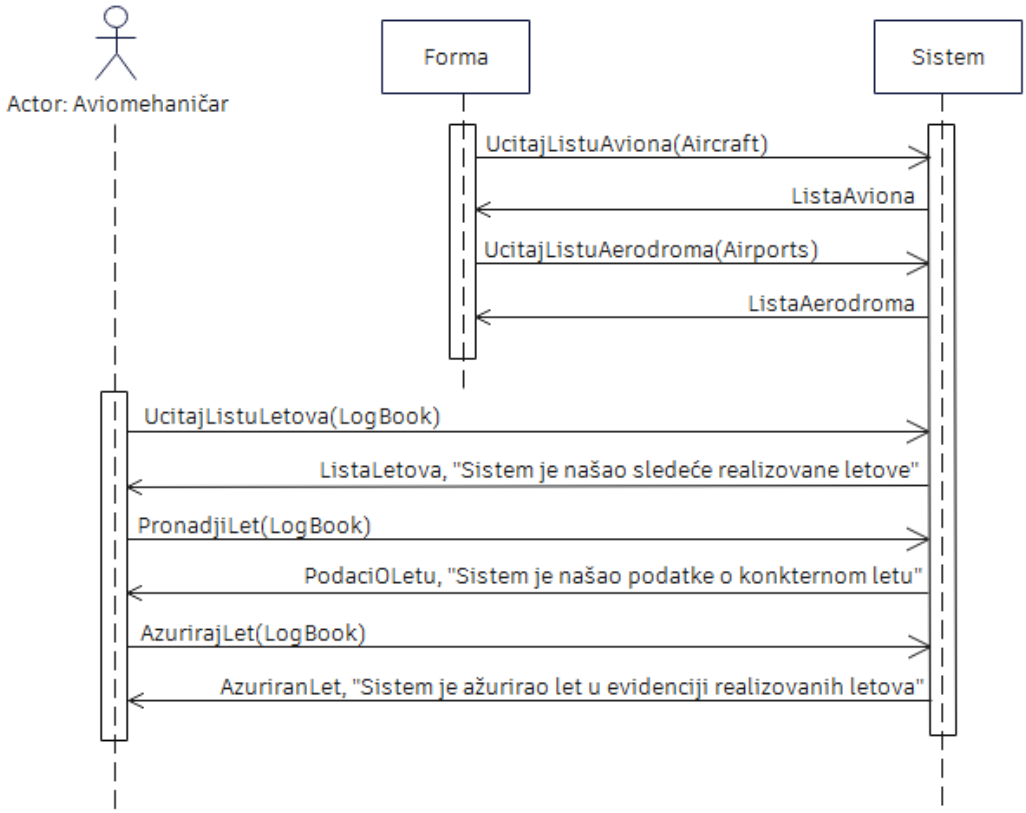
Sa navedenih sekvencijskih dijagrama uočavaju se sledeće sistemske operacije:

- Signal UcitajListuAviona(Aircraft)
- Signal UcitajListuAerodroma(Airports)
- Signal UcitajAvion(Aircraft)
- Signal ZapamtiLet(LogBook)

4.2.1.5 SK5: Slučaj korišćenja – Pregled realizovanih letova aviona

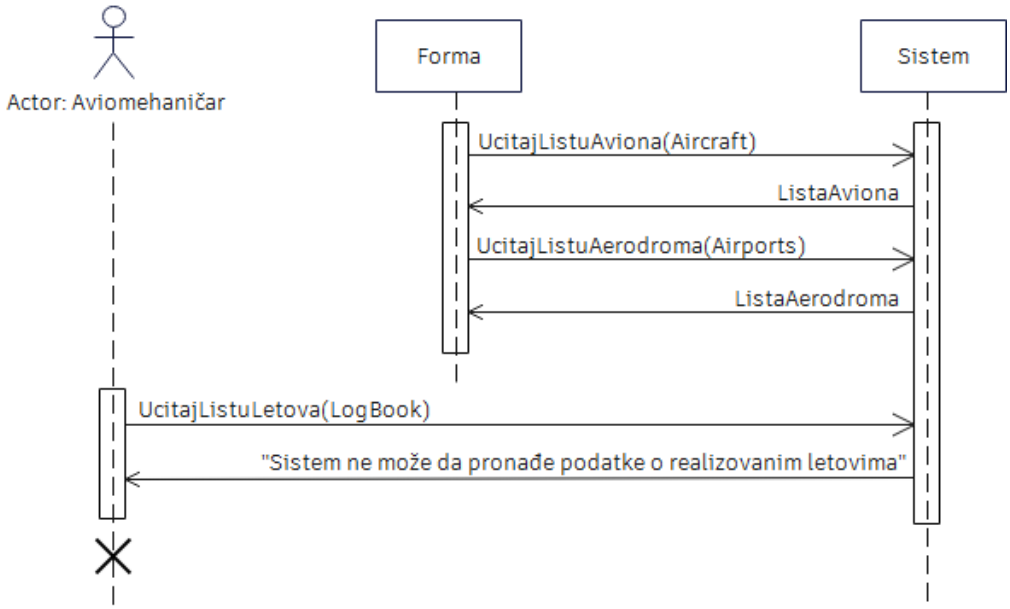
Osnovni scenario SK

1. Forma poziva sistem da učita listu aviona. (APSO)
2. Sistem vraća formi listu aviona. (IA)
3. Forma poziva sistem da učita listu aerodroma. (APSO)
4. Sistem vraća formi listu aerodroma. (IA)
5. Aviomehaničar poziva sistem da na osnovu izabranog aviona i vremenskog raspona vrati realizovane letove. (APSO)
6. Sistem vraća tražene podatke o realizovanim letovima uz poruku „Sistem je našao sledeće realizovane letove“. (IA)
7. Aviomehaničar zahtijeva od sistema podatke o konkretnom letu. (APSO)
8. Sistem vraća podatke o konkretnom letu uz poruku „Sistem je našao podatke o konkretnom letu“. (IA)
9. Aviomehaničar poziva sistem da zapamti podatke ažuriranog leta. (APSO)
10. Sistem prikazuje podatke o ažuriranom letu uz poruku „Sistem je ažurirao let u evidenciji realizovanih letova“. (IA)

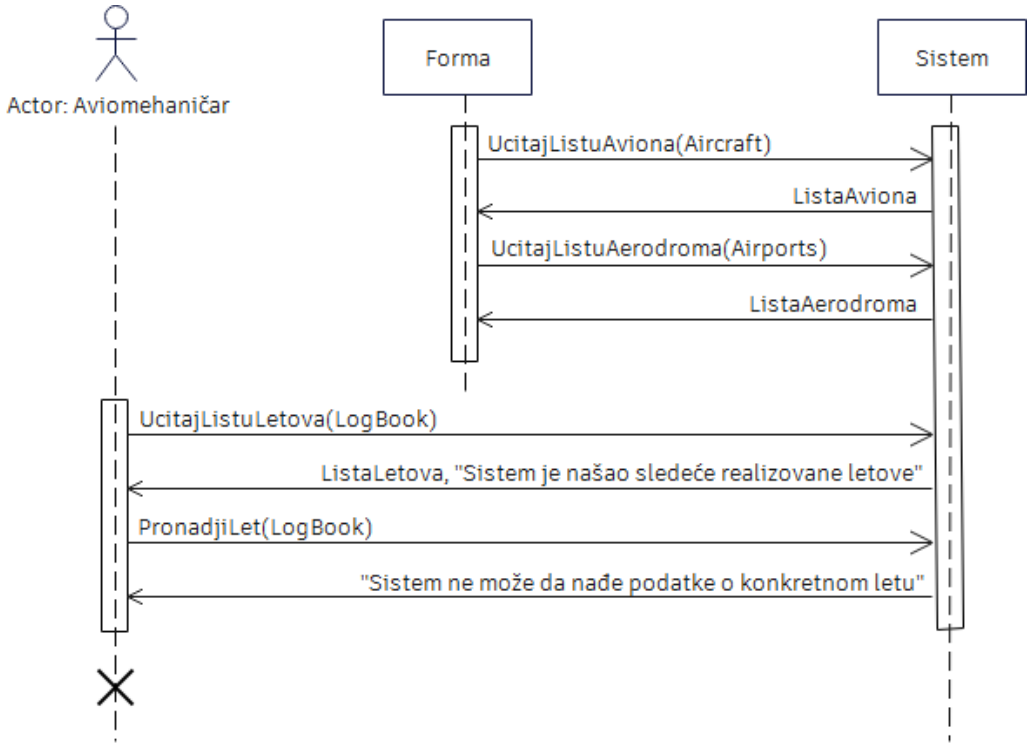


Alternativna scenarija

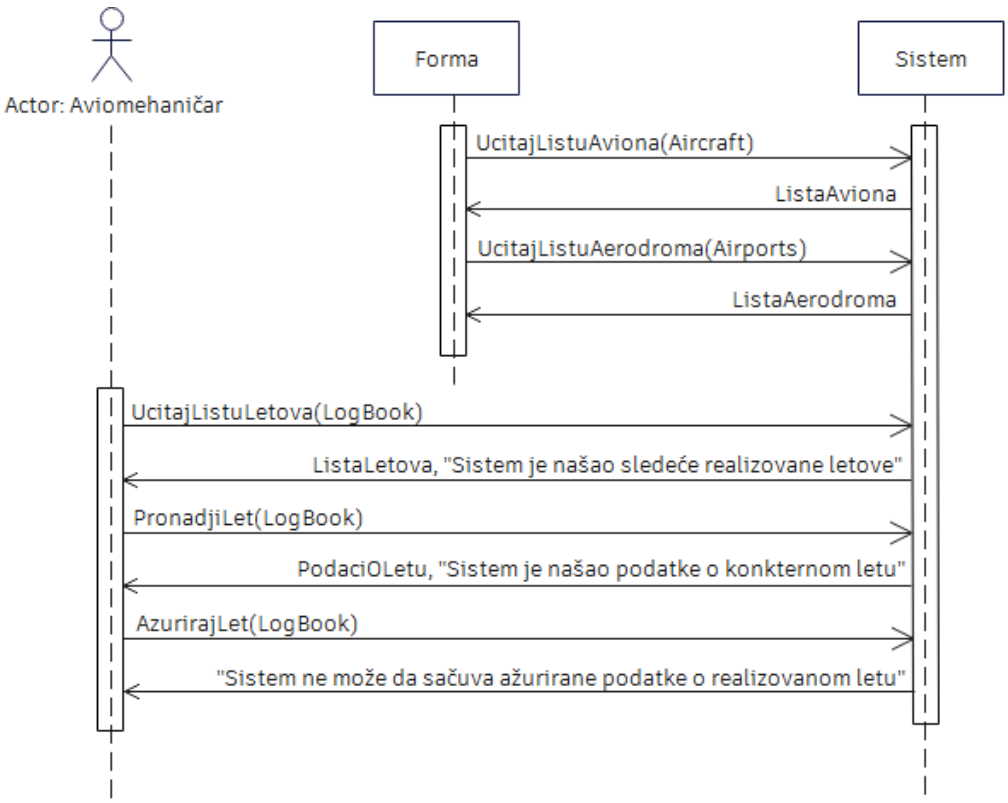
6.1 Ukoliko sistem ne može da nađe tražene podatke o realizovanim letovima on aviomehaničaru prikazuje poruku „Sistem ne može da pronađe podatke o realizovanim letovima“. Prekida se izvršavanje scenarija. (IA)



8.1 Ukoliko sistem ne može da nađe tražene podatke o konkretnom letu on aviomehaničaru prikazuje poruku „Sistem ne može da nađe podatke o konkretnom letu“. Prekida se izvršavanje scenarija. (IA)



10.1 Ukoliko sistem ne može da zapamti podatke nakon ažuriranja leta on aviomehaničaru prikazuje poruku Sistem ne može da sačuva ažurirane podatke o realizovanom letu“. (IA)



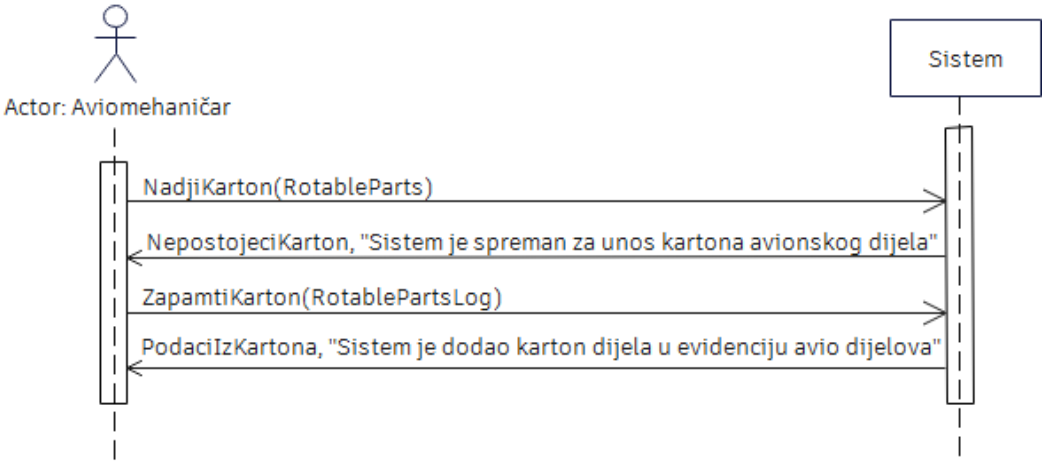
Sa navedenih sekvencijskih dijagrama uočavaju se sledeće sistemske operacije:

- Signal UcitajListuAviona(Aircraft)
- Signal UcitajListuAerodroma(Airports)
- Signal UcitajListuLetova(LogBook)
- Signal PronadjiLet(LogBook)
- Signal AzurirajLet(LogBook)

4.2.1.6 SK6: Slučaj korišćenja – Kreiranje kartona dijela aviona

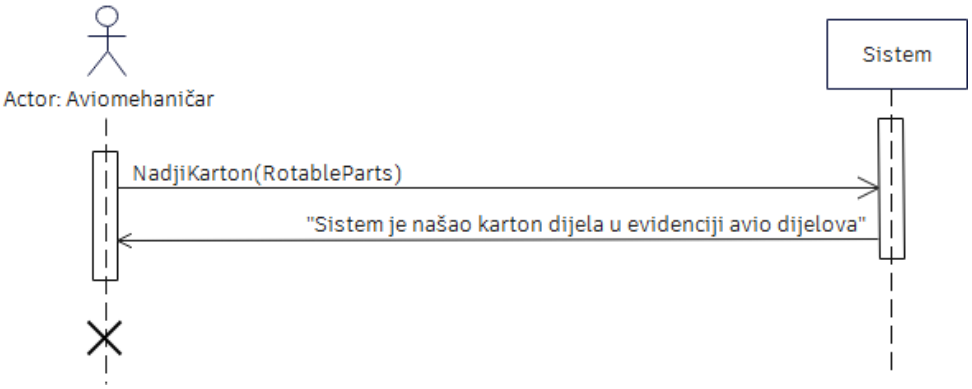
Osnovni scenario SK

1. Aviomehaničar poziva sistem da na osnovu PN (Part number) i SN (Serial number) kao parametara provjeri da li je karton avoinskog dijela već unešen u evidenciju avionskih dijelova. (APSO)
2. Sistem vraća informaciju da avionski dio ne postoji u evidenciji avionskih dijelova i vraća poruku „Sistem je spreman za unos kartona avionskog dijela“. (IA)
3. Aviomehaničar poziva sistem da zapamti podatke iz kartona avionskog dijela. (APSO)
4. Sistem prikazuje podatke iz kartona avionskog dijela uz poruku „Sistem je dodao karton dijela u evidenciju avio dijelova“. (IA)

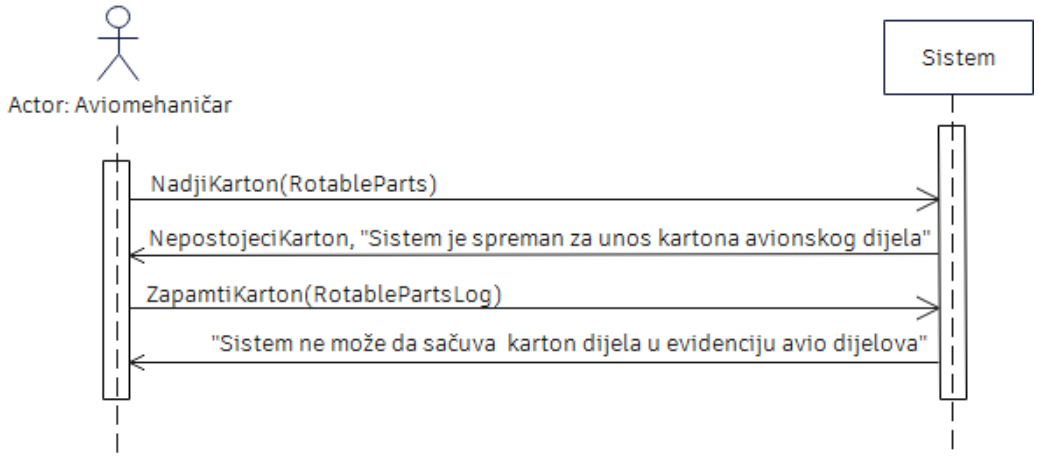


Alternativna scenarija

2.1 Ukoliko sistem nađe da avionski dio postoji u evidenciji avionskih dijelova sistem šalje aviomehaničaru poruku „Sistem je našao karton dijela u evidenciji avio dijelova“. Prekida se izvršavanje scenarija. (IA)



4.1 Ukoliko sistem ne može da zapamti podatke iz kartona avionskog dijela on aviomehaničaru šalje poruku „Sistem ne može da sačuva karton dijela u evidenciju avio dijelova“. (IA)



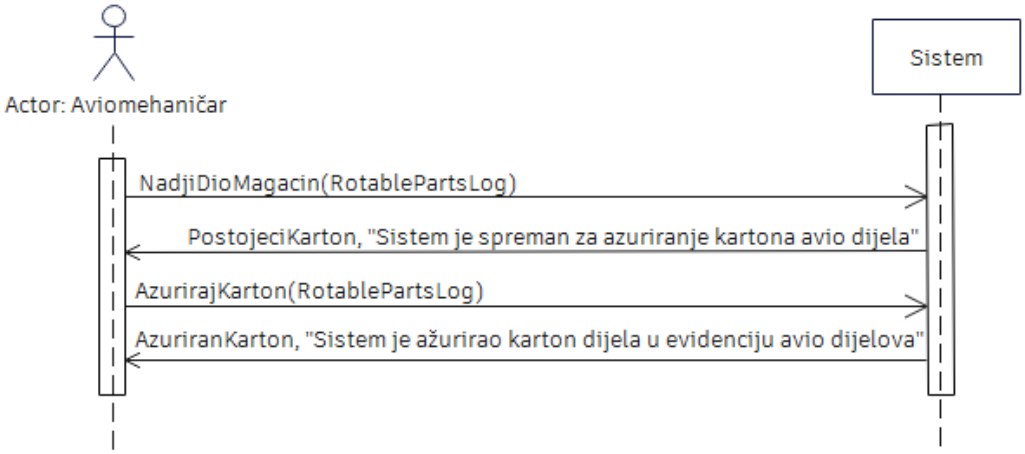
Sa navedenih sekvencijskih dijagrama uočavaju se sledeće sistemske operacije:

- Signal NadjiKarton(RotableParts)
- Signal ZapamtiKarton(RotablePartsLog)

4.2.1.7 SK7: Slučaj korišćenja – Pregled kartona dijela aviona

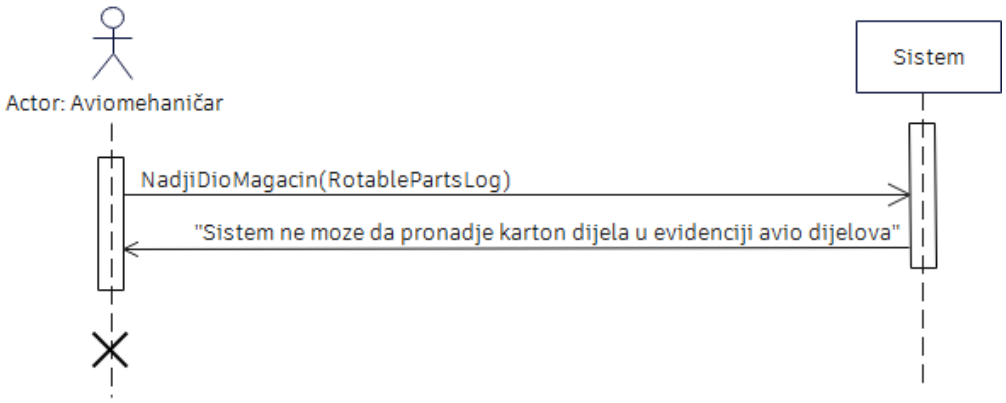
Osnovni scenario SK

1. Aviomehaničar poziva sistem da na osnovu PN (Part number) i SN (Serial number) kao parametara nađe karton avionskog dijela iz evidencije avionskih dijelova. (APSO)
2. Sistem vraća podatke iz kartona avionskog dijela kao i poruku „Sistem je spreman za ažuriranje kartona avio dijela“. (IA)
3. Aviomehaničar poziva sistem da zapamti podatke iz kartona avionskog dijela. (APSO)
4. Sistem prikazuje podatke iz kartona avionskog dijela uz poruku „Sistem je ažurirao karton dijela u evidenciji avio dijelova“. (IA)

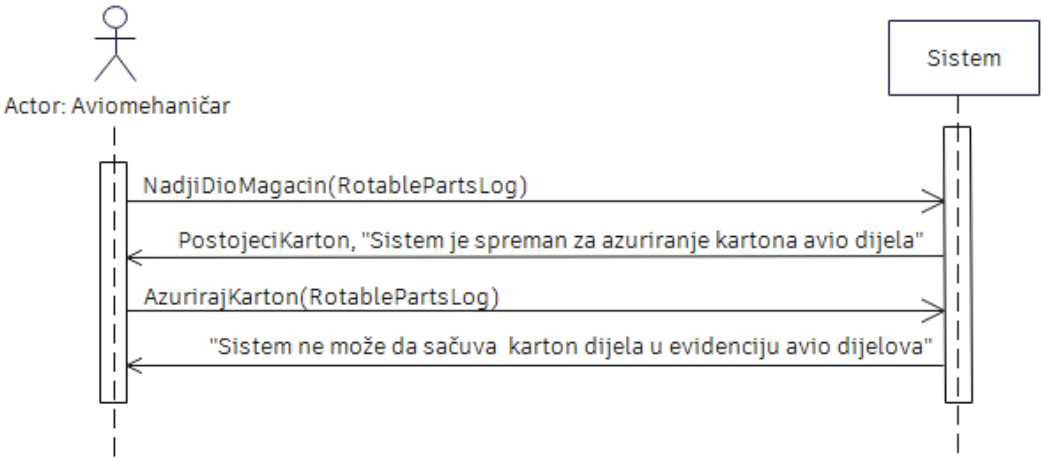


Alternativna scenarija

2.1 Ukoliko sistem ne može da nađe tražene podatke o avionskom dijelu sistem šalje aviomehaničaru poruku „Sistem ne može da pronađe karton dijela u evidenciji avio dijelova“. Prekida se izvršavanje scenarija. (IA)



4.1 Ukoliko sistem ne može da zapamti podatke iz kartona avionskog dijela on aviomehaničaru šalje poruku „Sistem ne može da sačuva kartona dijela u evidenciju avio dijelova“. (IA)



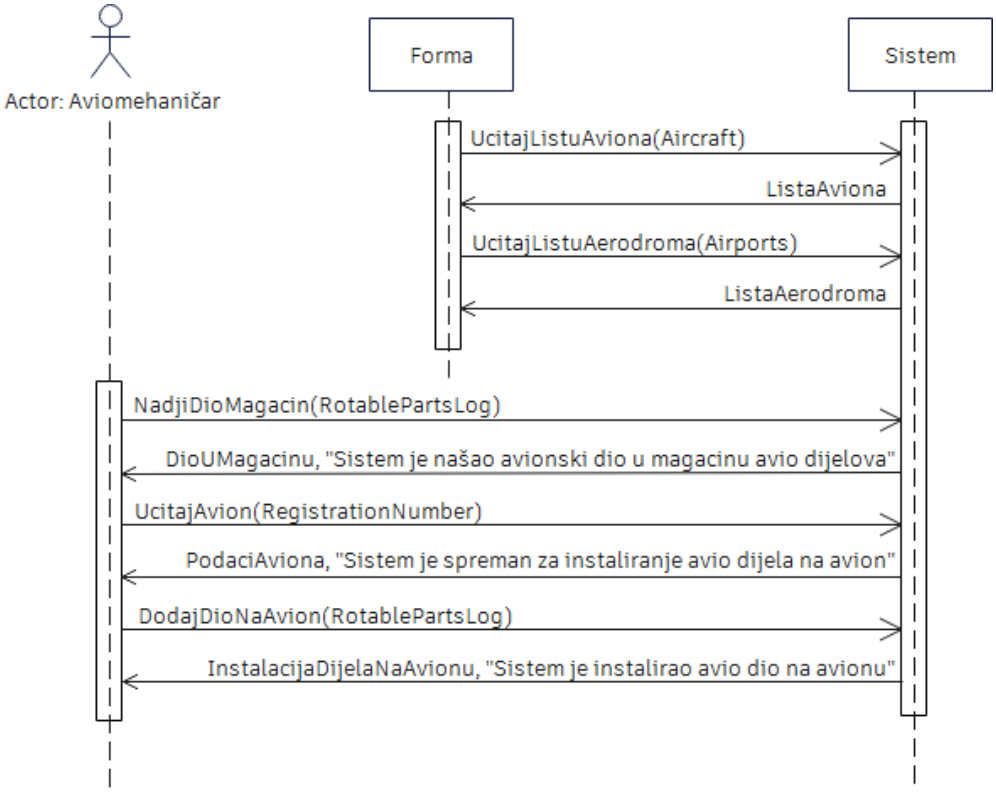
Sa navedenih sekvencijskih dijagrama uočavaju se sledeće sistemske operacije:

- Signal NadjiDioMagacin(RotablePartsLog)
- Signal AzurirajKarton(RotablePartsLog)

4.2.1.8 SK8: Slučaj korišćenja – Instaliranje dijela iz magacina dijelova na avion

Osnovni scenario SK

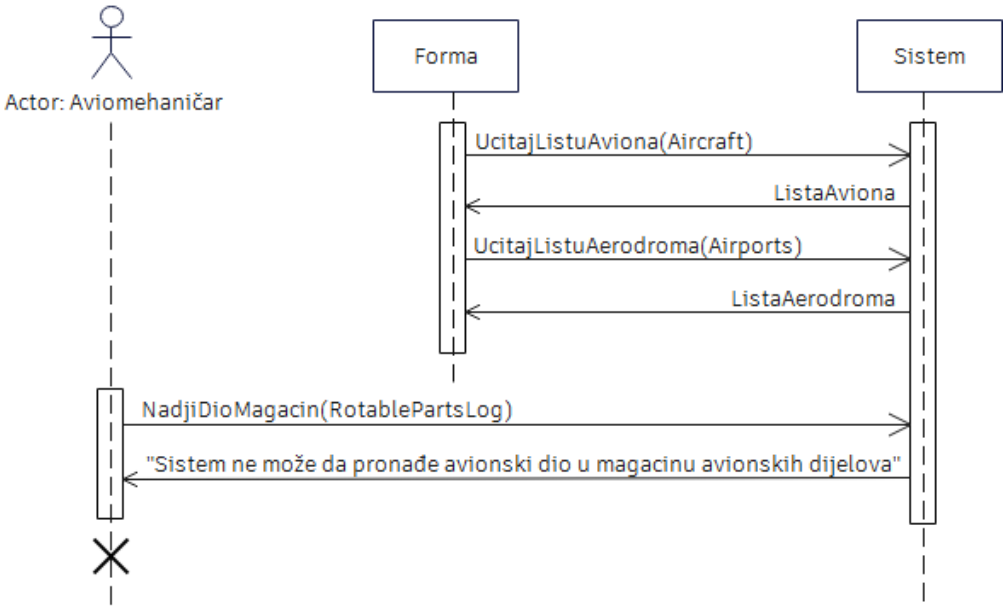
1. Forma poziva sistem da učita listu aviona. (APSO)
2. Sistem vraća formi listu aviona. (IA)
3. Forma poziva sistem da učita listu aerodroma. (APSO)
4. Sistem vraća formi listu aerodroma. (IA)
5. Aviomehaničar poziva sistem da na osnovu PN (Part number) i SN (Serial number) kao parametara nađe avionski dio u magacinu avionskih dijelova. (APSO)
6. Sistem vraća podatke iz kartona avionskog dijela kao i poruku „Sistem je našao avionski dio u magacinu avionskih dijelova“. (IA)
7. Aviomehaničar poziva sistem da na osnovu izabranog aviona vrati relevantne podatke bitne za instalaciju avionskog dijela (nalet aviona). (APSO)
8. Sistem vraća informacije o naletu aviona korisniku uz poruku „Sistem je spreman za instaliranje avio dijela na avion“. (IA)
9. Aviomehaničar poziva sistem da zapamti podatke vezane za instalaciju avionskog dijela. (APSO)
10. Sistem prikazuje podatke vezane za instalaciju avionskog dijela uz poruku „Sistem je instalirao avio dio na avionu“. (IA)



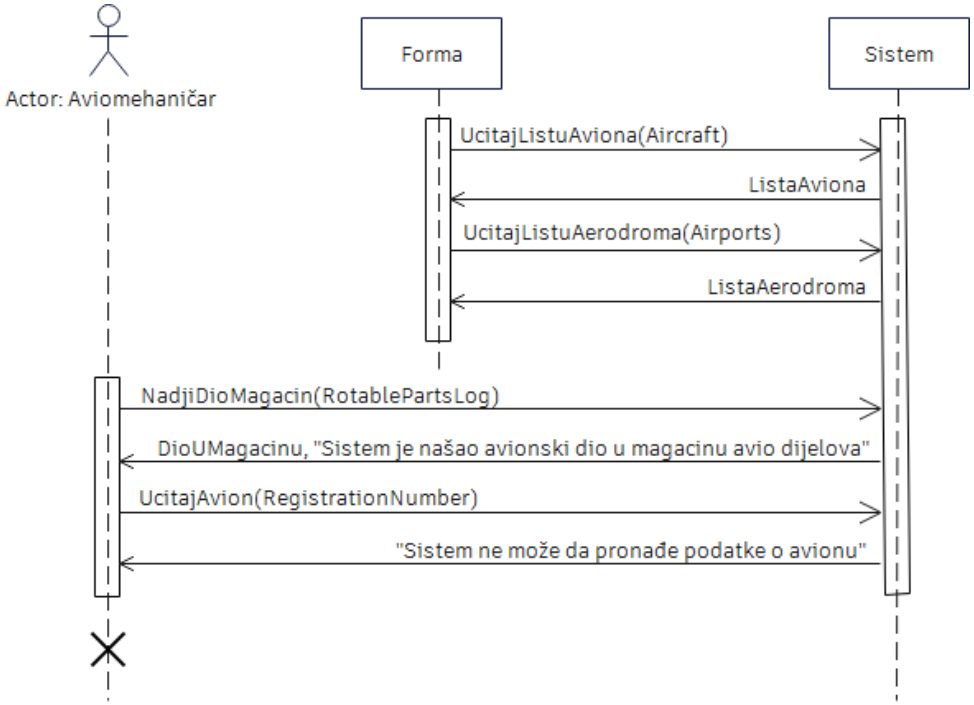
Alternativna scenarija

6.1 Ukoliko sistem ne može da nađe tražene podatke iz kartona avionskog dijela koji je lociran u magacinu on aviomehaničaru prikazuje poruku „Sistem ne može da

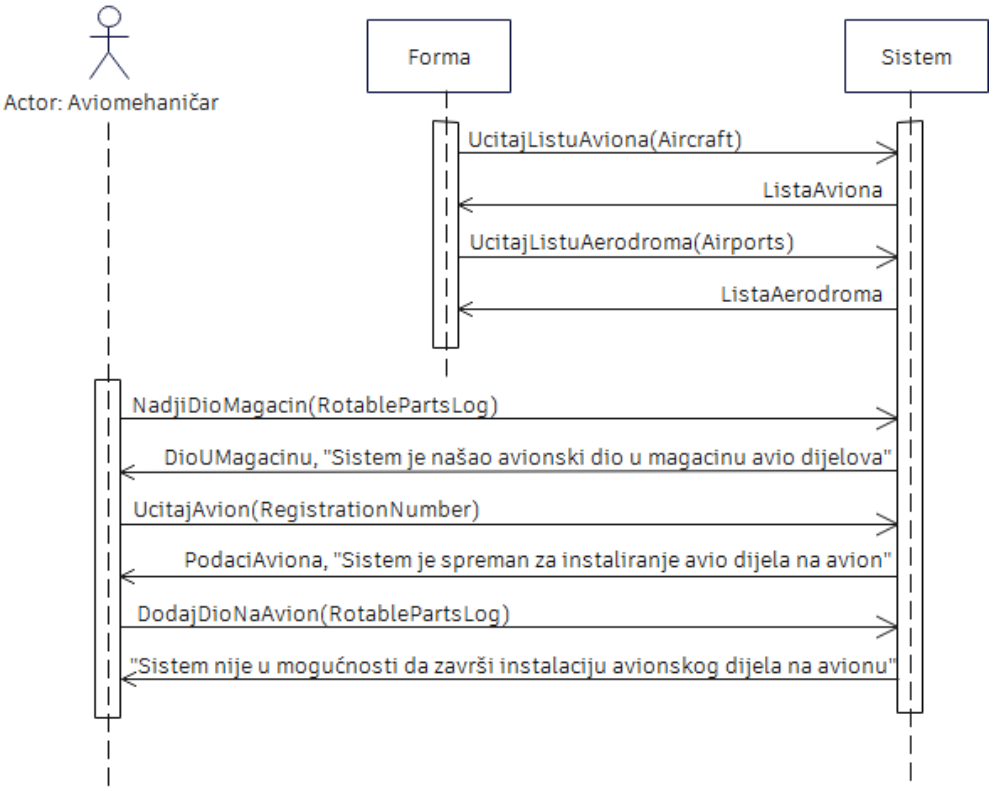
pronađe avionski dio u magacinu avionskih dijelova“. Prekida se izvršavanje scenarija. (IA)



8.1 Ukoliko sistem ne može da nađe tražene podatke o avionu on aviomehaničaru prikazuje poruku „Sistem ne može da pronađe podatke o avionu“. Prekida se izvršavanje scenarija. (IA)



10.1 U slučaju da sistem ne može da završi instalaciju avionskog dijela na avion on aviomehaničaru šalje poruku „Sistem nije u mogućnosti da završi instalaciju avionskog dijela na avionu“. (IA)



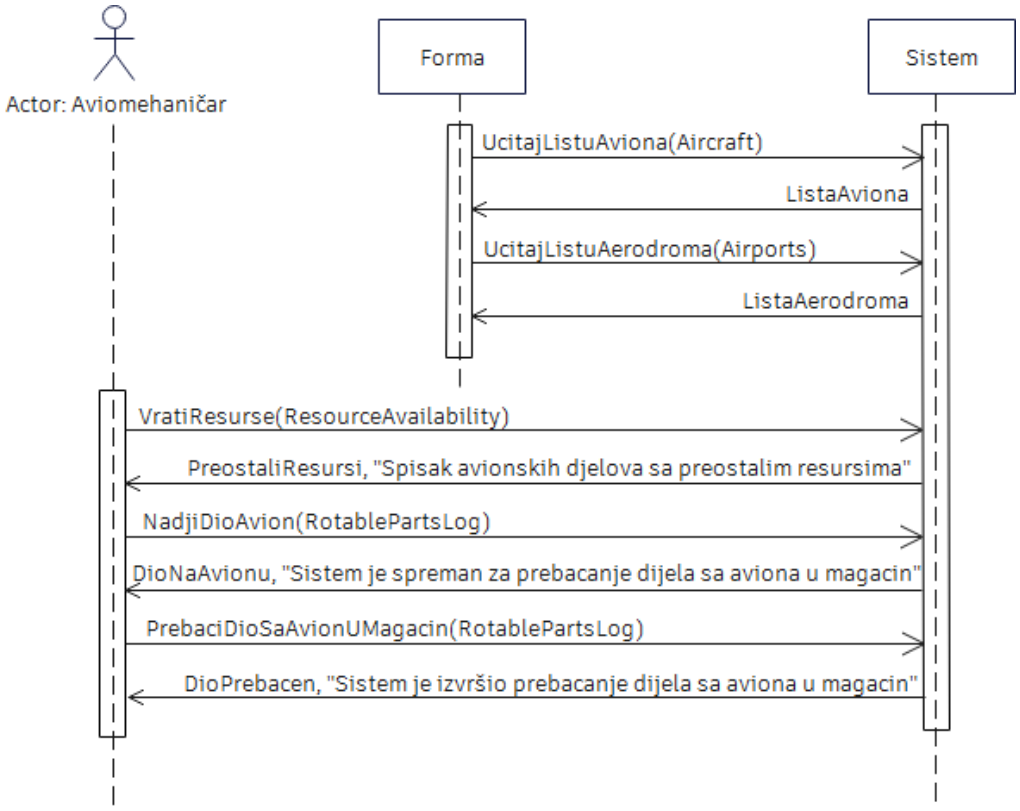
Sa navedenih sekvencijskih dijagrama uočavaju se sledeće sistemske operacije:

- Signal UcitajListuAviona(Aircraft)
- Signal UcitajListuAerodroma(Airports)
- Signal NadjiDioMagacin(RotablePartsLog)
- Signal UcitajAvion(RegistrationNumber)
- Signal DodajDioNaAvion(RotablePartsAircraft)

4.2.1.9 SK9: Slučaj korišćenja – Skidanje dijela sa aviona i slanje u magacin

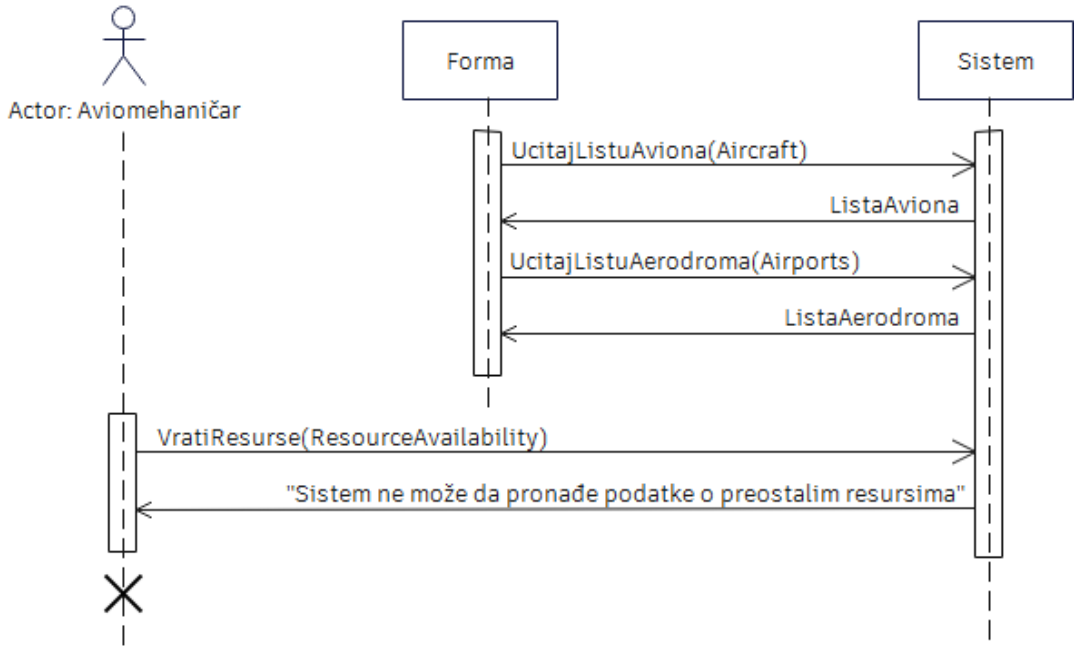
Osnovni scenario SK

1. Forma poziva sistem da učita listu aviona. (APSO)
2. Sistem vraća formi listu aviona. (IA)
3. Forma poziva sistem da učita listu aerodroma. (APSO)
4. Sistem vraća formi listu aerodroma. (IA)
5. Aviomehaničar poziva sistem da na osnovu izabranog aviona vrati sve dijelove instalirane na njega. (APSO)
6. Sistem vraća tražene podatke korisniku uz poruku „Spisak avionskih djelova sa preostalim resursima“. (IA)
7. Aviomehaničar poziva sistem da na osnovu PN (Part number) i SN (Serial number) vrati podatke o dijelu instaliranom na avionu. (APSO)
8. Sistem vraća podatke o dijelu instaliranom na avionu uz poruku „Sistem je spreman za prebacanje dijela sa aviona u magacin“. (IA)
9. Aviomehaničar poziva sistem da prebaci dio sa aviona u magacin avio dijelova. (APSO)
10. Sistem prikazuje podatke dijela u magacinu avionskih dijelova uz poruku „Sistem je izvršio prebacanje dijela sa aviona u magacin“. (IA)

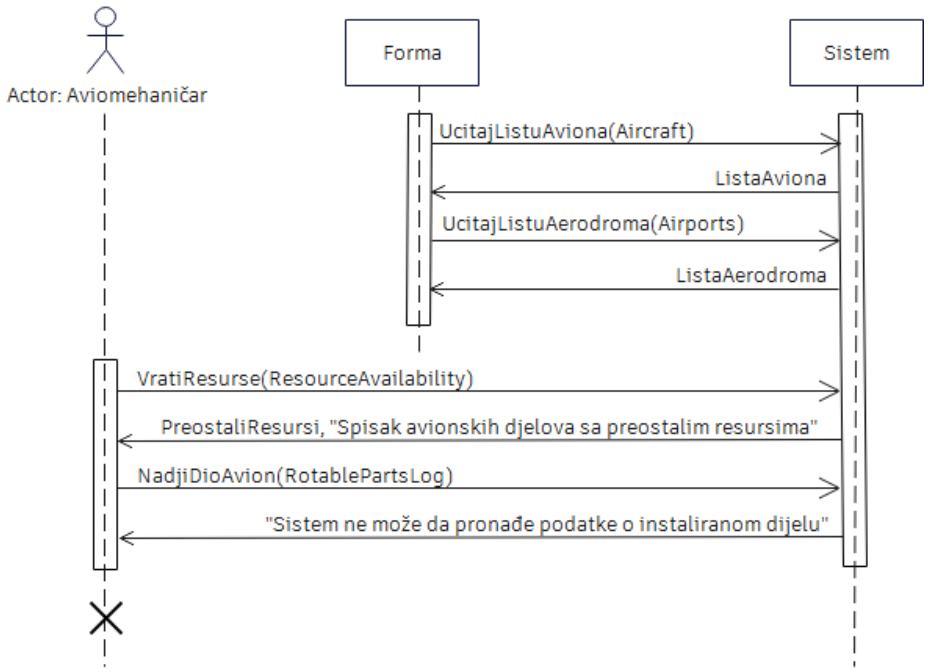


Alternativna scenarija

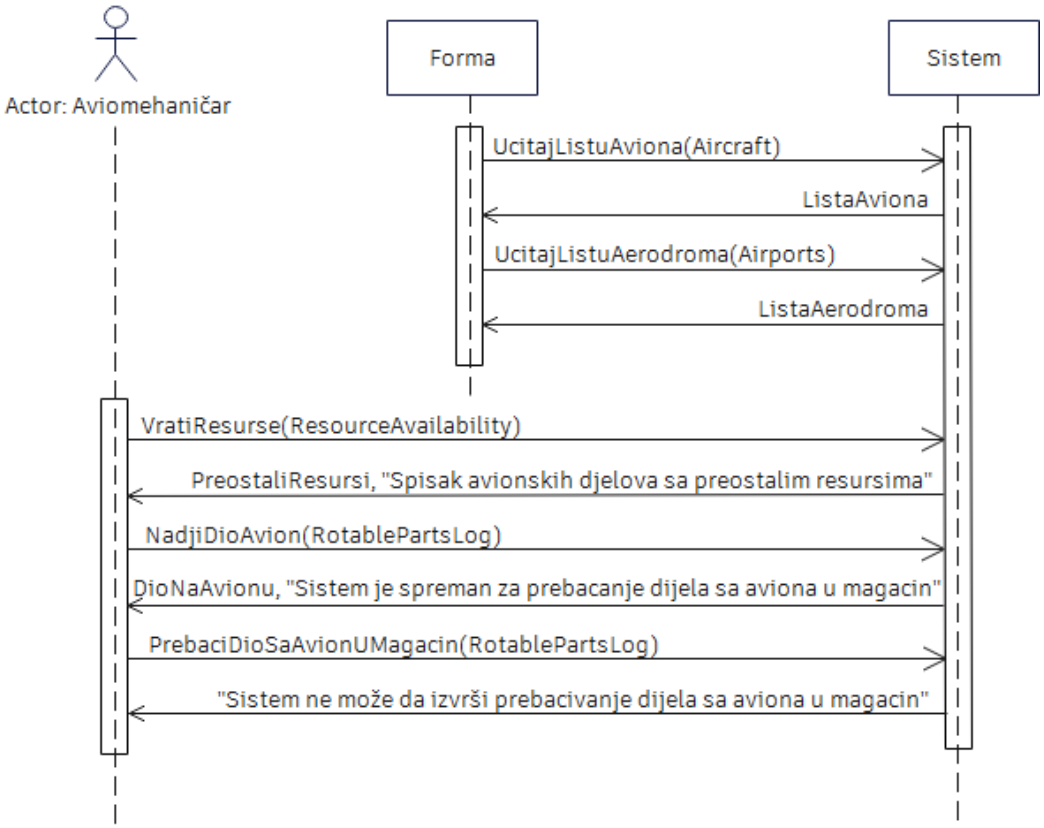
6.1 Ukoliko sistem ne može da nađe tražene podatke o instaliranim dijelovima na avionu on aviomehaničaru prikazuje poruku „Sistem ne može da pronađe podatke o preostalim resursima“. Prekida se izvršavanje scenarija. (IA)



8.1 Ukoliko sistem ne može da nađe tražene podatke o instaliranom dijelu na avionu on aviomehaničaru prikazuje poruku „Sistem ne može da pronađe podatke o instaliranom dijelu“. Prekida se izvršavanje scenarija. (IA)



10.1 Ukoliko sistem ne može da izvrši prebacivanje dijela sa aviona u magacin avionskih dijelova on aviomehaničaru prikazuje poruku „Sistem ne može da izvrši prebacivanje dijela sa aviona u magacin“.



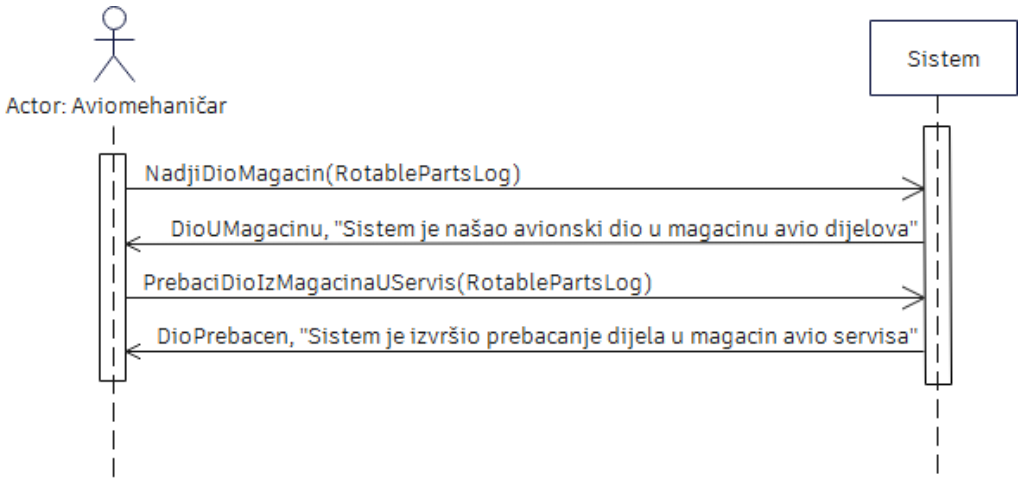
Sa navedenih sekvencijskih dijagrama uočavaju se sledeće sistemske operacije:

- Signal UcitajListuAviona(Aircraft)
- Signal UcitajListuAerodroma(Airports)
- Signal VratiResurse(ResourceAvailability)
- Signal NadjiDioAvion(RotablePartsLog)
- Signal PrebaciDioSaAvionaUMagacin(RotablePartsLog)

4.2.1.10 SK10: Slučaj korišćenja – Slanje dijela iz magacina u avio servis

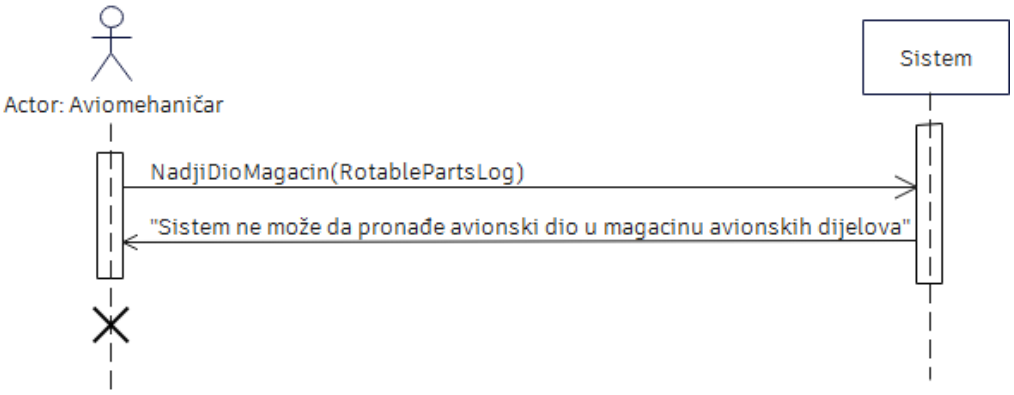
Osnovni scenario SK

1. Aviomehaničar poziva sistem da na osnovu PN (Part number) i SN (Serial number) kao parametara nađe avionski dio u magacinu avionskih dijelova. (APSO)
2. Sistem vraća podatke iz kartona avionskog dijela kao i poruku „Sistem je našao avionski dio u magacinu avionskih dijelova“. (IA)
3. Aviomehaničar poziva sistem da izabrani avionski dio prebaci iz magacina avionskih dijelova u magacin avio servisa. (APSO)
4. Sistem prikazuje podatke o prebačenom dijelu u avio servis uz poruku „Sistem je izvršio prebacanje dijela u magacin avio servisa“. (IA)

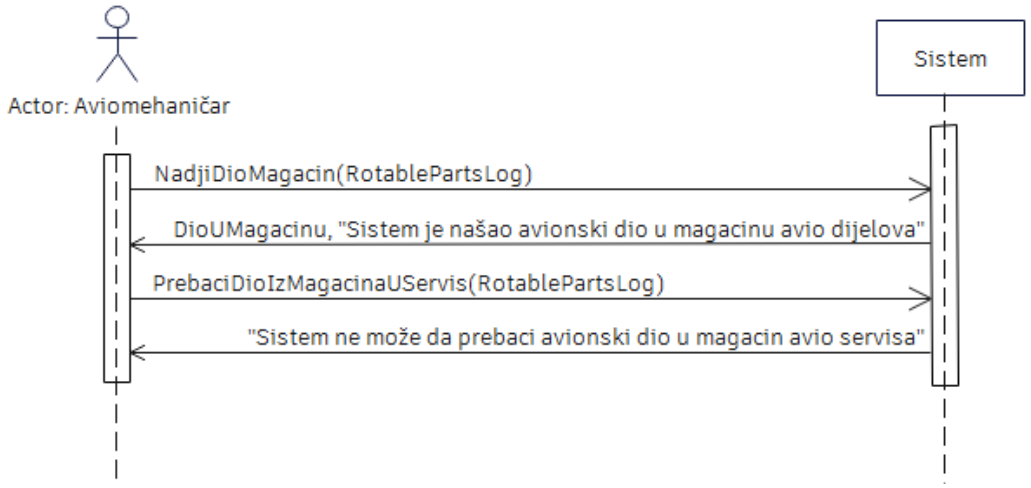


Alternativna scenarija

2.1 Ukoliko sistem ne može da nađe tražene podatke iz kartona avionskog dijela koji je lociran u magacinu on aviomehaničaru prikazuje poruku „Sistem ne može da pronađe avionski dio u magacinu avionskih dijelova“. Prekida se izvršavanje scenarija. (IA)



4.1 Ukoliko sistem ne može da izvršio prebacanje dijela iz magacina avionskih dijelova u magacin avio servisa on aviomehaničaru prikazuje poruku „Sistem ne može da prebaci avionski dio u magacin avio servisa“. (IA)



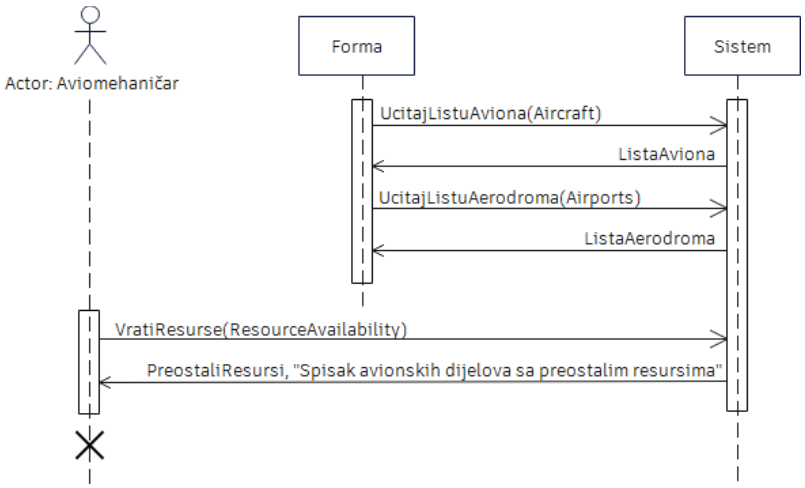
Sa navedenih sekvencijskih dijagrama uočavaju se sledeće sistemske operacije:

- Signal NadjiDioMagacin(RotablePartsLog)
- Signal PrebaciDioIzMagacinaUServis(RotablePartsLog)

4.2.1.11 SK11: Slučaj korišćenja – Izvještavanje o preostalim resursima dijela aviona

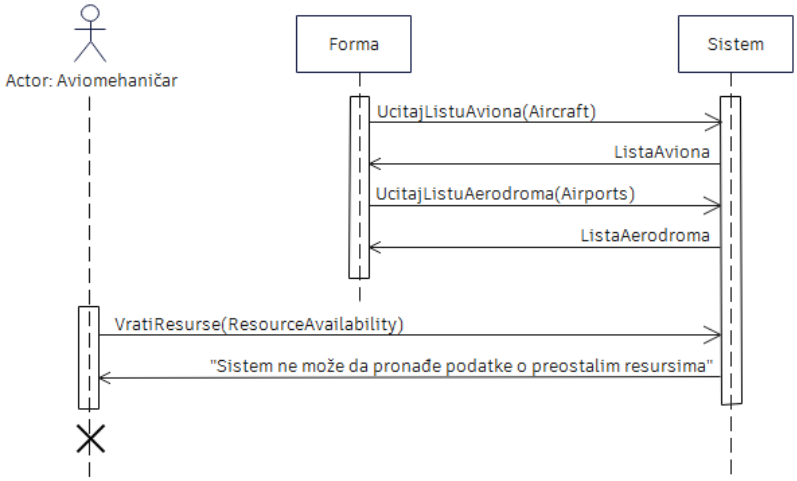
Osnovni scenario SK

1. Forma poziva sistem da učita listu aviona. (APSO)
2. Sistem vraća formi listu aviona. (IA)
3. Forma poziva sistem da učita listu aerodroma. (APSO)
4. Sistem vraća formi listu aerodroma. (IA)
5. Aviomehaničar poziva sistem da na osnovu izabranog aviona vrati spisak svih dijelova sa informacijom o preostaloj količini resursa. (APSO)
6. Sistem vraća tražene podatke korisnika uz poruku „Spisak avionskih dijelova sa preostalim resursima“. (IA)



Alternativna scenarija

6.1 Ukoliko sistem ne može da nađe tražene podatke o preostalim resursima on aviomehaničaru prikazuje poruku „Sistem ne može da pronađe podatke o preostalim resursima“.



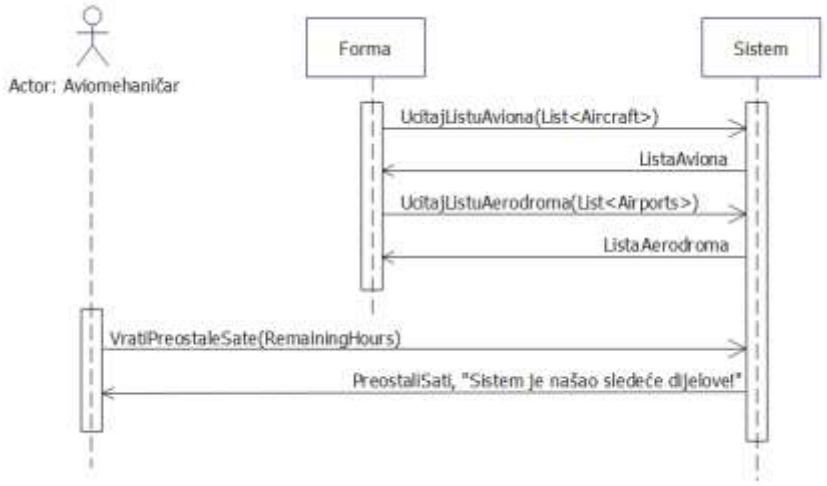
Sa navedenih sekvencijskih dijagrama uočavaju se sledeće sistemske operacije:

- Signal VratiResurse(ResourceAvailability)

4.2.1.12 SK12: Slučaj korišćenja – Izvještavanje o isteku resursa avionskog dijela (resurs u satima)

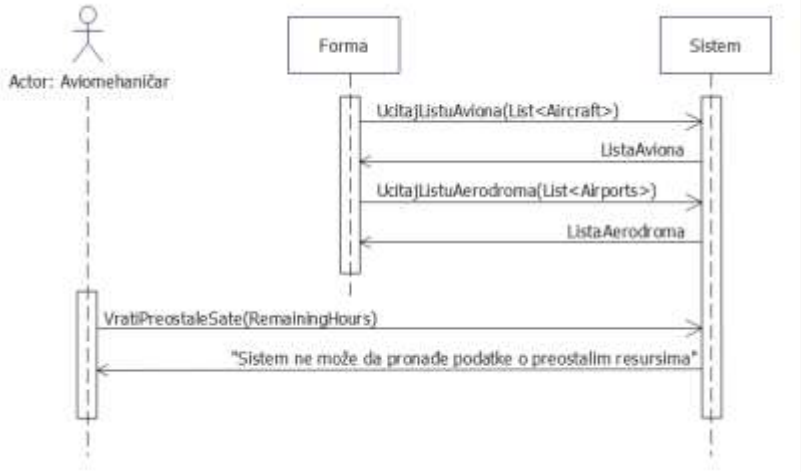
Osnovni scenario SK

1. Forma poziva sistem da učita listu aviona. (APSO)
2. Sistem vraća formi listu aviona. (IA)
3. Forma poziva sistem da učita listu aerodroma. (APSO)
4. Sistem vraća formi listu aerodroma. (IA)
5. Aviomehaničar poziva sistem da na osnovu izabranog aviona vrati spisak svih dijelova sa informacijom o isteku resusa avionskih dijelova u satima. (APSO)
6. Sistem vraća tražene podatke korisnika uz poruku „Sistem je našao sledeće dijelove!“. (IA)



Alternativna scenarija

6.1 Ukoliko sistem ne može da nađe tražene podatke on aviomehaničaru prikazuje poruku „Sistem ne može da pronađe podatke o preostalim resursima“.



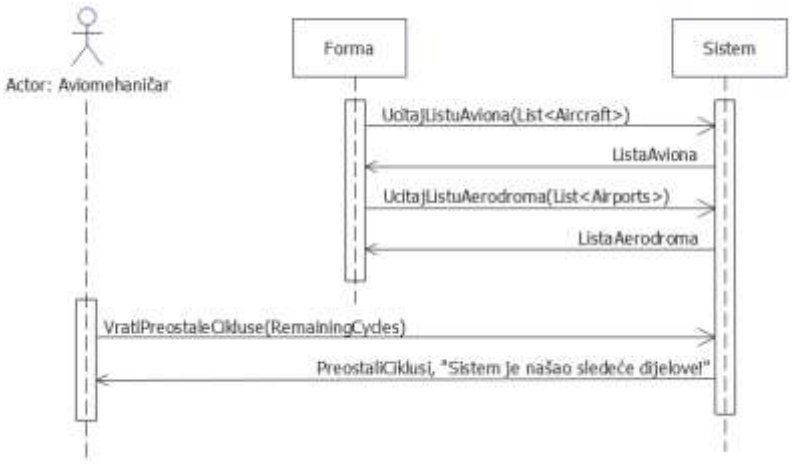
Sa navedenih sekvencijskih dijagrama uočavaju se sledeće sistemske operacije:

- Signal UcitajListuAviona(Aircraft)
- Signal UcitajListuAerodroma(Airports)
- Signal VratiPreostaleSate(RemainingHours)

4.2.1.13 SK13: Slučaj korišćenja – Izvještavanje o isteku resursa avionskog dijela (resurs u ciklusima)

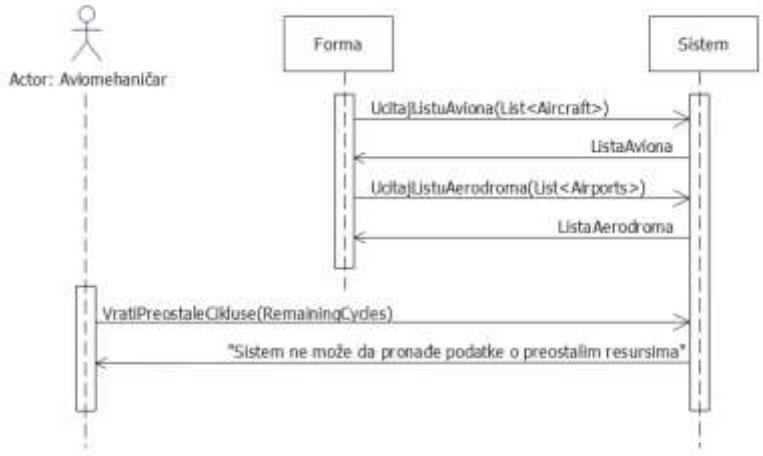
Osnovni scenario SK

1. Forma poziva sistem da učita listu aviona. (APSO)
2. Sistem vraća formi listu aviona. (IA)
3. Forma poziva sistem da učita listu aerodroma. (APSO)
4. Sistem vraća formi listu aerodroma. (IA)
5. Aviomehaničar poziva sistem da na osnovu izabranog aviona vrati spisak svih dijelova sa informacijom o isteku resusa avionskih dijelova u ciklusima. (APSO)
6. Sistem vraća tražene podatke korisnika uz poruku „Sistem je našao sledeće dijelove! “. (IA)



Alternativna scenarija

6.1 Ukoliko sistem ne može da nađe tražene podatke on aviomehaničaru prikazuje poruku „Sistem ne može da pronađe podatke o preostalim resursima“.



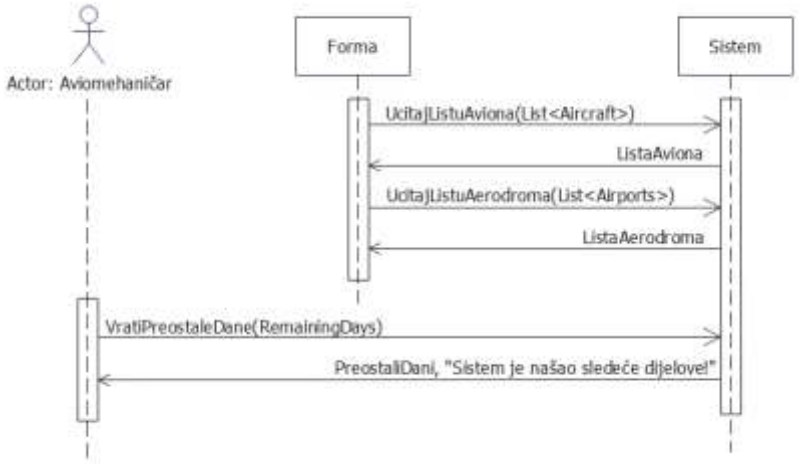
Sa navedenih sekvencijskih dijagrama uočavaju se sledeće sistemske operacije:

- Signal UcitajListuAviona(Aircraft)
- Signal UcitajListuAerodroma(Airports)
- Signal VratiPreostaleCikluse(RemainingCycles)

4.2.1.14 SK14: Slučaj korišćenja – Izvještavanje o isteku resursa avionskog dijela (resurs u danima)

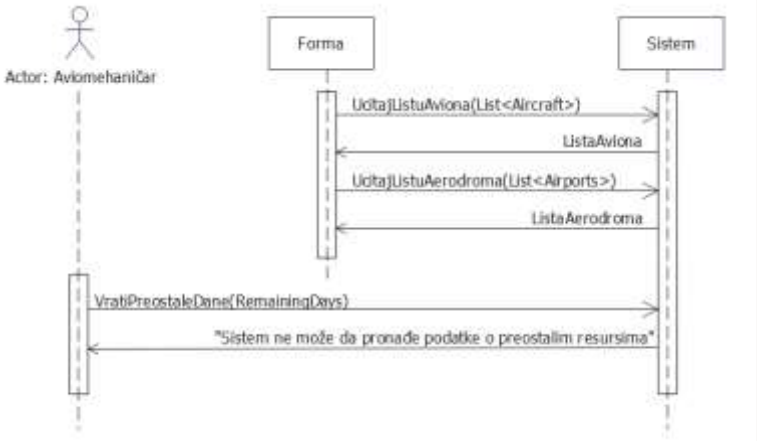
Osnovni scenario SK

1. Forma poziva sistem da učita listu aviona. (APSO)
2. Sistem vraća formi listu aviona. (IA)
3. Forma poziva sistem da učita listu aerodroma. (APSO)
4. Sistem vraća formi listu aerodroma. (IA)
5. Aviomehaničar poziva sistem da na osnovu izabranog aviona vrati spisak svih dijelova sa informacijom o isteku resusa avionskih dijelova u danima. (APSO)
6. Sistem vraća tražene podatke korisnika uz poruku „Sistem je našao sledeće dijelove!“. (IA)



Alternativna scenarija

6.1 Ukoliko sistem ne može da nađe tražene podatke on aviomehaničaru prikazuje poruku „Sistem ne može da pronađe podatke o preostalim resursima“.



Sa navedenih sekvencijskih dijagrama uočavaju se sledeće sistemske operacije:

- Signal UcitajListuAviona(Aircraft)
- Signal UcitajListuAerodroma(Airports)
- Signal VratiPreostaleDane(RemainingDays)

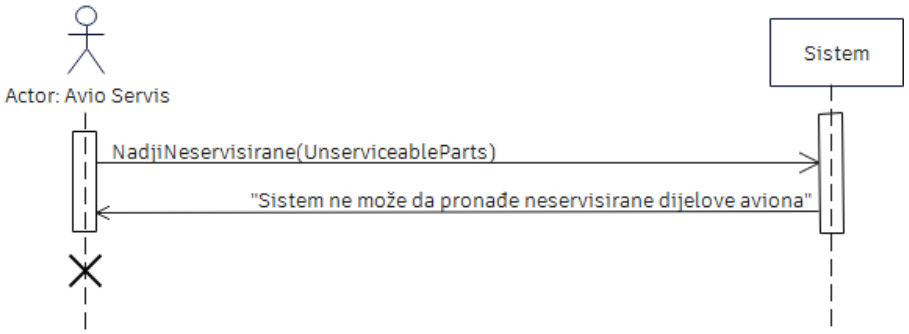
4.2.1.15 SK15: Slučaj korišćenja – Servisiranje i produženje resursa dijela aviona

Osnovni scenario SK

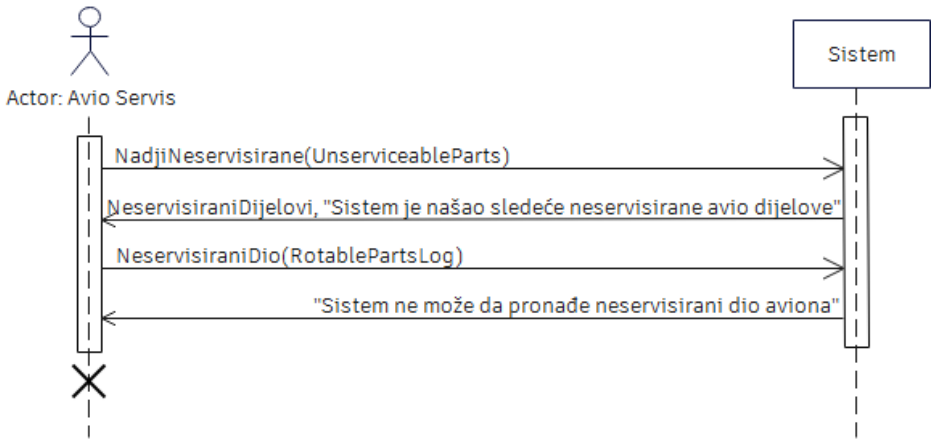
1. Avio servis poziva sistem da vrati spisak svih neservisiranih dijelova aviona. (APSO)
2. Sistem vraća spisak neservisiranih dijelova aviona uz poruku „Sistem je našao sledeće neservisirane avio dijelove“. (IA)
3. Avio servis poziva sistem da na osnovu PN (Part number) i SN (Serial number) kao parametara vrati informacije o selektovanom dijelu iz liste neservisiranih djelova. (APSO)
4. Sistem vraća avio servisu poruku „Sistem je spreman da sačuva rezultat inspekcije dijela“. (IA)
5. Avio servis poziva sistem da podatke vezane za inspekciju avionskog dijela sačuva i prebaci dio u magacin servisiranih dijelova. (APSO)
6. Sistem prikazuje podatke o servisiranom dijelu uz „Sistem je prebacio dio u magacin servisiranih avio dijelova“. (IA)

Alternativna scenarija

2.1 Ukoliko sistem ne može da nađe neservisirane dijelove aviona on avio servisu prikazuje poruku „Sistem ne može da pronađe neservisirane dijelove aviona“. Prekida se izvršavanje scenarija. (IA)



4.1 Ukoliko sistem ne može da nađe neservisirani dio aviona on avio servisu prikazuje poruku „Sistem ne može da pronađe neservisirani dio aviona“. Prekida se izvršavanje scenarija. (IA)



6.1 Ukoliko sistem ne može da zapamti podatke o servisiranju i prebacanju dijela u magacin servisiranih dijelova on avio servisu prikazuje poruku „Sistem ne može da prebaci dio u magacin servisiranih avionskih dijelova“.

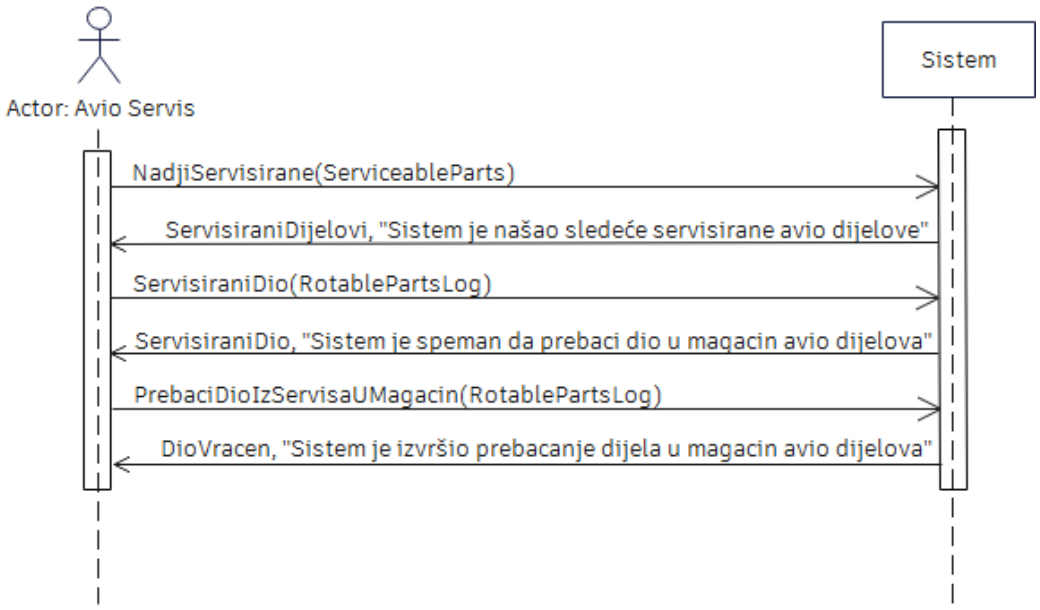
Sa navedenih sekvencijskih dijagrama uočavaju se sledeće sistemske operacije:

- Signal NadjiNeservisirane(UnserviceableParts)
- Signal NeservisiraniDio(RotablePartsLog)
- Signal RezultatInspekcije(RotablePartsService)

4.2.1.16 SK16: Slučaj korišćenja – Vraćanje dijela nakon popravke u magacin dijelova

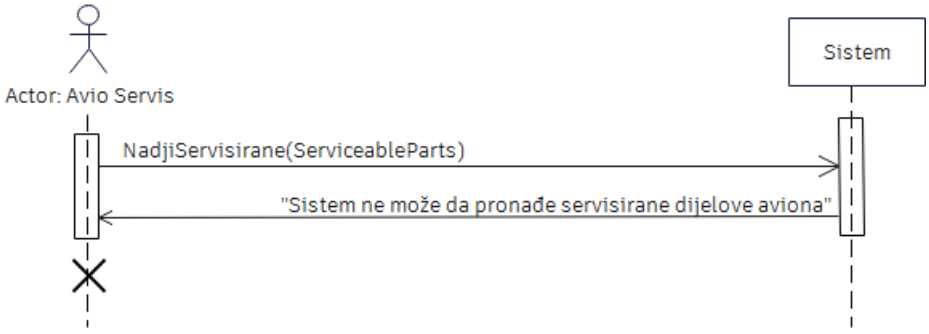
Osnovni scenario SK

1. Avio servis poziva sistem da iz magacin servisiranih dijelova vrati sve servisirane dijelove radi vraćanja avionskog dijela u magacin pošiljaoca. (APSO)
2. Sistem vraća spisak servisiranih dijelova aviona uz poruku „Sistem je našao sledeće servisirane avio dijelove“. (IA)
3. Avio servis poziva sistem da na osnovu PN (Part number) i SN (Serial number) kao parametara vrati informacije o selektovanom dijelu iz liste servisiranih djelova. (APSO)
4. Sistem vraća avio servisu poruku „Sistem je spreman da prebaci dio u magacin avio dijelova“. (IA)
5. Avio servis poziva sistem da prebaci dio u magacin avionskih dijelova. (APSO)
6. Sistem prikzuje podatke o dijelu koji se vraća u magacin avio dijelova uz poruku „Sistem je izvršio prebacanje dijela u magacin avio dijelova“. (IA)



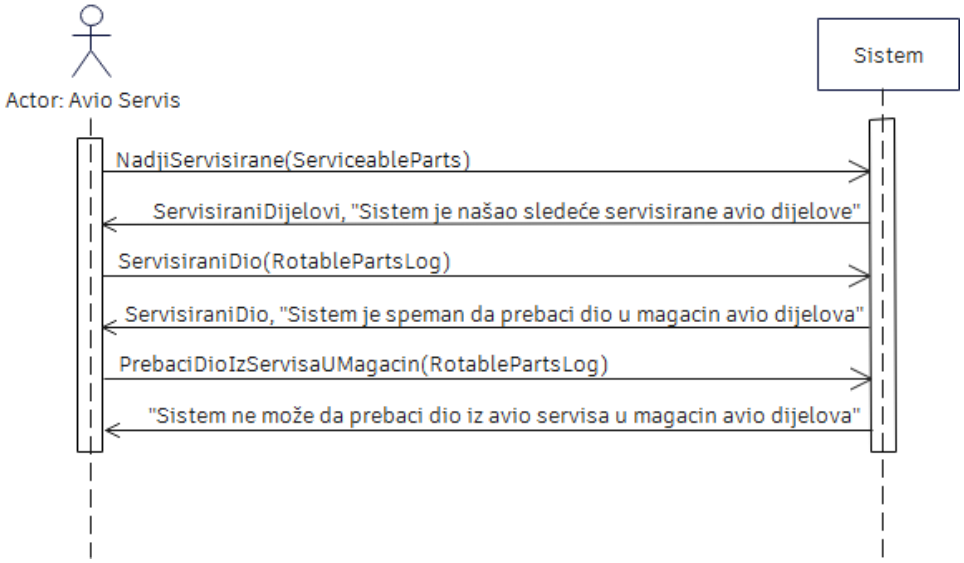
Alternativna scenarija

2.1 Ukoliko sistem ne može da nađe servisirane dijelove avionu on avio servisu prikazuje poruku „Sistem ne može da pronađe servisirane dijelove aviona“. Prekida se izvršavanje scenarija. (IA)



4.1 Ukoliko sistem ne može da nađe servisirani dio avionu on avio servisu prikazuje poruku „Sistem ne može da pronađe servisirani dio aviona“. Prekida se izvršavanje scenarija. (IA)

6.1 Ukoliko sistem ne može da izvrši prebacivanje dijela iz avio servisa u magacin avionskih dijelova on avio servisu prikazuje poruku „Sistem ne može da prebaci dio iz avio servisa u magacin avio dijelova“.



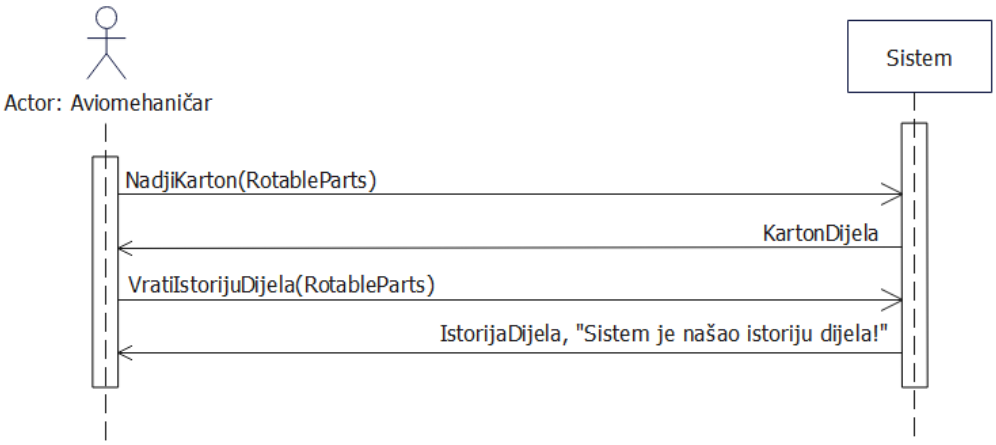
Sa navedenih sekvencijskih dijagrama uočavaju se sledeće sistemske operacije:

- Signal NadjiServisirane(ServiceableParts)
- Signal ServisiraniDio(RotablePartsLog)
- Signal PrebaciDioIzServisaUMagacin(RotablePartsLog)

4.2.1.17 SK17: Slučaj korišćenja – Prikaz kretanja dijela kroz sistem

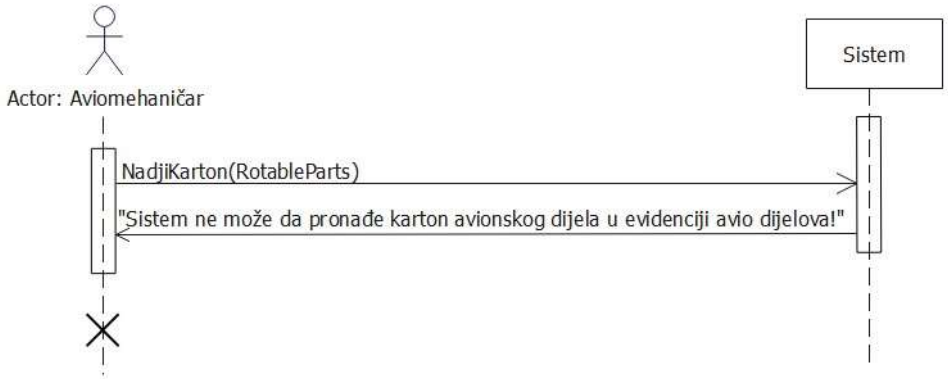
Osnovni scenario SK

1. Aviomehaničar poziva sistem da na osnovu PN (Part number) i SN (Serial number) kao parametara pronađe odgovarajući karton dijela. (APSO)
2. Sistem vraća odgovarajući karton dijela. (IA)
3. Aviomehaničar poziva sistem da na osnovu kartona dijela vrati informacije o istoriji kretanja dijela. (APSO)
4. Sistem vraća istoriju dijela uz poruku „Sistem je našao istoriju dijela!“. (IA)



Alternativna scenarija

2.1 Ukoliko sistem ne može da odgovarajući karton dijela on aviomehaničaru prikazuje poruku „Sistem ne može da pronađe karton avionskog dijela u evidenciji avio dijelova!“. Prekida se izvršavanje scenarija. (IA)



4.1 Ukoliko sistem ne može da nađe istoriju dijela on aviomehaničaru prikazuje poruku „Sistem ne može da pronađe podatke o istoriji dijela“. (IA)

Sa navedenih sekvencijskih dijagrama uočavaju se sledeće sistemske operacije:

- Signal NadjiKarton(RotableParts)
- Signal VratiIstorijuDijela(RotablePartsHistory)

Na osnovu analize scenarija dobijeno je 29 sistemskih operacija:

1. Signal ZapamtiRegistraciju(User)
2. Signal VerifikacijaKorisnika(User)
3. Signal UcitajListuAviona(Aircraft)
4. Signal UcitajAvion(Aircraft)
5. Signal UcitajListuAerodroma(Airports)
6. Signal PronadjiLet(LogBook)
7. Signal UcitajListuLetova(LogBook)
8. Signal ZapamtiLet(LogBook)
9. Signal AzurirajLet(LogBook)
10. Signal NadjiKarton(RotableParts)
11. Signal ZapamtiKarton(RotablePartsLog)
12. Signal AzurirajKarton(RotablePartsLog)
13. Signal NadjiDioAvion(RotablePartsLog)
14. Signal NadjiDioMagacin(RotablePartsLog)
15. Signal DodajDioNaAvion(RotablePartsLog)
16. Signal PrebaciDioSaAvionUMagacin(RotablePartsLog)
17. Signal PrebaciDioIzMagacinaUServis(RotablePartsLog)
18. Signal PrebaciDioIzServisaUMagacin(RotablePartsLog)
19. Signal NadjiNeservisirane(UnserviseableParts)
20. Signal NadjiServisirane(ServiseableParts)
21. Signal NeservisiraniDio(RotablePartsLog)
22. Signal ServisiraniDio(RotablePartsLog)
23. Signal RezultatInspekcije(RotablePartsService)
24. Signal VratiResurse(ResourceAvailability)
25. Signal DodajAvion(Aircraft)
26. Signal VratiPreostaleSate(RemainingHours)
27. Signal VratiPreostaleCikluse(RemainingCycles)
28. Signal VratiPreostaleDane(RemainingDays)
29. Signal VratiIstorijuDijela(RotablePartsHistory)

4.2.2 Ponašanje softverskog sistema – Definisanje ugovora o sistemskim operacijama

Ponašanje softverskog sistema se opisuje preko sistemskih operacija, a za svaku sistemsku operaciju se pravi ugovor. Ugovor opisuju ponašanje sistemske operacije, to jest opisuje se ono šta ta sistemska opeacija treba da odradi (ali ne i kako to treba da odradi).

Jedan ugovor vezuje se za jednu sistemsku operaciju, i sastoji se od sledećih sekcija:

- operacija
- veza sa SK
- preduslov
- postuslov

Ugovor UG1: ZapamtiRegistraciju(User) Signal;
Veza sa SK: SK1
Preduslovi: Vrijednosna i strukturna ograničenja nad objektom *User* moraju biti zadovoljena.
Postuslovi: Podaci o korisniku su zapamćeni.

Ugovor UG2: VerifikacijaKorisnika(User) Signal;
Veza sa SK: SK2
Preduslovi:
Postuslovi:

Ugovor UG3: UcitajListuAviona(Aircraft) Signal;
Veza sa SK: SK3, SK4, SK5, SK8, SK9, SK12, SK13, SK14
Preduslovi:
Postuslovi:

Ugovor UG4: UcitajAvion(Aircraft) Signal;
Veza sa SK: SK4, SK8
Preduslovi:
Postuslovi:

Ugovor UG5: UcitajListuAerodroma(Airports) Signal;
Veza sa SK: SK3, SK4, SK5, SK8, SK9, SK12, SK13, SK14
Preduslovi:
Postuslovi:

Ugovor UG6: PronadjiLet(LogBook) Signal;
Veza sa SK: SK5
Preduslovi:
Postuslovi:

Ugovor UG7: UcitajListuLetova(LogBook) Signal;
Veza sa SK: SK5
Preduslovi:
Postuslovi:

Ugovor UG8: ZapamtiLet(LogBook) Signal;
Veza sa SK: SK4
Preduslovi: Vrijednosna i strukturna ograničenja nad objektom *LogBook* moraju biti zadovoljena
Postuslovi: Podaci o letu su zapamćeni

Ugovor UG9: AzurirajLet(LogBook) Signal;
Veza sa SK: SK5
Preduslovi: Vrijednosna i strukturna ograničenja nad objektom *LogBook* moraju biti zadovoljena.
Postuslovi: Podaci o ažuriranom letu su zapamćeni.

Ugovor UG10: NadjiKarton(RotableParts) Signal;
Veza sa SK: SK6, SK17
Preduslovi:
Postuslovi:

Ugovor UG11: ZapamtiKarton(RotablePartsLog) Signal;
Veza sa SK: SK6
Preduslovi: Vrijednosna i strukturna ograničenja nad objektima *RotableParts, RotablePartsLog, RotablePartsStock* moraju biti zadovoljena.
Postuslovi: Podaci iz kartona avio dijela su zapamćeni.

Ugovor UG12: AzurirajKarton(RotablePartsLog) Signal;
Veza sa SK: SK7
Preduslovi: Vrijednosna i strukturna ograničenja nad objektima *RotableParts, RotablePartsStock* moraju biti zadovoljena.
Postuslovi: Ažurirani podaci iz kartona avio dijela su zapamćeni.

Ugovor UG13: NadjiDioAvion(RotablePartsLog) Signal;
Veza sa SK: SK9
Preduslovi:
Postuslovi:

Ugovor UG14: NadjiDioMagacin(RotablePartsLog) Signal;
Veza sa SK: SK7, SK8, SK10
Preduslovi:
Postuslovi:

Ugovor UG15: DodajDioNaAvion(RotablePartsLog) Signal;
Veza sa SK: SK8
Preduslovi: Vrijednosna i strukturna ograničenja nad objektima *RotablePartsAircraft, RotablePartsLog* moraju biti zadovoljena.
Postuslovi: Podaci o instalaciji dijela na avion su zapamćeni.

Ugovor UG16: PrebaciDioSaAvionaUMagacin(RotablePartsLog) Signal;
Veza sa SK: SK9
Preduslovi: Vrijednosna i strukturna ograničenja nad objektima *RotablePartsStock, RotablePartsLog* moraju biti zadovoljena.
Postuslovi: Podaci o prebacanju dijela sa aviona u magacin su zapamćeni.

Ugovor UG17: PrebaciDioIzMagacinaUServis(RotablePartsLog) Signal;
Veza sa SK: SK10
Preduslovi: Vrijednosna i strukturna ograničenja nad objektima *RotablePartsService, RotablePartsLog* moraju biti zadovoljena.
Postuslovi: Podaci o prebacanju dijela iz magacina u servis su zapamćeni.

Ugovor UG18: PrebaciDioIzServisaUMagacin(RotablePartsLog) Signal;
Veza sa SK: SK16
Preduslovi: Vrijednosna i strukturna ograničenja nad objektima *RotablePartsStock, RotablePartsLog* moraju biti zadovoljena.
Postuslovi: Podaci o prebacanju dijela iz servisa u magacin avio dijelova su zapamćeni.

Ugovor UG19: NadjiNeservisirane(UnserviseableParts) Signal;
Veza sa SK: SK15
Preduslovi:
Postuslovi:

Ugovor UG20: NadjiServisirane(ServiseableParts) Signal;
Veza sa SK: SK16
Preduslovi:
Postuslovi:

Ugovor UG21: NeservisiraniDio(RotablePartsLog) Signal;
Veza sa SK: SK15
Preduslovi:
Postuslovi:

Ugovor UG22: ServisiraniDio(RotablePartsLog) Signal;
Veza sa SK: SK16
Preduslovi:
Postuslovi:

Ugovor UG23: RezultatInspekcije(RotablePartsService) Signal;
Veza sa SK: SK15
Preduslovi:
Postuslovi:

Ugovor UG24: VratiResurse(ResourceAvailability) Signal;
Veza sa SK: SK9, SK11
Preduslovi:
Postuslovi:

Ugovor UG25: DodajAvion(Aircraft) Signal;
Veza sa SK: SK3
Preduslovi: Vrijednosna i strukturna ograničenja nad objektom *Aircraf* moraju biti zadovoljena.
Postuslovi: Podaci o avionu su zapamćeni.

Ugovor UG26: VratiPreostaleSate(RemainingHours) Signal;
Veza sa SK: SK12
Preduslovi:

Postuslovi:

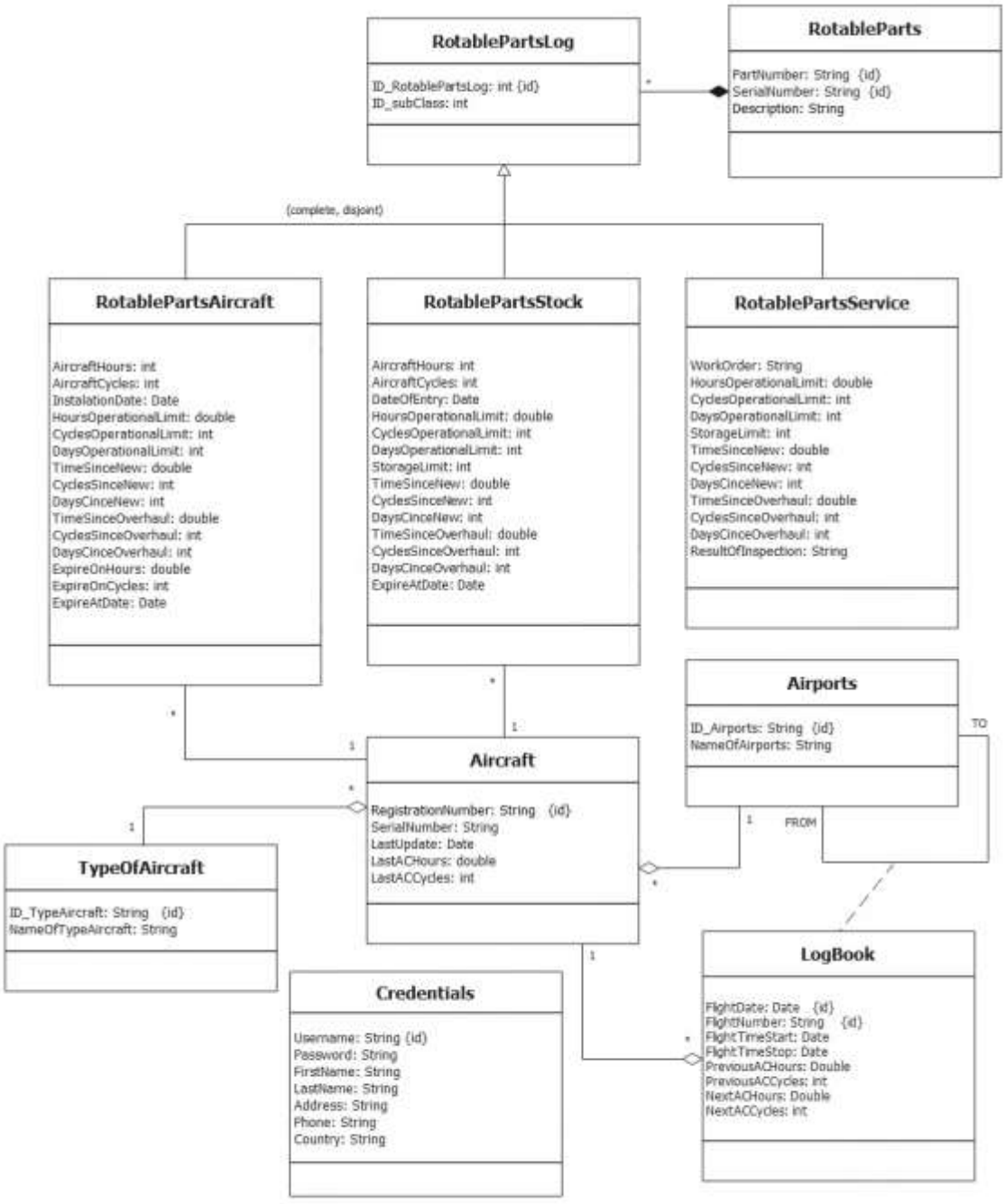
Ugovor UG27: VratiPreostaleCikluse(RemainingCycles) Signal;
Veza sa SK: SK13
Preduslovi:
Postuslovi:

Ugovor UG28: VratiPreostaleDane(RemainingDays) Signal;
Veza sa SK: SK14
Preduslovi:
Postuslovi:

Ugovor UG29: VratiIstorijuDijela(RotablePartsHistory) Signal;
Veza sa SK: SK17
Preduslovi:
Postuslovi:

4.2.3 Struktura softverskog sistema – Konceptualni (domenski) dijagram

Pomoću konceptualnog modela opisujemo strukturu sistema. Konceptualni model sadrži konceptualne klase (domenske objekte) i asocijacije između konceptualnih klasa.



Slika 7: Konceptualni model

4.2.4 Struktura softverskog sistema – Relacioni model

Na osnovu konceptualnog modela, pravi se relacioni model, a na osnovu njega se projektuje relaciona baza podataka.

U konceptualnom modelu se mogu indentifikovati sledeće klase: Credentials, RotableParts, RotableLog, RotablePartsAircraft, RotablePartsStock, RotablePartsService, Aircraft, Airports, LogBook. Svaka relacija će biti predstavljena kao jedna tabela u relacionom modelu.

Credentials (Username, Password, FirstName, LastName, Address, Phone, Country)

RotableParts (ID_RotableParts, PartNumber, SerialNumber, Description)

RotablePartsLog (ID_RotablePartsLog, ID_RotableParts, ID_subClass)

RotablePartsAircraft (ID_RotablePartsLog, ID_RotableParts, *RegistrationNumber*, AircraftHours, AircraftCycles, InstalationDate, HoursOperationalLimit, CyclesOperationalLimit, DaysOperationalLimit, StorageLimit, TimeSinceNew, CyclesSinceNew, DaysSinceNew, TimeSinceOverhaul, CyclesSinceOverhaul, DaysSinceOverhaul, ExpireOnHours, ExpireOnCycles, ExpireAtDate)

RotablePartsStock (ID_RotablePartsLog, ID_RotableParts, *RegistrationNumber*, AircraftHours, AircraftCycles, DateOfEntry, HoursOperationalLimit, CyclesOperationalLimit, DaysOperationalLimit, StorageLimit, TimeSinceNew, CyclesSinceNew, DaysSinceNew, TimeSinceOverhaul, CyclesSinceOverhaul, DaysSinceOverhaul, ExpireAtDate, IsInitial)

RotablePartsService (ID_RotablePartsLog, ID_RotableParts, WorkOrder, WorkOrderDescription, HoursOperationalLimit, CyclesOperationalLimit, DaysOperationalLimit, StorageLimit, TimeSinceNew, CyclesSinceNew, DaysSinceNew, TimeSinceOverhaul, CyclesSinceOverhaul, DaysSinceOverhaul, ResultOfInspection, NewHoursOperationalLimit, NewCyclesOperationalLimit, NewDaysOperationalLimit, NewStorageLimit, Description)

Aircraft (RegistrationNumber, SerialNumber, LastUpdate, *ID_Airports*, LastACHours, LastACCycles)

Airports (ID_Airports, NameOfAirports)

LogBook (ID_LogBook, ID_Airports, ID_Airports-2, FlightDate, FlightNumber, R*egistrationNumber* FlightTimeStart, FlightTimeStop, PreviousACHours, PreviousACCycles, NextACHours, NextACCycles)

Tabele ograničenja

Tabela Credentials	**Prosto vremensko ograničenje**		**Složeno vremensko ograničenje**		**Strukturno ograničenje**
Naziv atributa	**Tip atributa**	**Vrijednost atributa**	**Međuzavis-nost atributa jedne tabele**	**Međuzavis-nost atributa vise tabela**	
Username	String	Not NULL			INSERT / UPDATE / DELETE /
Password	String				
FirstName	String				
LastName	String				
Address	String				
Phone	String				
Country	String				

Tabela RotableParts	**Prosto vremensko ograničenje**		**Složeno vremensko ograničenje**		**Strukturno ograničenje**
Naziv atributa	**Tip atributa**	**Vrijednost atributa**	**Međuzavis-nost atributa jedne tabele**	**Međuzavis-nost atributa vise tabela**	INSERT / UPDATE Cascade RotablePartsLog DELETE Restricted RotablePartsLog
ID_RotableParts	Integer	Not NULL			
PartNumber	String	Not NULL			
SerialNumber	String	Not NULL			
Description	String	Not NULL			

Tabela RotablePartsLog	**Prosto vremensko ograničenje**		**Složeno vremensko ograničenje**		**Strukturno ograničenje**
Naziv atributa	**Tip atributa**	**Vrijednost atributa**	**Međuzavis-nost atributa jedne tabele**	**Međuzavis-nost atributa vise tabela**	INSERT Restricted PotableParts UPDATE Restricted PotableParts UPDATE Cascade RotablePartsAircraft RotablePartsStock RotablePartsService DELETE Cascade RotablePartsAircraft RotablePartsStock RotablePartsService
ID_RotablePartsLog	Integer	Not NULL			
ID_RotableParts	Integer	Not NULL			
ID_SubClass	Integer	Not NULL			

Tabela RotablePartsAircraft	**Prosto vremensko ograničenje**		**Složeno vremensko ograničenje**		**Strukturno ograničenje**
Naziv atributa	**Tip atributa**	**Vrijednost atributa**	**Međuzavis-nost atributa jedne tabele**	**Međuzavis-nost atributa vise tabela**	
ID_RotablePartsLog	Integer	Not NULL			
ID_RotableParts	Integer	Not NULL			
RegistrationNumber	String	Not NULL			
AircraftHours	Double	Not NULL			
AircraftCycles	Integer	Not NULL			
InstalationDate	DateTime	Not NULL			INSERT Restricted PotablePartsLog Aircraft UPDATE Restricted PotablePartsLog Aircraft DELETE /
HoursOperationalLimit	Double				
CyclesOperationalLimit	Integer				
DaysOperationalLimit	Integer				
StorageLimit	Integer				
TimeSinceNew	Double				
CyclesSinceNew	Integer				
DaysSinceNew	Integer				
TimeSinceOverhaul	Double				
CyclesSinceOverhaul	Integer				
DaysSinceOverhaul	Integer				
ExpireOnHours	Double		*		
ExpireOnCycles	Integer		**		
ExireAtDate	DateTime		***		

* AircraftHours + HoursOperationalLimit – TimeSinceOverhaul
** AircraftCycless + CyclesOperationalLimit - CyclesSinceOverhaul
*** InstalationDate + DaysOperationalLimit – DaysSinceOverhaul

Tabela RotablePartsService	**Prosto vremensko ograničenje**		**Složeno vremensko ograničenje**		**Strukturno ograničenje**
Naziv atributa	**Tip atributa**	**Vrijednost atributa**	**Međuzavis-nost atributa jedne tabele**	**Međuzavis-nost atributa vise tabela**	
ID_RotablePartsLog	Integer	Not NULL			INSERT Restricted PotablePartsLog UPDATE Restricted PotablePartsLog DELETE /
ID_RotableParts	Integer	Not NULL			
WorkOrder	String	Not NULL			
WorkOrderDescription	String	Not NULL			
HoursOperationalLimit	Double				
CyclesOperationalLimit	Integer				
DaysOperationalLimit	Integer				
StorageLimit	Integer				
TimeSinceNew	Double				
CyclesSinceNew	Integer				
DaysSinceNew	Integer				
TimeSinceOverhaul	Double				
CyclesSinceOverhaul	Integer				
DaysSinceOverhaul	Integer				
ResultOfInspection	Integer				
NewHoursOperationalLimit	Double				
NewCyclesOperationalLimit	Integer				
NewDaysOperationalLimit	Integer				
NewStorageLimit	Integer				
Description	String				

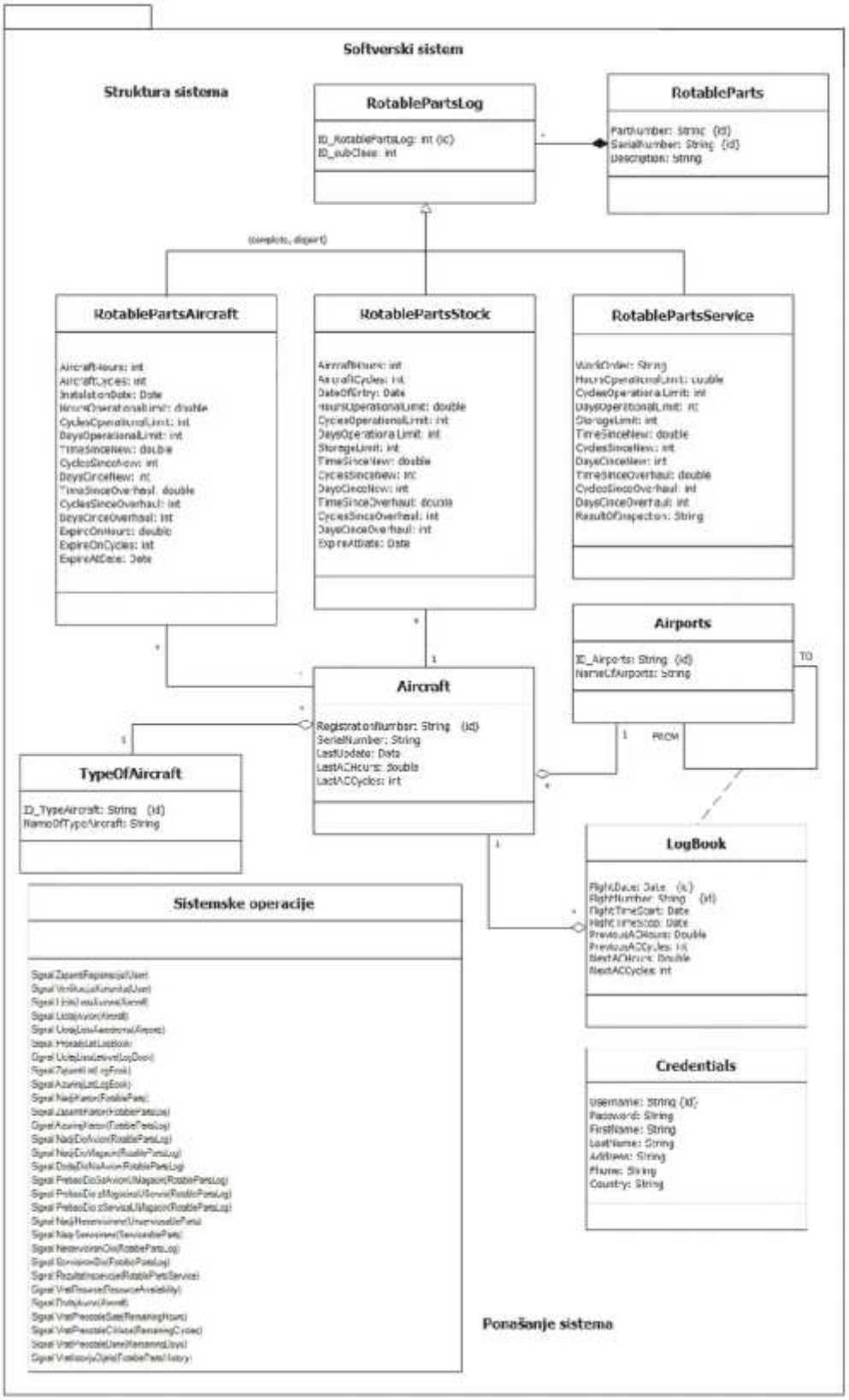
Tabela Airports	**Prosto vremensko ograničenje**		**Složeno vremensko ograničenje**		**Strukturno ograničenje**
Naziv atributa	**Tip atributa**	**Vrijednost atributa**	**Međuzavis-nost atributa jedne tabele**	**Međuzavis-nost atributa vise tabela**	
ID_Airports	Integer	Not NULL			INSERT / UPDATE Cascade Aircraft LogBook DELETE Restricted Airraft LogBook
NameOfAirports	String	Not NULL			

Tabela RotablePartsStock	**Prosto vremensko ograničenje**		**Složeno vremensko ograničenje**		**Strukturno ograničenje**
Naziv atributa	**Tip atributa**	**Vrijednost atributa**	**Međuzavis-nost atributa jedne tabele**	**Međuzavis-nost atributa vise tabela**	
ID_RotablePartsLog	Integer	Not NULL			INSERT Restricted PotablePartsLog Aircraft UPDATE Restricted PotablePartsLog Aircraft DELETE /
ID_RotableParts	Integer	Not NULL			
RegistrationNumber	String				
AircraftHours	Double				
AircraftCycles	Integer				
DateOfEntry	DateTime	Not NULL			
HoursOperationalLimit	Double				
CyclesOperationalLimit	Integer				
DaysOperationalLimit	Integer				
StorageLimit	Integer				
TimeSinceNew	Double				
CyclesSinceNew	Integer				
DaysSinceNew	Integer				
TimeSinceOverhaul	Double				
CyclesSinceOverhaul	Integer				
DaysSinceOverhaul	Integer				
ExireAtDate	DateTime		*		
IsInitial	Byte				
* DateOfEntry + StorageLimit					

Tabela LogBook	**Prosto vremensko ograničenje**		**Složeno vremensko ograničenje**		**Strukturno ograničenje**
Naziv atributa	**Tip atributa**	**Vrijednost atributa**	**Međuzavis-nost atributa jedne tabele**	**Međuzavis-nost atributa vise tabela**	
ID_LogBook	Integer	Not NULL			INSERT Restricted Airport Aircraft UPDATE Restricted Airport Aircraft DELETE /
ID_Airports (FROM)	Integer	Not NULL			
ID_Airports-2 (TO)	Integer	Not NULL			
FlightDate	DateTime	Not NULL			
FlightNumber	String	Not NULL			
RegistrationNumber	String	Not NULL			
FlightTimeStart	DateTime	Not NULL			
FlightTimeStop	DateTime	Not NULL			
PreviousACHours	Double	Not NULL			
PreviousACCycles	Integer	Not NULL			
NextACHours	Double	Not NULL	*		
NextACCycles	Integer	Not NULL	**		
* PreviousACHours + FlightTimeStop – FlightTimeStart ** PreviousACCycles + 1					

Tabela Aircraft	**Prosto vremensko ograničenje**		**Složeno vremensko ograničenje**		**Strukturno ograničenje**
Naziv atributa	**Tip atributa**	**Vrijednost atributa**	**Međuzavis-nost atributa jedne tabele**	**Međuzavis-nost atributa vise tabela**	
RegistrationNumber	String	Not NULL			INSERT Restricted Airports UPDATE Restricted Airports UPDATE Cascade LogBook RotablePartsAircraft RotablePartsStock DELETE Restricted LogBook RotablePartsAircraft RotablePartsStock
SerialNumber	String	Not NULL			
LastUpdate	DateTime	Not NULL			
ID_Airports	Integer	Not NULL			
LastACHours	Double	Not NULL			
LastACCycles	Integer	Not NULL			

Kao rezultat analize scenarija SK i pravljenja konceptualnog modela dobija se logička struktura i ponašanje softverskog sistema:



Slika 8: Softverski sistem

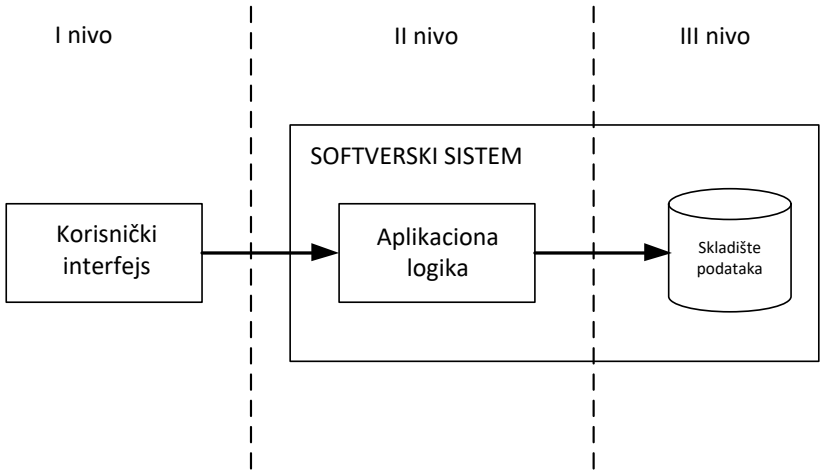
4.3 FAZA PROJEKTOVANJA

Faza projektovanja opisuje fizičku strukturu i ponašanje softverskog sistema. Projektovanje arhitekture softverskog sistema obuhvata projektovanje korisničkog interfejsa (projektovanje kontrolera korisničkog interfejsa i ekranskih formi), aplikacione logike (projektovanje kontrolera aplikacione logike, sistemske operacije) i skladišta podataka (broker baze podataka).

Arhitektura sistema je tronivovska i sastoji se od sledećih nivoa:

- korisnički interfejs
- aplikaciona logika
- skladište podataka

Nivo korisničkog interfejsa ja na strani klijenta, dok su aplikaciona logika i skladište na strani servera.

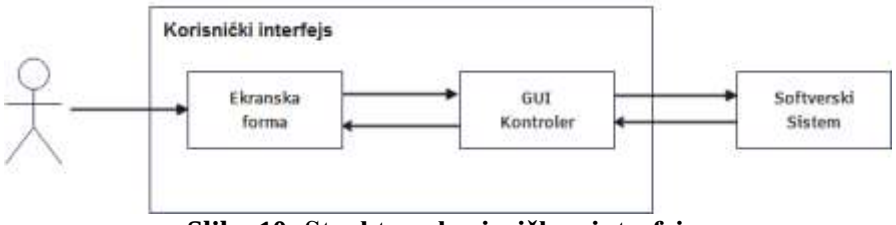


Slika 9: Tronivovska arhitektura

4.3.1 Projektovanje korisničkog interfejsa

Korisnički interfejs predstavlja ulazno-izlaznu realizaciju softverskog sistema. Sastoji se od:

- ekranske forme
- kontrolera korisničkog interfejsa



Slika 10: Struktura korisničkog interfejsa

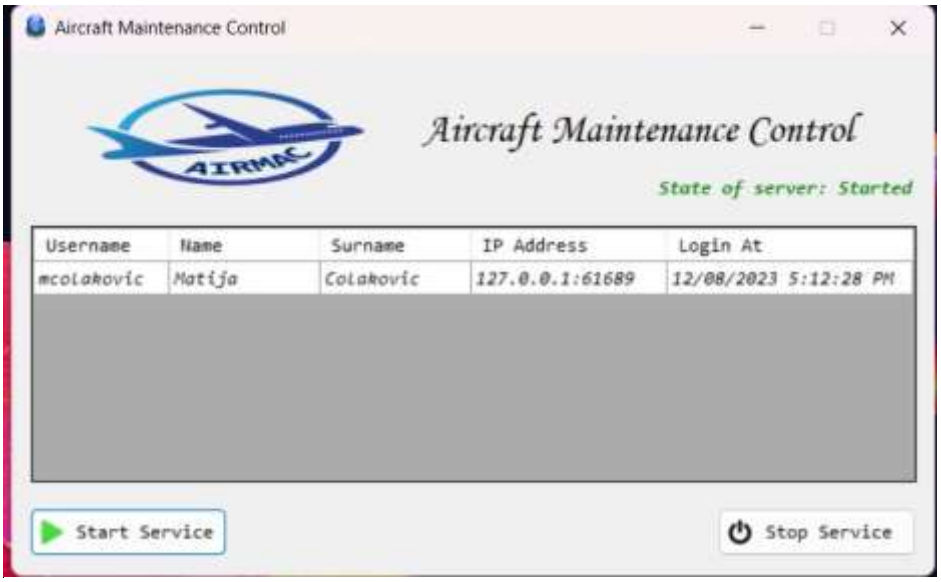
4.3.1.1 Projektovanje ekranskih formi

Korisnički interfejs je definisan preko skupa ekranskih formi. Scenario korišćenja ekranskih formi je direktno povezan sa scenarijima slučajeva korišćenja.

Postoje dva aspekta projektovanja ekranske forme:

- Projektovanje scenarija SK koji se izvode preko ekranske forme
- Projektovanje metoda ekranske forme

Na serverskoj strani programa projektovana je korisnička forma koja prihvata zahtjeve korisnika i prikazuje evidenciju prijavljenih korisnika.



Na klijentskoj strani prvo je potrebno prijaviti se kako bi se dospjelo u mogućnost korišćenja aplikacije.

Nakon prijavljivanja, sistem prikazuje klijentu glavnu ekransku formu iz koje se može doći do ostalih ekranskih formi.

SK1: Slučaj korišćenja – Registracija korisnika na sistem

Naziv SK
Registracija korisnika na sistem

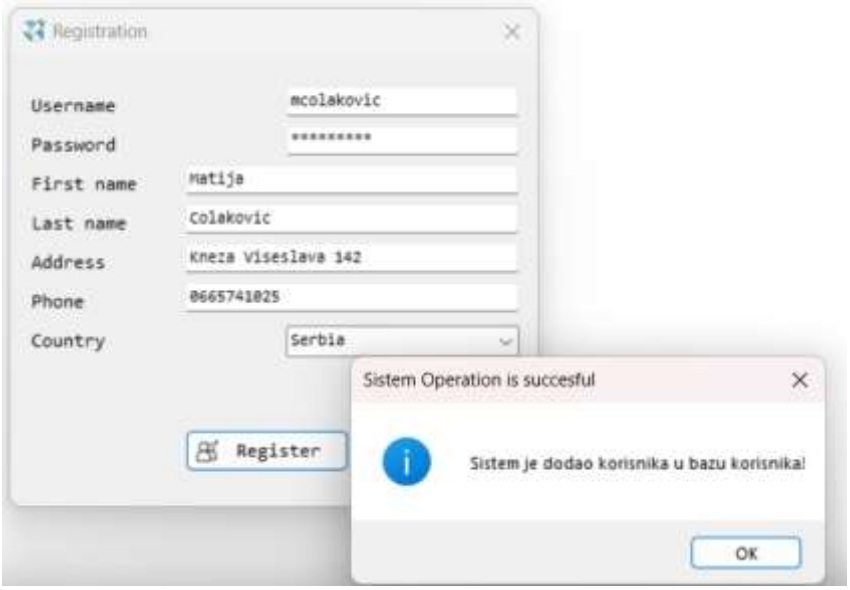
Aktori SK
Aviomehaničar/avio servis

Učesnici SK
Aviomehaničar/avio servis i sistem

Preduslov
Sistem je funkcionalan, sistem prikazuje formu za unos podataka.

Osnovni scenario SK

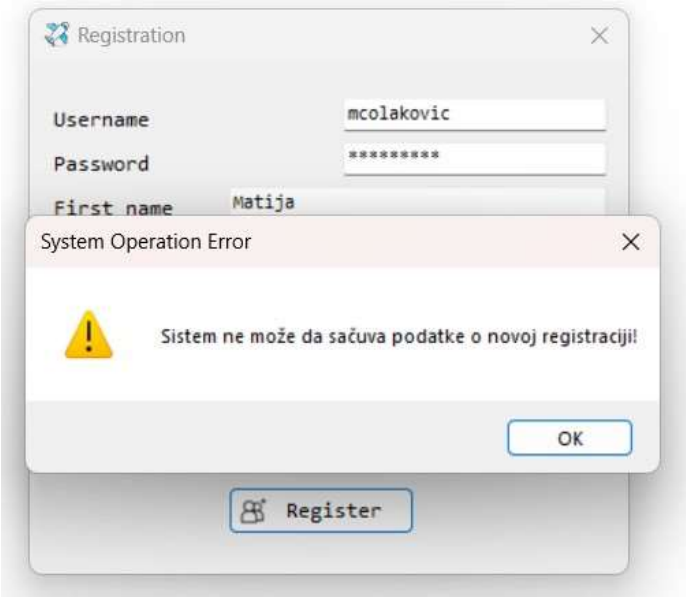
1. Aviomehaničar/avio servis unosi informacije o korisniku. (APUSO)
2. Aviomehaničar/avio servis poziva sistem da na osnovu unijetih podataka kreira korisnički nalog. (APSO)
3. Sistem kreira korisnički nalog na osnovu unijetih podataka. (SO)
4. Sistem prikazuje poruku „Sistem je dodao korisnika u bazu korisnika! “ (IA)



Opis akcije: Klikom na dugme „Register“ aviomehaničar poziva sistemsku operaciju *ZapamtiRegistraciju(User)*.

Alternativna scenarija

4.1 Ukoliko sistem ne može da kreira korisnički nalog on aviomehaničaru/avio servisu prikazuje poruku „Sistem ne može da sačuva podatke o novoj registraciji!“. (IA)


Registration
Username
mcolakovic
Password
****
First name
Matija
System Operation Error
Sistem ne može da sačuva podatke o novoj registraciji!
OK
Register

SK2: Slučaj korišćenja – Prijavljivanje korisnika na sistem

Naziv SK
Prijavljivanje korisnika na sistem

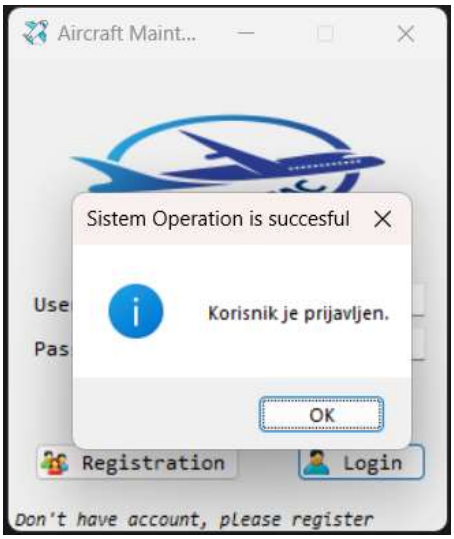
Aktori SK
Aviomehaničar/avio servis

Učesnici SK
Aviomehaničar/avio servis i sistem

Preduslov
Sistem je funkcionalan, korisnički nalog postoji.

Osnovni scenario SK

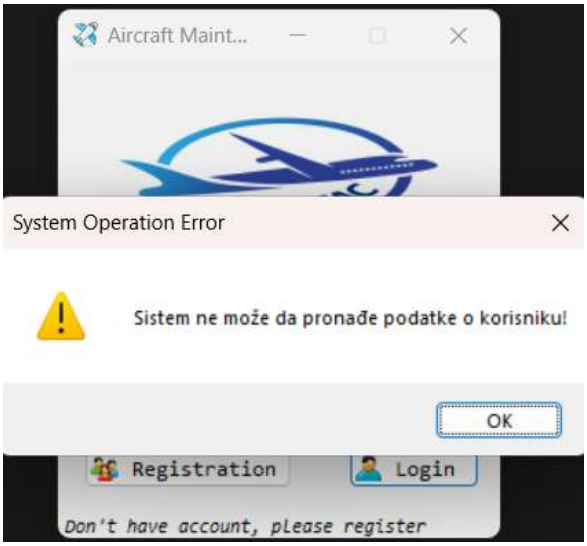
1. Aviomehaničar/avio servis unosi korisničko ime i lozinku. (APUSO)
2. Aviomehaničar/avio servis poziva sistem da na osnovu unijetih podataka pronađe korisnički nalog. (APSO)
3. Sistem pronalazi korisnički nalog i prijavljuje korisnika na sistem. (SO)
4. Sistem prikazuje poruku „Korisnik je prijavljen.“. (IA)



Opis akcije: Klikom na dugme „Login“ aviomehaničar poziva sistemsku operaciju *VerifikacijaKorisnika(User)*.

Alternativna scenarija

4.1 Ukoliko sistem ne može da pronađe korisnički nalog on aviomehaničaru/avio servisu prikazuje poruku „Sistem ne može da pronađe podatke o korisniku.“. (IA)


Aircraft Maint...
System Operation Error
Sistem ne može da pronađe podatke o korisniku!
OK
Registration
Login
Don't have account, please register

SK3: Slučaj korišćenja – Unos aviona

Naziv SK
Unos aviona

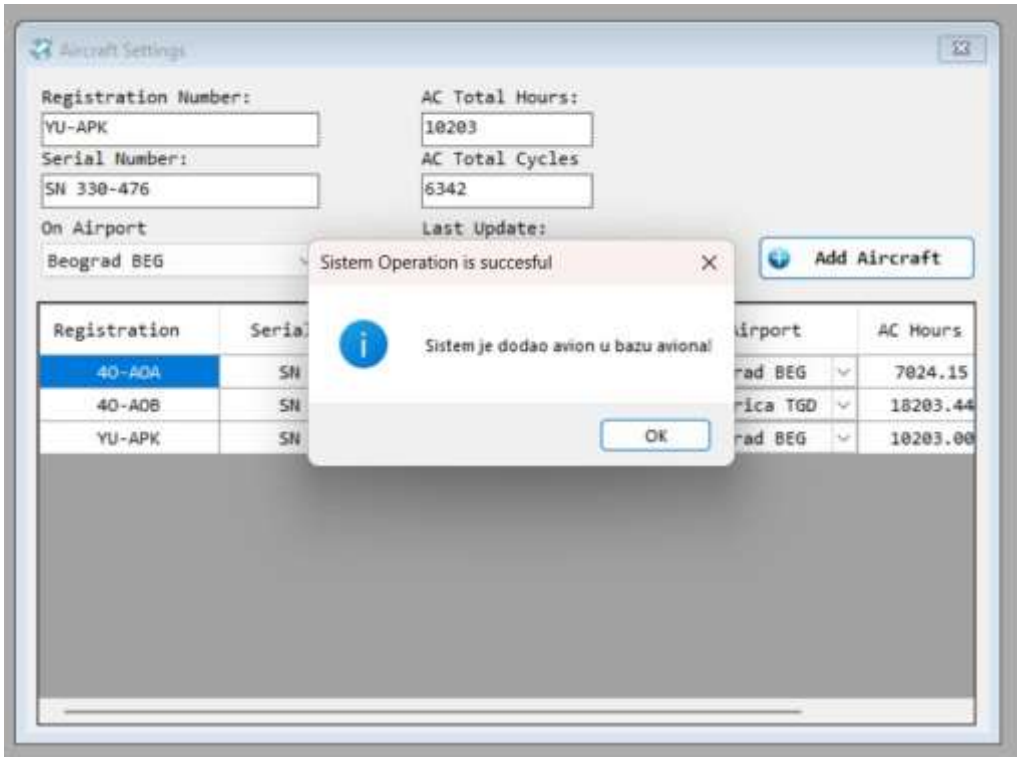
Aktori SK
Aviomehaničar

Učesnici SK
Aviomehaničar i sistem

Preduslov
Sistem je funkcionalan, aviomehaničar je prijavljen na sistem pod svojim nalogom, sistem prikazuje formu za unos podataka. Šifarnici aerodroma su inicijalizovani. Učitana je lista aerodroma.

Osnovni scenario SK

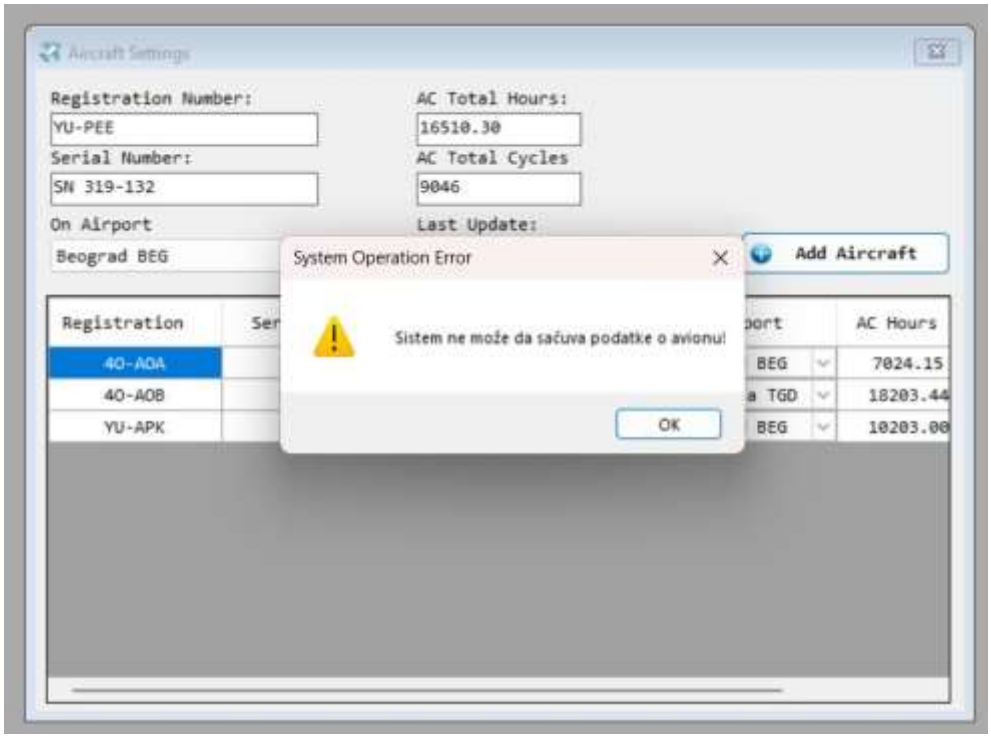
1. Aviomehaničar unosi podatke koji identifikuju avion. (APUSO)
2. Aviomehaničar poziva sistem da na osnovu unijetih podataka sačuva avion. (APSO)
3. Sistem kreira avion na osnovu unijetih podataka. (SO)
4. Sistem prikazuje poruku „Sistem je dodao avion u bazu aviona!“. (IA)



Opis akcije: Klikom na dugme „Add Aircraft“ aviomehaničar poziva sistemsku operaciju *DodajAvion(Aircraft)*.

Alternativna scenarija

4.1 Ukoliko sistem ne može da kreira avion on aviomehaničaru prikazuje poruku „Sistem ne može da sačuva podatke o avionu!“. (IA)



SK4: Slučaj korišćenja – Unos realizovanih letova aviona

Naziv SK
Unos realizovanih letova aviona

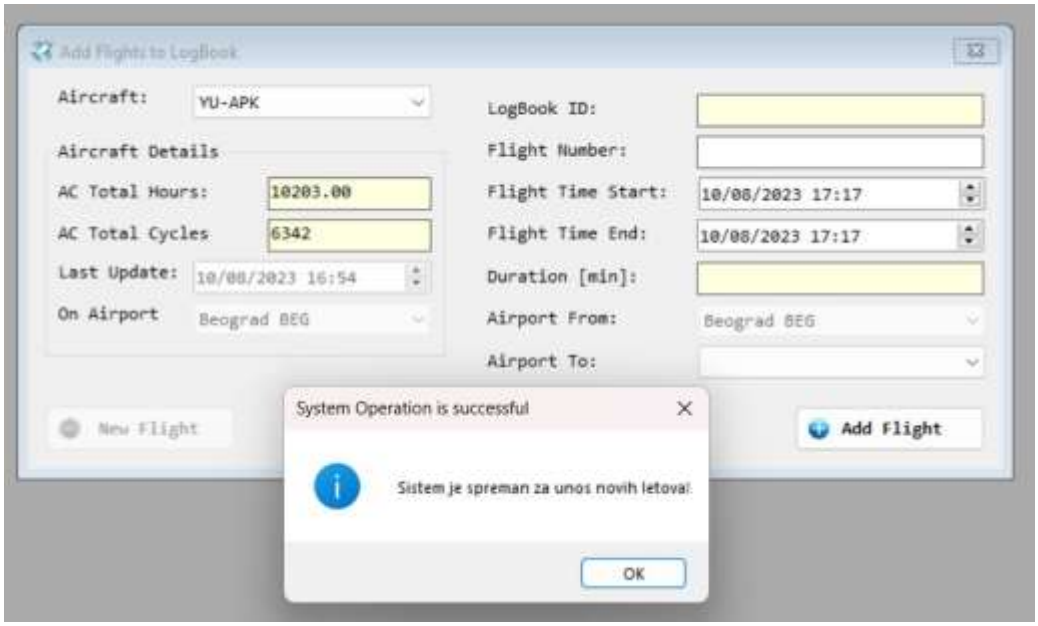
Aktori SK
Aviomehaničar

Učesnici SK
Aviomehaničar i sistem

Preduslov
Sistem je funkcionalan, aviomehaničar je ulogovan na sistem pod svojim nalogom, sistem prikazuje formu za unos podataka. Šifarnici aviona i aerodroma su inicijalizovani. Učitana lista aviona i aerodroma.

Osnovni scenario SK

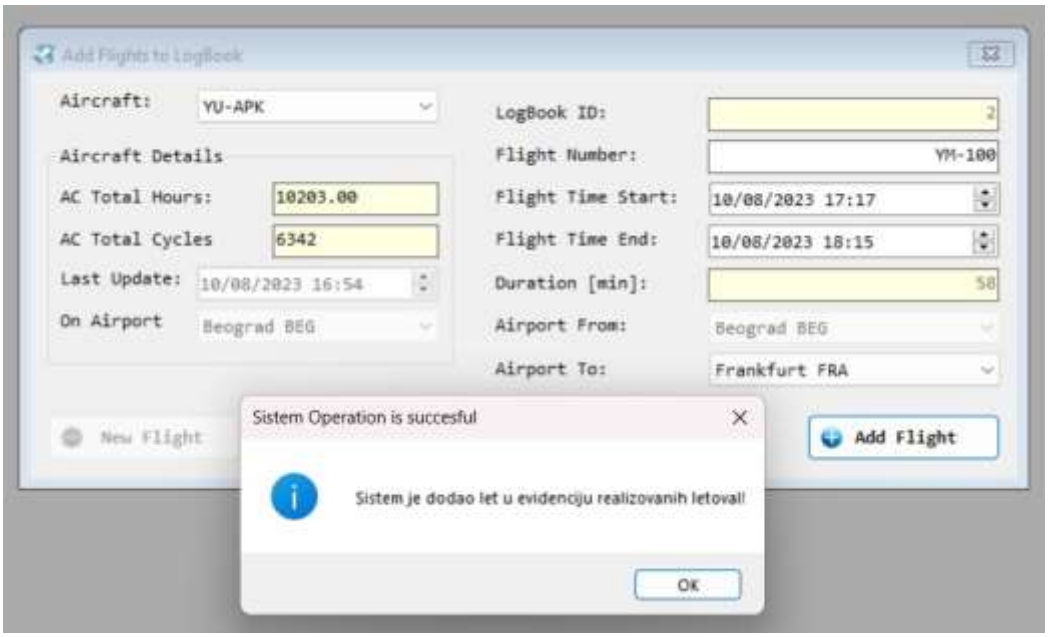
1. Aviomehaničar bira avion za koji želi unos realizovanih letova. (APUSO)
2. Aviomehaničar poziva sistem da na osnovu izabranog aviona vrati relevantne podatke bitne za unos letova (poziciju aviona nakon zadnjeg realizovanog leta, kao i podatke o naletu aviona). (APSO)
3. Sistem traži podatke o avionu na osnovu zadatog kriterijuma. (SO)
4. Sistem prikazuje tražene podatke o avionu uz poruku „Sistem je spreman za unos novih letova“. (IA)



Opis akcije: Izborom aviona iz ComboBox kontrole aviomehaničar poziva sistemsku operaciju *UcitajAvion(Aircraft)*.

5. Aviomehaničar unosi podatke koji jednoznačno identifikuju realizovani let. (APUSO)

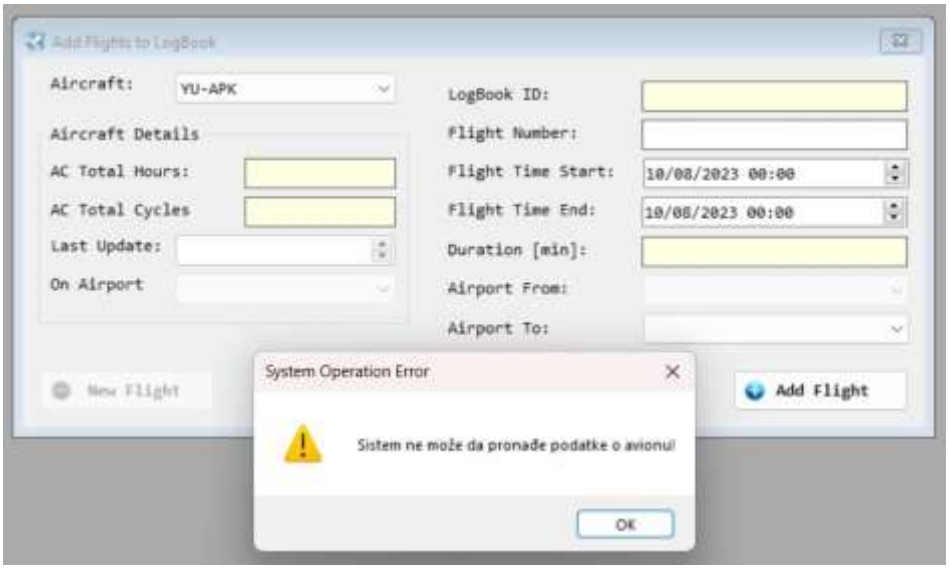
6. Aviomehaničar kontroliše da li je korektno unio podatke o realizovanom letu. (ANSO)
7. Aviomehaničar poziva sistem da zapamti podatke o realizovanom letu. (APSO)
8. Sistem pamti podatke o realizovanom letu. (SO)
9. Sistem prikazuje podatke o realizovanom letu uz poruku „Sistem je dodao let u evidenciju realizovanih letova“. (IA)



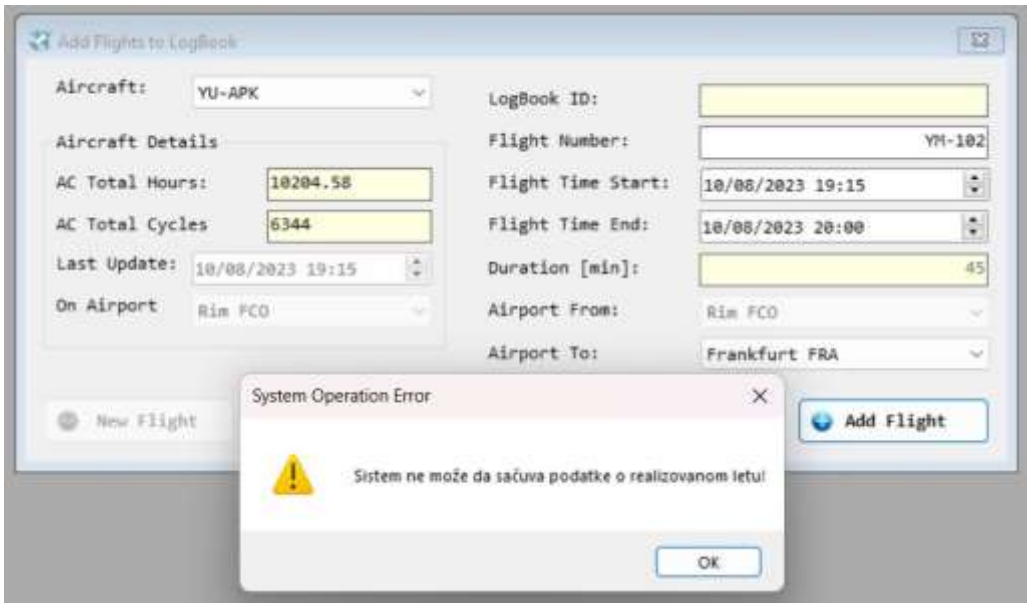
Opis akcije: Klikom na dugme „Add Flight“ aviomehaničar poziva sistemsku operaciju *ZapamtiLet(Logbook)*.

Alternativna scenarija

4.1 Ukoliko sistem ne može da nađe tražene podatke o avionu on aviomehaničaru prikazuje poruku „Sistem ne može da pronađe podatke o avionu“. Prekida se izvršavanje scenarija. (IA)



9.1 Ukoliko sistem ne može da zapamti podatke o realizovanom letu on aviomehaničaru prikazuje poruku „Sistem ne može da sačuva podatke o realizovanom letu“.



SK5: Slučaj korišćenja – Pregled realizovanih letova aviona

Naziv SK
Pregled realizovanih letova aviona

Aktori SK
Aviomehaničar

Učesnici SK
Aviomehaničar i sistem

Preduslov
Sistem je funkcionalan, aviomehaničar je ulogovan na sistem pod svojim nalogom, sistem prikazuje formu za unos podataka. Šifarnici aviona i aerodroma su inicijalizovani. Učitana lista aviona i aerodroma.

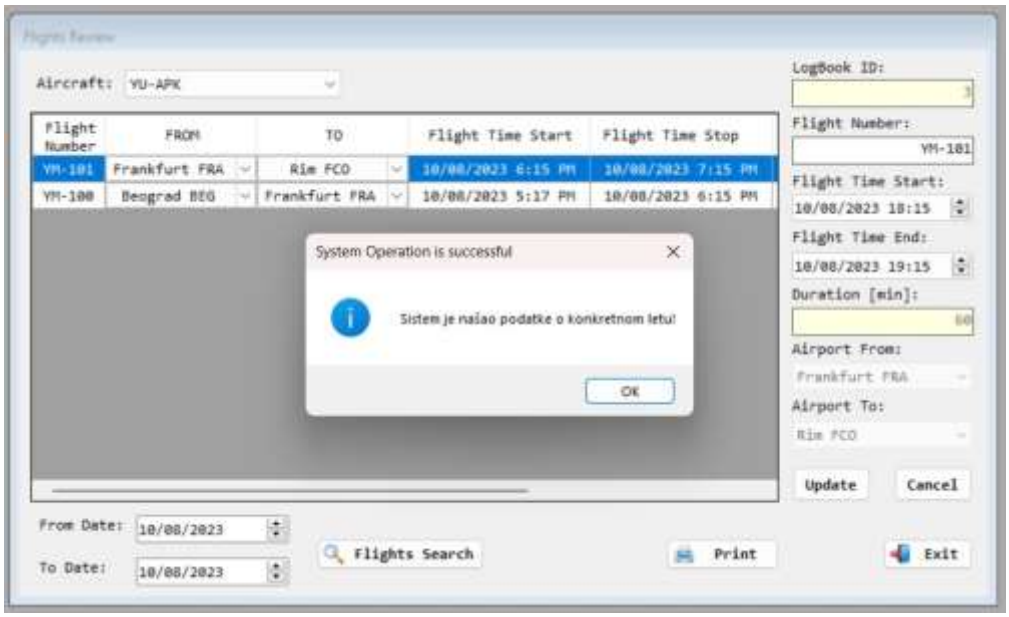
Osnovni scenario SK

1. Aviomehaničar unosi vremenski raspon i bira avion za koji želi pregled realizovanih letova. (APUSO)
2. Aviomehaničar poziva sistem da na osnovu izabranog aviona i vremenskog raspona vrati realizovane letove. (APSO)
3. Sistem traži podatke o realizovanim letovima na osnovu definisanog kriterijuma. (SO)
4. Sistem vraća tražene podatke o realizovanim letovima uz poruku „Sistem je našao sledeće realizovane letove“. (IA)

Opis akcije: Izborom aviona iz ComboBox kontrole, unošenjem vremenskog intervala i klikom na dugme „Flights Search“ aviomehaničar poziva sistemsku operaciju *UcitajListuLetova(Logbook).*

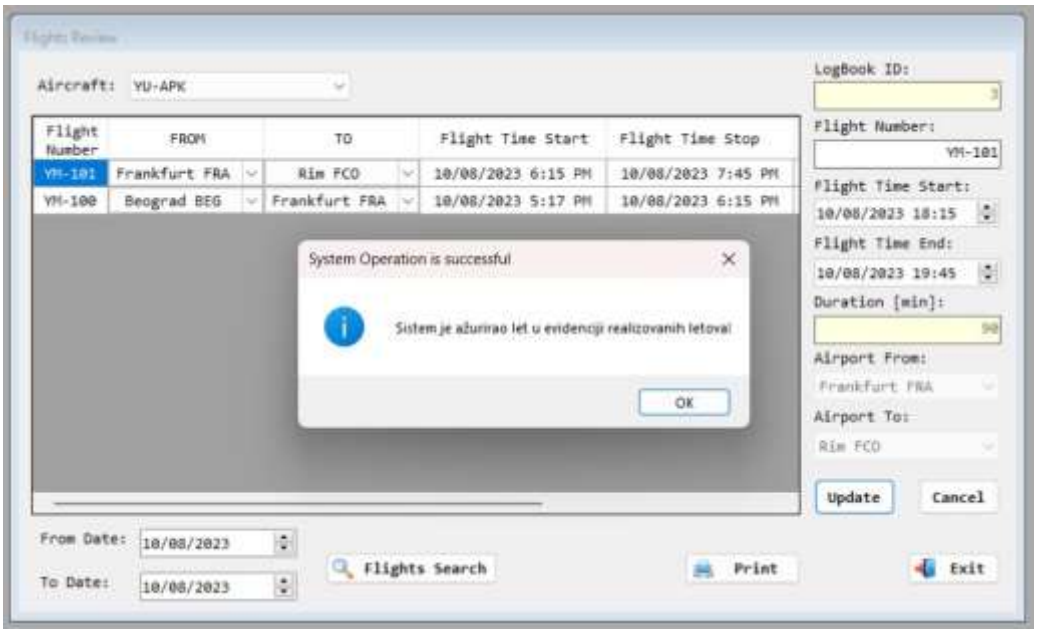
5. Aviomehaničar bira konkretan let. (APUSO)
6. Aviomehaničar zahtijeva od sistema podatke o konkretnom letu. (APSO)

7. Sistem traži podatke o konkretnom letu. (SO)
8. Sistem vraća podatke o konkretnom letu uz poruku „Sistem je našao podatke o konkretnom letu“. (IA)



Opis akcije: Duplim klikom na konkretan let u DataGridView kontroli aviomehaničar poziva sistemsku operaciju *PronadjiLet(Logbook)*.

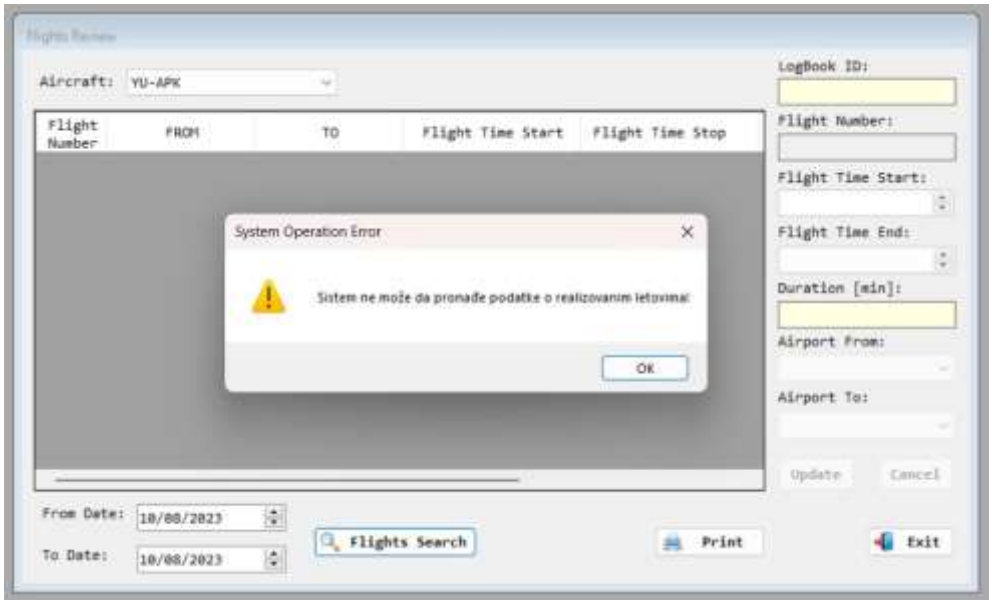
9. Aviomehaničar ažurira podatke za korektan let. (APUSO)
10. Aviomehaničar poziva sistem da zapamti podatke ažuriranog leta. (APSO)
11. Sistem pamti podatke o ažuriranom letu. (SO)
12. Sistem prikazuje podatke o ažuriranom letu uz poruku „Sistem je ažurirao let u evidenciji realizovanih letova“. (IA)



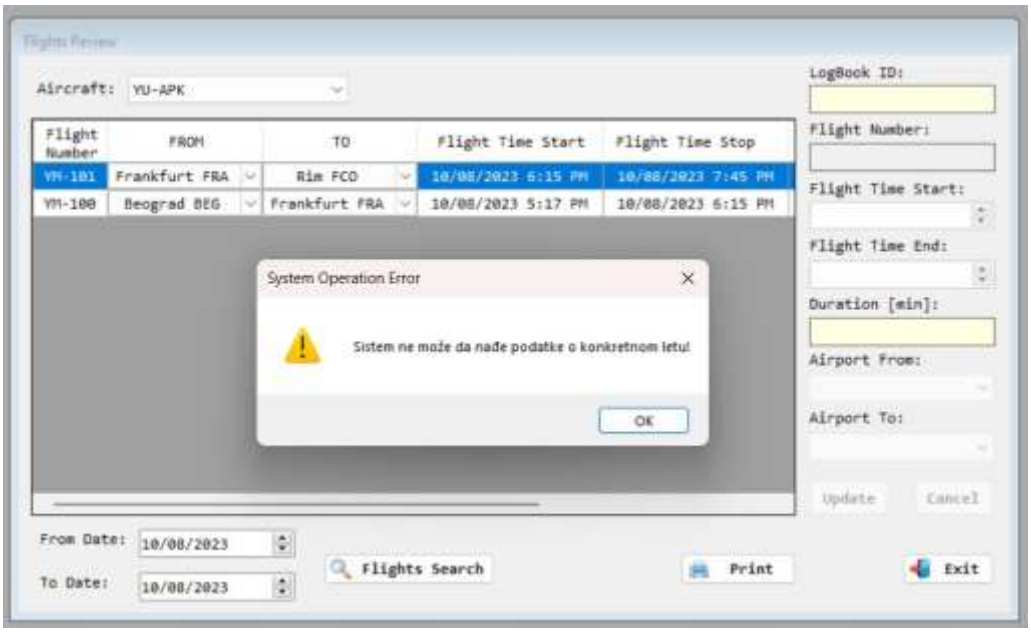
Opis akcije: Klikom na dugme „Update“ aviomehaničar poziva sistemsku operaciju *AzurirajLet(Logbook)*.

Alternativna scenarija

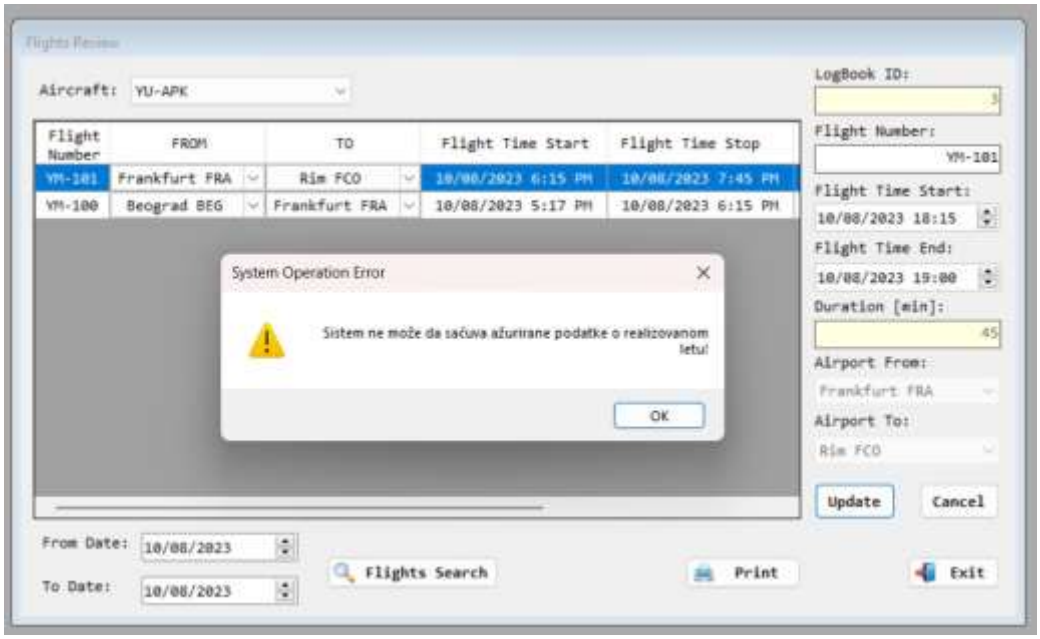
4.1 Ukoliko sistem ne može da nađe tražene podatke o realizovanim letovima on aviomehaničaru prikazuje poruku „Sistem ne može da pronađe podatke o realizovanim letovima“. Prekida se izvršavanje scenarija. (IA)



8.1 Ukoliko sistem ne može da nađe tražene podatke o konkretnom letu on aviomehaničaru prikazuje poruku „Sistem ne može da nađe podatke o konkretnom letu“. Prekida se izvršavanje scenarija. (IA)



12.1 Ukoliko sistem ne može da zapamti podatke nakon ažuriranja leta on aviomehaničaru prikazuje poruku „Sistem ne može da sačuva ažurirane podatke o realizovanom letu“.


Aircraft: YU-APK
Flight Number
FROM
TO
Flight Time Start
Flight Time Stop
YM-101
Frankfurt FRA
Rim FCO
10/08/2023 6:15 PM
10/08/2023 7:45 PM
YM-100
Beograd BEG
Frankfurt FRA
10/08/2023 5:17 PM
10/08/2023 6:15 PM
System Operation Error
Sistem ne može da sačuva ažurirane podatke o realizovanom letu!
OK
LogBook ID:
Flight Number:
YM-101
Flight Time Start:
10/08/2023 18:15
Flight Time End:
10/08/2023 19:00
Duration [min]:
45
Airport From:
Frankfurt FRA
Airport To:
Rim FCO
Update
Cancel
From Date:
10/08/2023
To Date:
10/08/2023
Flights Search
Print
Exit

SK6: Slučaj korišćenja – Kreiranje kartona dijela aviona

Naziv SK
Kreiranje kartona dijela aviona

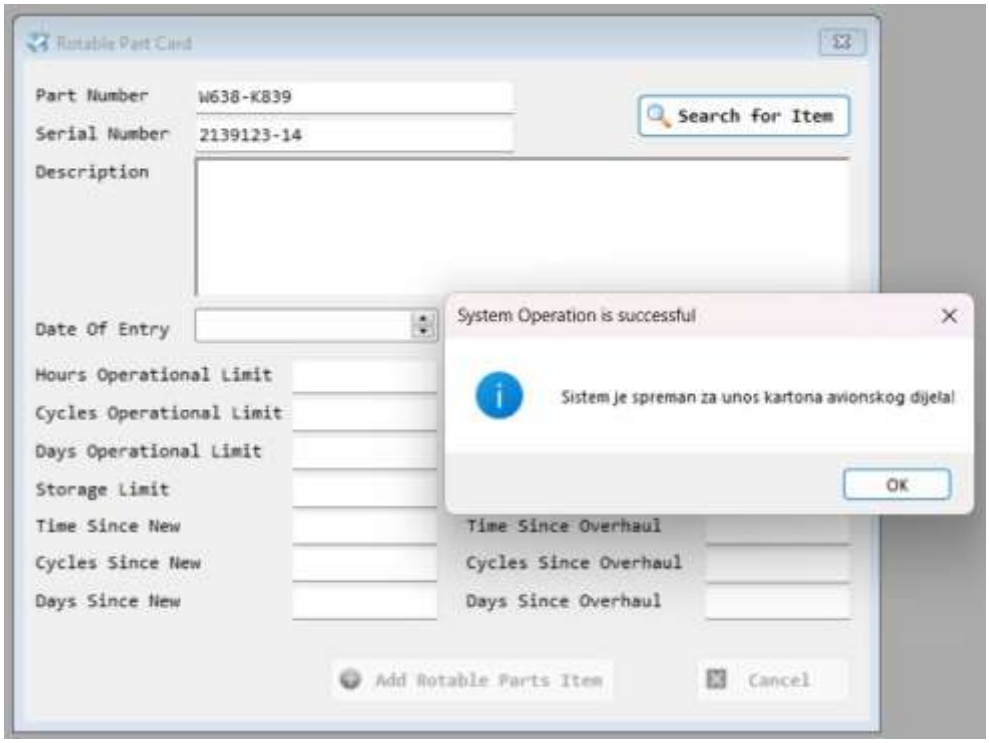
Aktori SK
Aviomehaničar

Učesnici SK
Aviomehaničar i sistem

Preduslov
Sistem je funkcionalan, aviomehaničar je ulogovan na sistem pod svojim nalogom, sistem prikazuje formu za unos podataka. Šifarnici aviona i aerodroma su inicijalizovani.

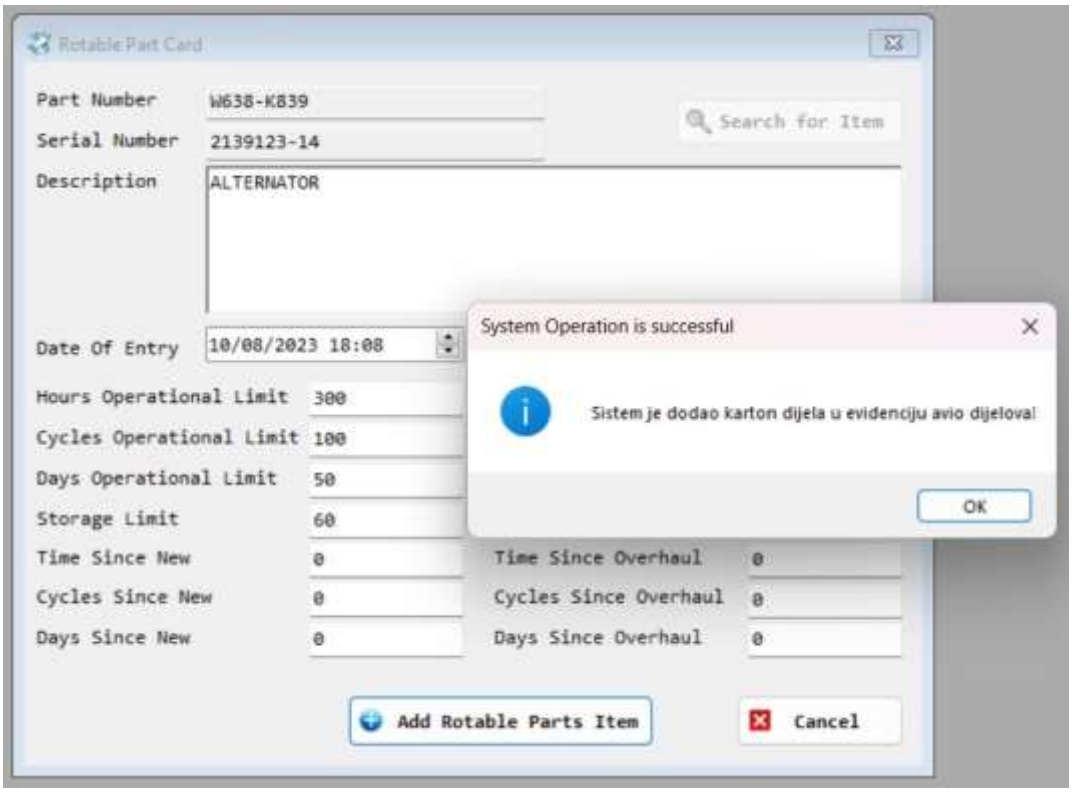
Osnovni scenario SK

1. Aviomehaničar unosi PN (Part number) i SN (Serial number) avionskog dijela radi provjere da li je karton avionskog dijela već kreiran. (APUSO)
2. Aviomehaničar poziva sistem da na osnovu PN (Part number) i SN (Serial number) kao parametara provjeri da li je karton avoinskog dijela već unešen u evidenciju avionskih dijelova. (APSO)
3. Sistem traži karton avionskog dijela po zadatom kriterijumu. (SO)
4. Sistem vraća informaciju da avionski dio ne postoji u evidenciji avionskih dijelova i vraća poruku „Sistem je spreman za unos kartona avionskog dijela“. (IA)



Opis akcije: Klikom na dugme „Search for Item“ aviomehaničar poziva sistemsku operaciju *NadjiKarton(RotableParts)*

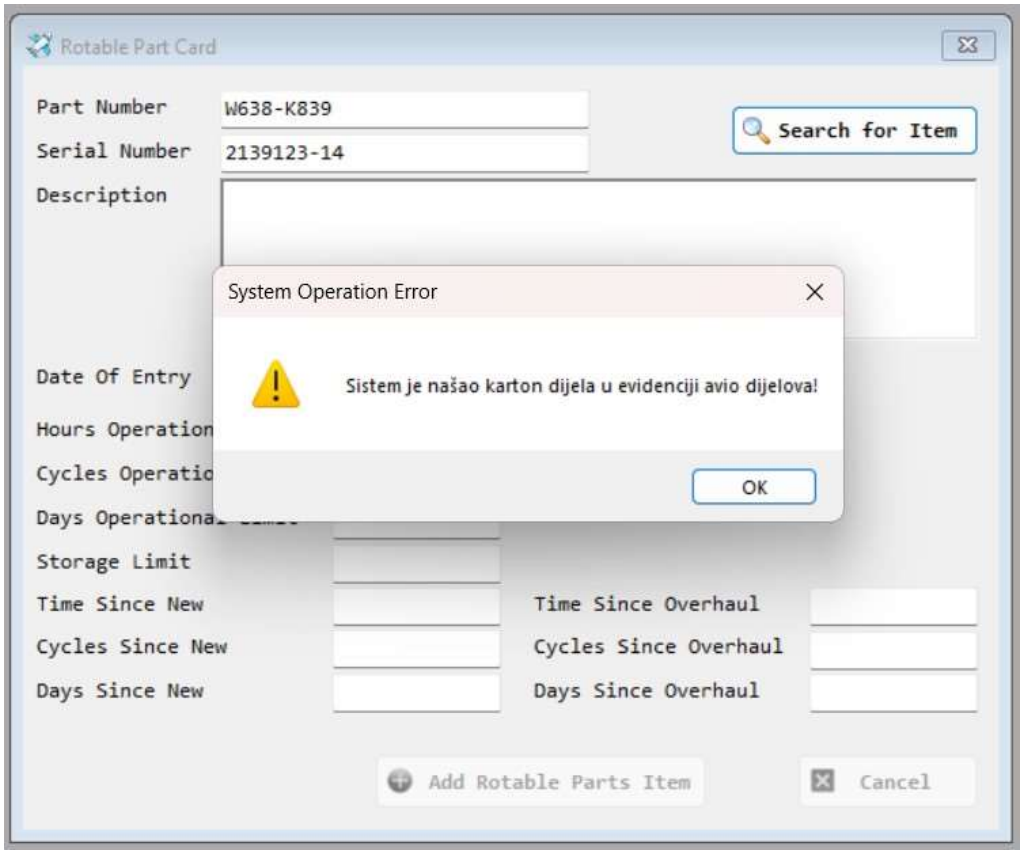
5. Aviomehaničar unosi podatke relevantne za opis avionskog dijela kao i podatke o njegovim resursima. (APUSO)
6. Aviomehaničar kontroliše da li je korektno unio podatke u karton avionskog dijela. (ANSO)
7. Aviomehaničar poziva sistem da zapamti podatke iz kartona avionskog dijela. (APSO)
8. Sistem pamti podatke iz kartona avionskog dijela. (SO)
9. Sistem prikazuje podatke iz kartona avionskog dijela uz poruku „Sistem je dodao karton dijela u evidenciju avio dijelova“. (IA)



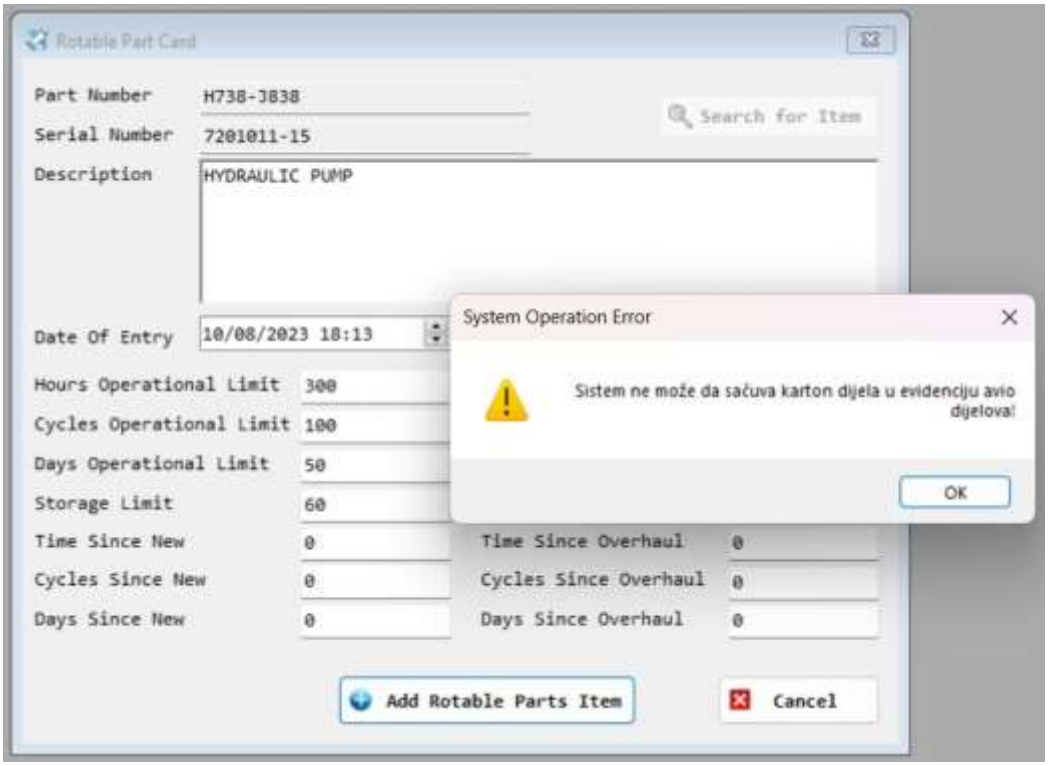
Opis akcije: Klikom na dugme „Add Rotable Parts Item“ aviomehaničar poziva sistemsku operaciju *ZapamtiKarton(RotablePartsLog).*

Alternativna scenarija

4.1 Ukoliko sistem nađe da avionski dio postoji u evidenciji avionskih dijelova sistem šalje aviomehaničaru poruku „Sistem je našao karton dijela u evidenciji avio dijelova“. Prekida se izvršavanje scenarija. (IA)



9.1 Ukoliko sistem ne može da zapamti podatke iz kartona avionskog dijela on aviomehaničaru šalje poruku „Sistem ne može da sačuva karton dijela u evidenciju avio dijelova“.



SK7: Slučaj korišćenja – Pregled kartona dijela aviona

Naziv SK
Pregled kartona dijela aviona

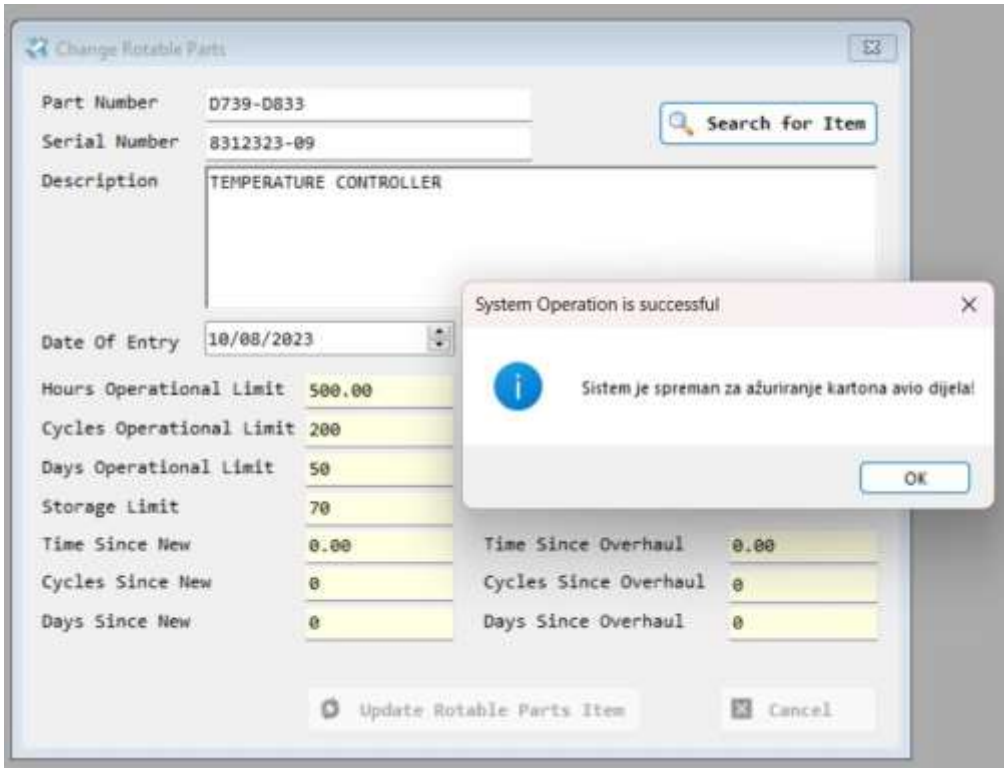
Aktori SK
Aviomehaničar

Učesnici SK
Aviomehaničar i sistem

Preduslov
Sistem je funkcionalan, aviomehaničar je ulogovan na sistem pod svojim nalogom, sistem prikazuje formu za unos podataka. Šifarnici aviona i aerodroma su inicijalizovani.

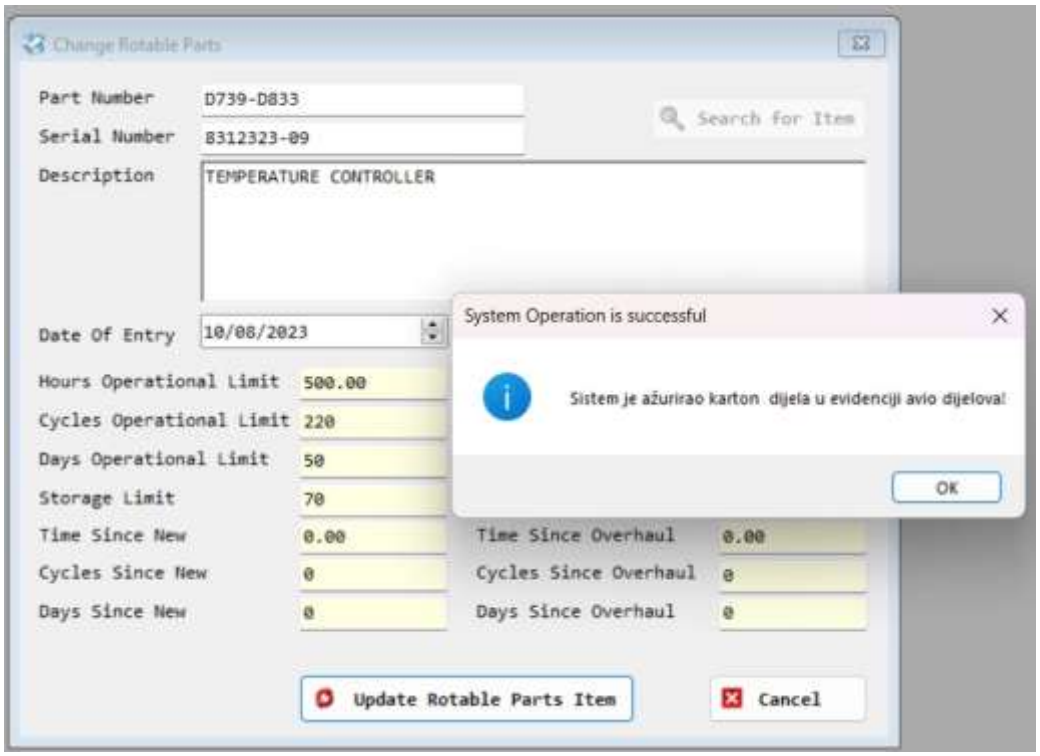
Osnovni scenario SK

1. Aviomehaničar unosi PN (Part number) i SN (Serial number) avionskog dijela radi pristupa kartonu avionskog dijela. (APUSO)
2. Aviomehaničar poziva sistem da na osnovu PN (Part number) i SN (Serial number) kao parametara nađe karton avionskog dijela iz evidencije avionskih dijelova. (APSO)
3. Sistem traži karton avionskog dijela po zadatom kriterijumu. (SO)
4. Sistem vraća podatke iz kartona avionskog dijela kao i poruku „Sistem je spreman za ažuriranje kartona avio dijela“. (IA)



Opis akcije: Klikom na dugme „Search for Item“ aviomehaničar poziva sistemske operacije *NadjiDioMagacin(RotablePartsStock).*

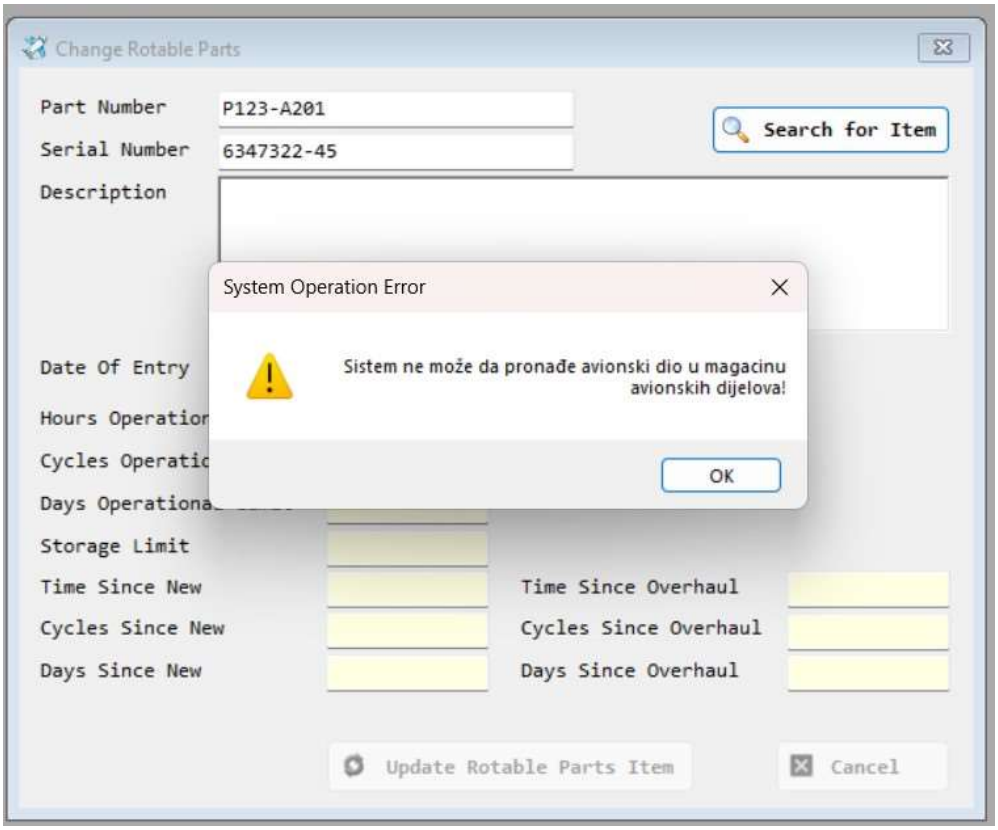
5. Aviomehaničar ažurira podatke iz kartona avionskog dijela. (APUSO)
6. Aviomehaničar kontroliše da li je korektno unio podatke u karton avionskog dijela. (ANSO)
7. Aviomehaničar poziva sistem da zapamti podatke iz kartona avionskog dijela. (APSO)
8. Sistem pamti podatke iz kartona avionskog dijela. (SO)
9. Sistem prikazuje podatke iz kartona avionskog dijela uz poruku „Sistem je ažurirao karton dijela u evidenciji avio dijelova“. (IA)



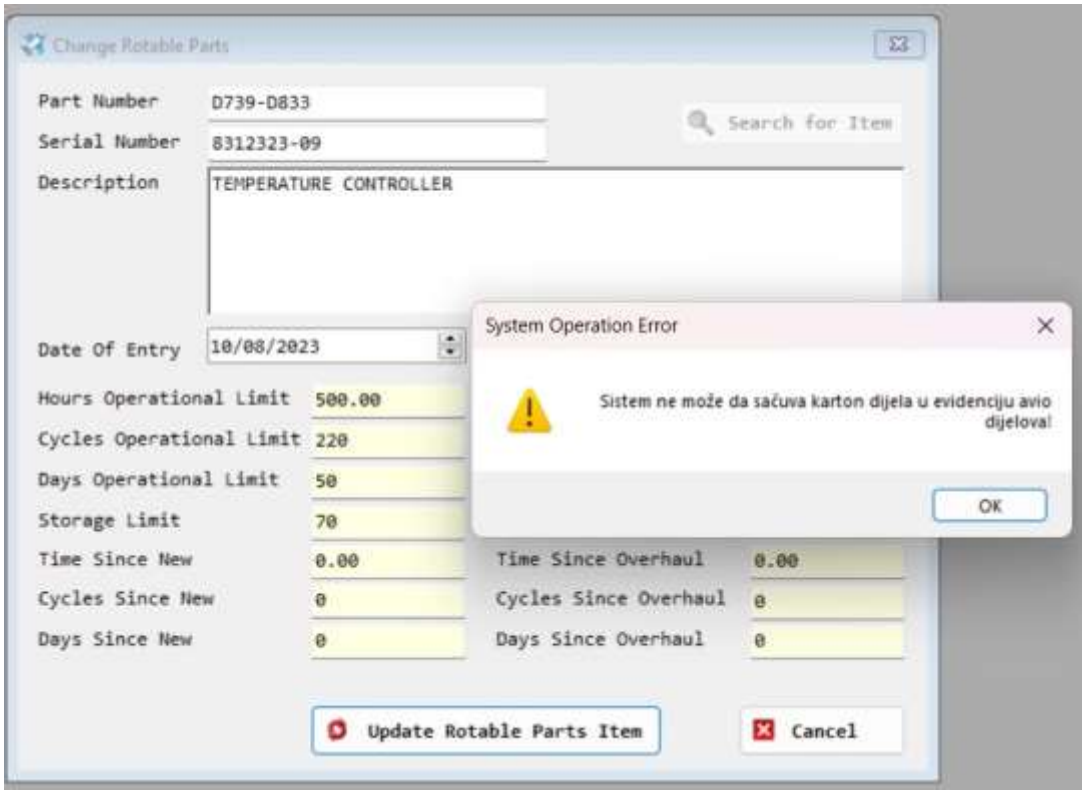
Opis akcije: Klikom na dugme „Update Rotable Parts Item“ aviomehaničar poziva sistemsku operaciju *AzurirajKarton(RotablePartsLog)*.

Alternativna scenarija

4.1 Ukoliko sistem ne može da nađe tražene podatke o avionskom dijelu sistem šalje aviomehaničaru poruku „Sistem ne može da pronađe karton dijela u evidenciji avio dijelova“. Prekida se izvršavanje scenarija. (IA)



9.1 Ukoliko sistem ne može da zapamti podatke iz kartona avionskog dijela on aviomehaničaru šalje poruku „Sistem ne može da sačuva kartona dijela u evidenciju avio dijelova“.



SK8: Slučaj korišćenja – Instaliranje dijela iz magacina dijelova na avion

Naziv SK
Instaliranje dijela iz magacina dijelova na avion

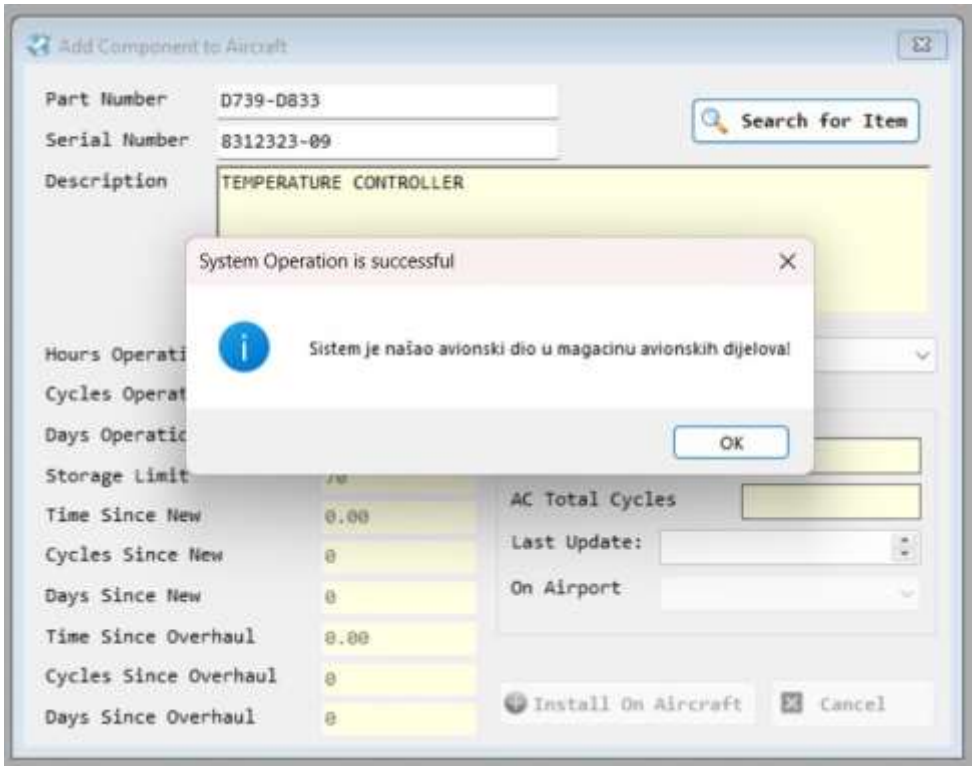
Aktori SK
Aviomehaničar

Učesnici SK
Aviomehaničar i sistem

Preduslov
Sistem je funkcionalan, aviomehaničar je ulogovan na sistem pod svojim nalogom, sistem prikazuje formu za unos podataka. Šifarnici aviona i aerodroma su inicijalizovani. Učitana lista aviona.

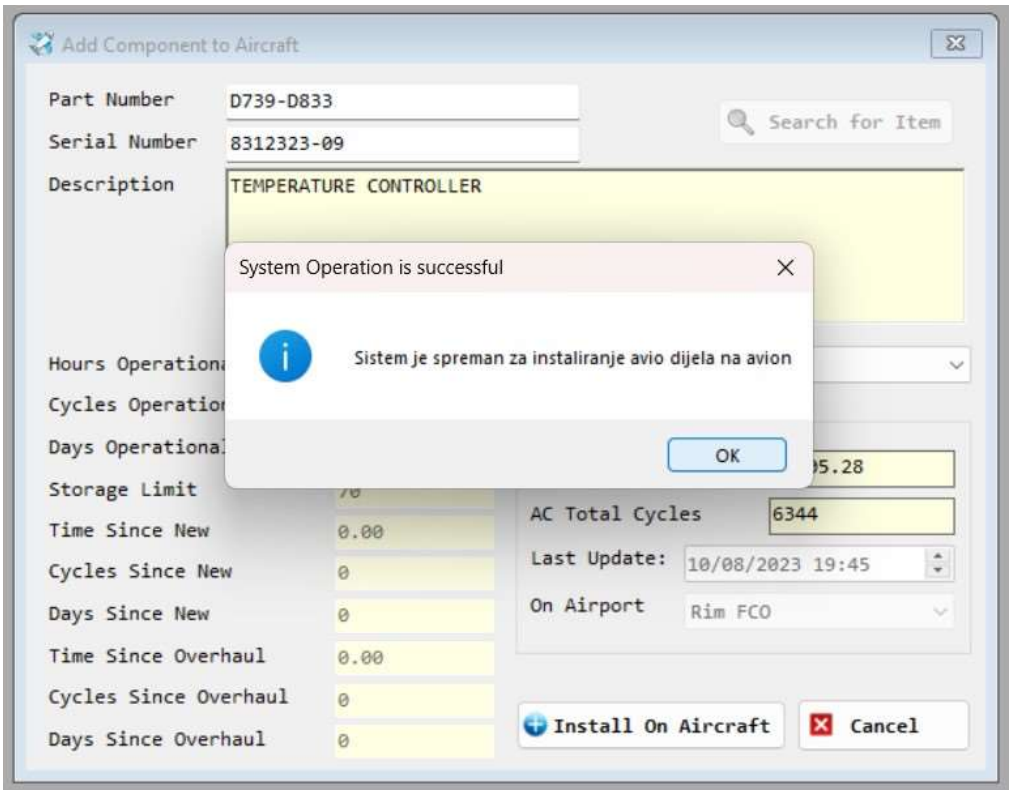
Osnovni scenario SK

1. Aviomehaničar unosi PN (Part number) i SN (Serial number) avionskog dijela radi pristupa kartonu avionskog dijela koji je lociran u magacinu. (APUSO)
2. Aviomehaničar poziva sistem da na osnovu PN (Part number) i SN (Serial number) kao parametara nađe avionski dio u magacinu avionskih dijelova. (APSO)
3. Sistem traži karton avionskog dijela po zadatom kriterijumu. (SO)
4. Sistem vraća podatke iz kartona avionskog dijela kao i poruku „Sistem je našao avionski dio u magacinu avionskih dijelova“. (IA)



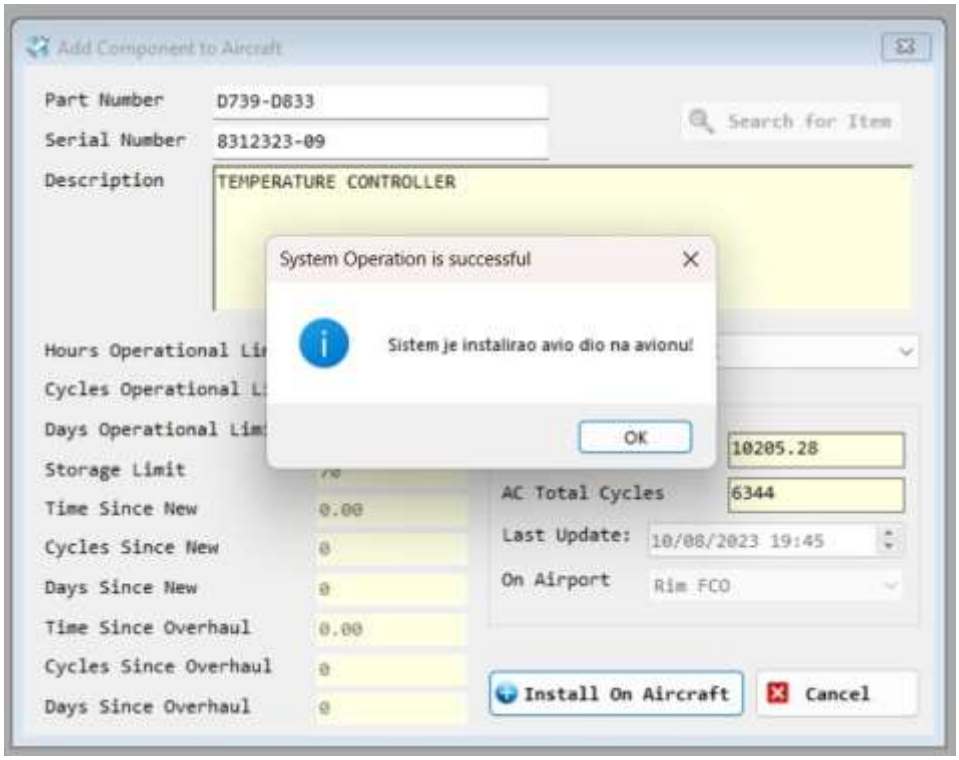
Opis akcije: Klikom na dugme „Search for Item“ aviomehaničar poziva sistemske operacije *NadjiDioMagacin(RotablePartsLog).*

5. Aviomehaničar bira avion na koji želi instalirati avionski dio. (APUSO)
6. Aviomehaničar poziva sistem da na osnovu izabranog aviona vrati relevantne podatke bitne za instalaciju avionskog dijela (nalet aviona). (APSO)
7. Sistem traži informacije o naletu aviona. (SO)
8. Sistem vraća informacije o naletu aviona korisniku uz poruku „Sistem je spreman za instaliranje avio dijela na avion“. (IA)



Opis akcije: Izborom aviona iz ComboBox kontrole aviomehaničar poziva sistemsku operaciju *UcitajAvion(Aircraft).*

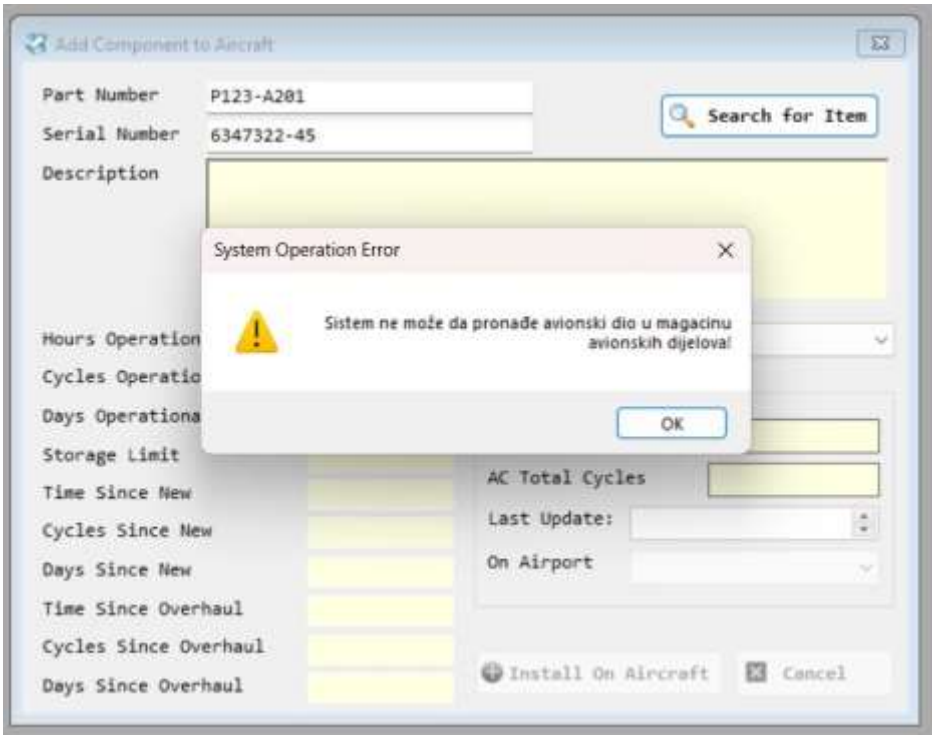
9. Aviomehaničar unosi podatke relevantne za instalaciju avionskog dijela. (APUSO)
10. Aviomehaničar kontroliše da li je korektno unio podatke o instalaciji avionskog dijela. (ANSO)
11. Aviomehaničar poziva sistem da zapamti podatke vezane za instalaciju avionskog dijela. (APSO)
12. Sistem pamti podatke vezane za instalaciju avionskog dijela. (SO)
13. Sistem prikazuje podatke vezane za instalaciju avionskog dijela uz poruku „Sistem je instalirao avio dio na avionu“. (IA)



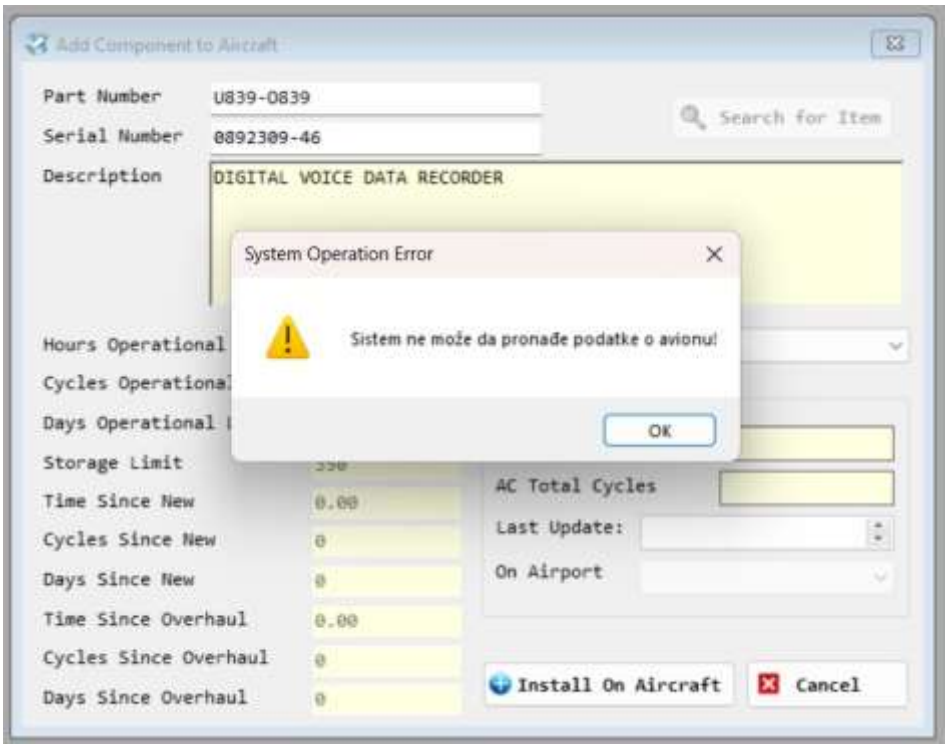
Opis akcije: Klikom na dugme „Install on Aircraft“ aviomehaničar poziva sistemsku operaciju *DodajDioAvion(RotablePartsLog).*

Alternativna scenarija

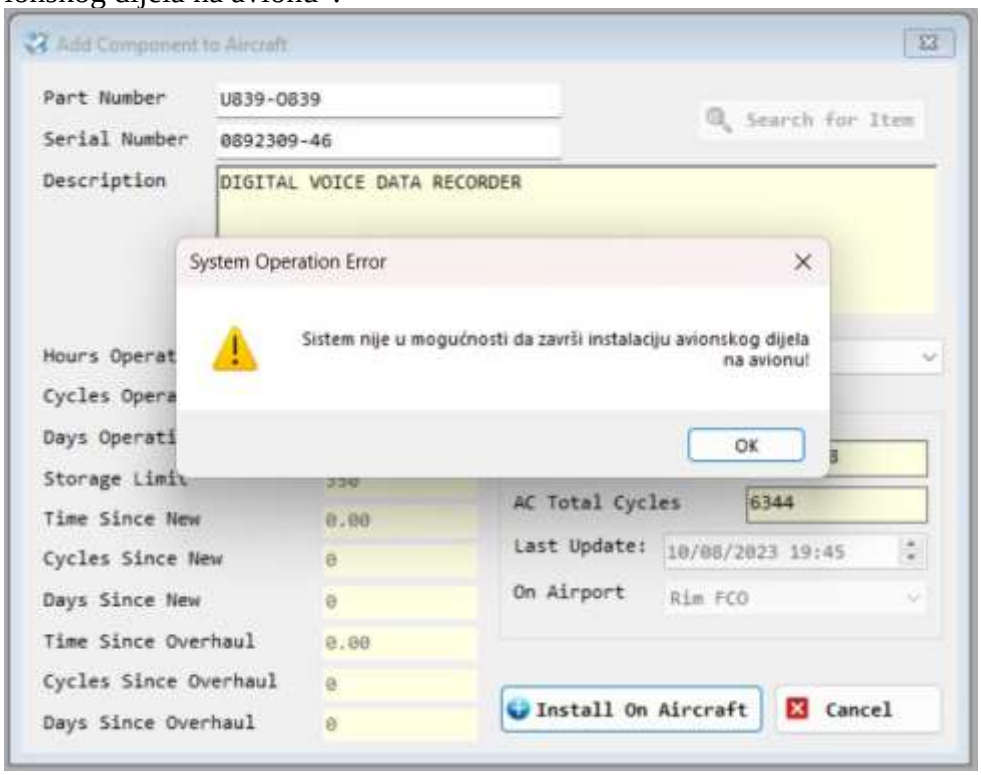
4.1 Ukoliko sistem ne može da nađe tražene podatke iz kartona avionskog dijela koji je lociran u magacinu on aviomehaničaru prikazuje poruku „Sistem ne može da pronađe avionski dio u magacinu avionskih dijelova“. Prekida se izvršavanje scenarija. (IA)



8.1 Ukoliko sistem ne može da nađe tražene podatke o avionu on aviomehaničaru prikazuje poruku „Sistem ne može da pronađe podatke o avionu“. Prekida se izvršavanje scenarija. (IA)



13.1 U slučaju da sistem ne može da završi instalaciju avionskog dijela na avion on aviomehaničaru šalje poruku „Sistem nije u mogućnosti da završi instalaciju avionskog dijela na avionu“.



SK9: Slučaj korišćenja – Skidanje dijela sa aviona i slanje u magacin

Naziv SK
Skidanje dijela sa aviona i slanje u magacin

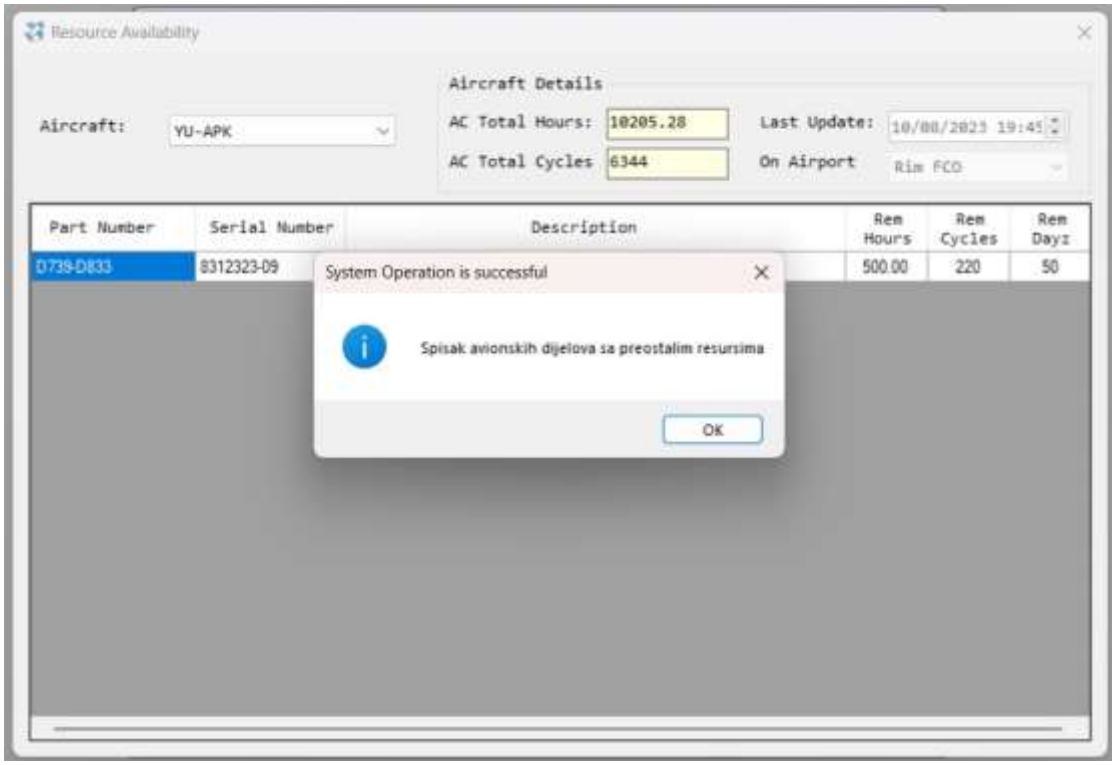
Aktori SK
Aviomehaničar

Učesnici SK
Aviomehaničar i sistem

Preduslov
Sistem je funkcionalan, aviomehaničar je ulogovan na sistem pod svojim nalogom, sistem prikazuje formu za unos podataka. Šifarnici aviona i aerodroma su inicijalizovani. Učitana lista aviona.

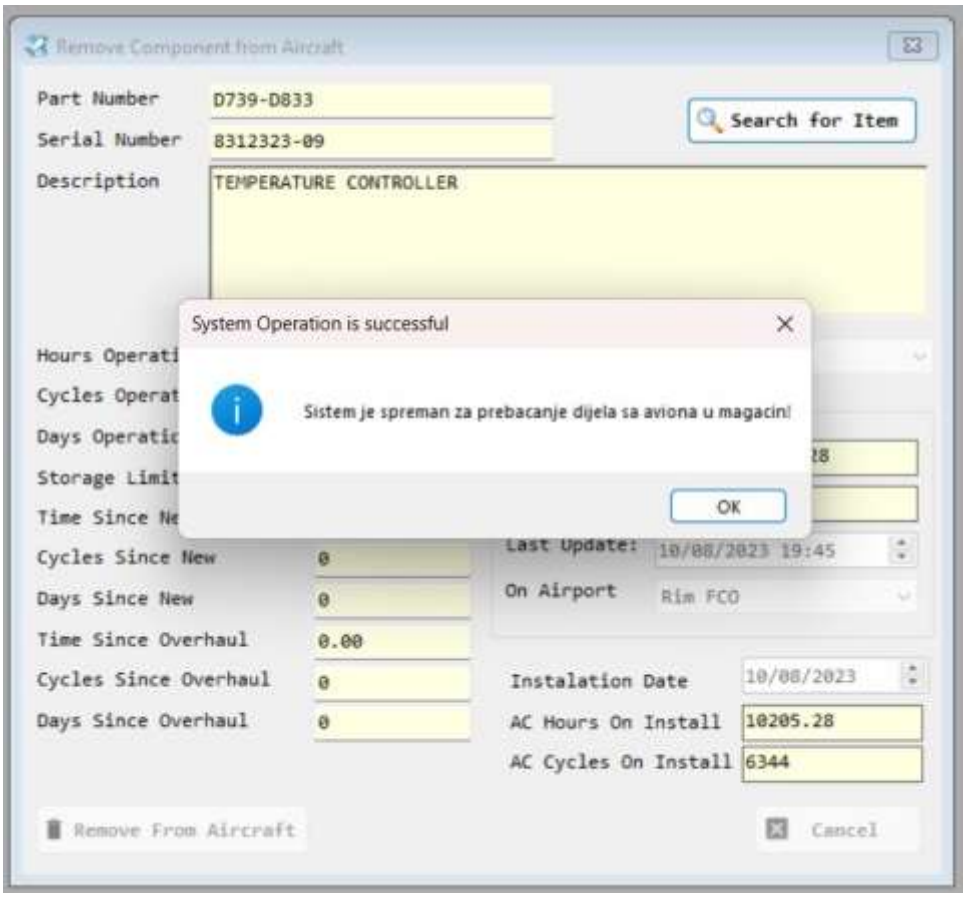
Osnovni scenario SK

1. Aviomehaničar bira avion sa koga želi skinuti avionski dio. (APUSO)
2. Aviomehaničar poziva sistem da na osnovu izabranog aviona vrati sve dijelove instalirane na njega. (APSO)
3. Sistem traži dijelove instalirane na njega po zadatom kriterijumu. (SO)
4. Sistem vraća tražene podatke korisniku uz poruku „Spisak avionskih djelova sa preostalim resursima“. (IA)



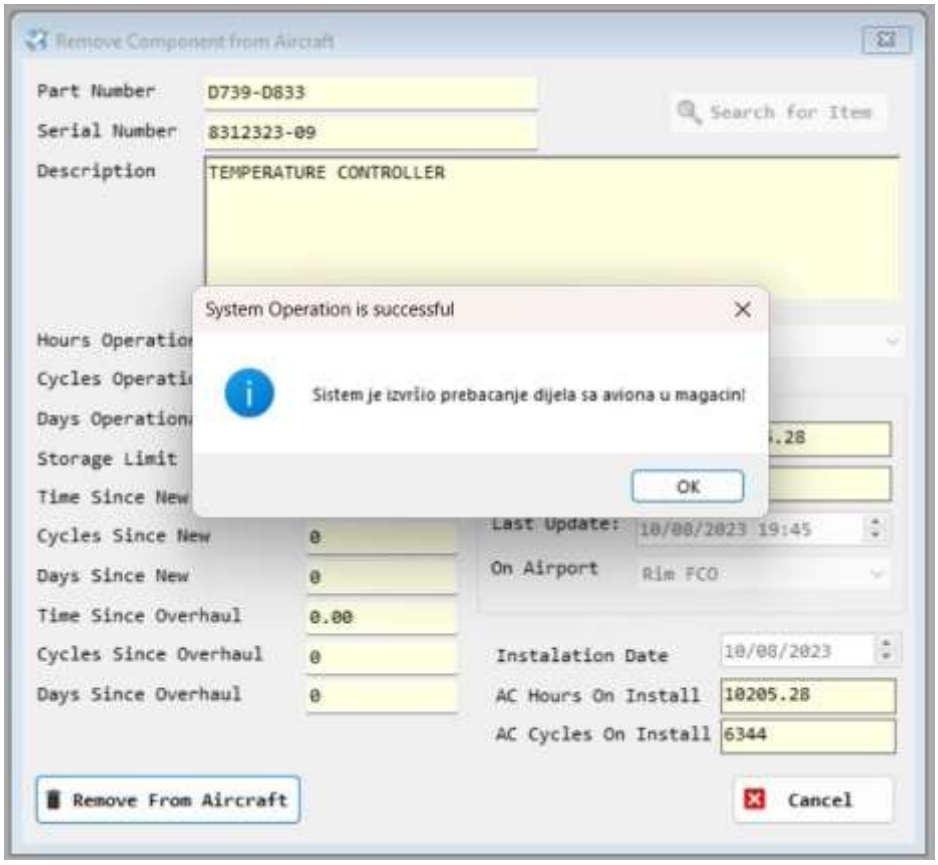
Opis akcije: Izborom aviona iz ComboBox kontrole aviomehaničar poziva sistemsku operaciju *VratiResurse(ResourceAvailability).*

5. Aviomehaničar bira PN (Part number) i SN (Serial number) avionskog dijela radi skidanja istog sa aviona i smještanja u magacin avionskih dijelova. (APUSO)
6. Aviomehaničar poziva sistem da na osnovu PN (Part number) i SN (Serial number) vrati podatke o dijelu instaliranom na avionu. (APSO)
7. Sistem pronalazi podatke relevantne za prebacivanje dijela sa aviona u magacin avionskih dijelova. (SO)
8. Sistem vraća podatke o dijelu instaliranom na avionu uz poruku „Sistem je spreman za prebacanje dijela sa aviona u magacin“. (IA)



Opis akcije: Duplim klikom na konkretan avio dio u DataGridView kontroli aviomehaničar poziva sistemsku operaciju *NadjiDioAvion(RotablePartsLog).*

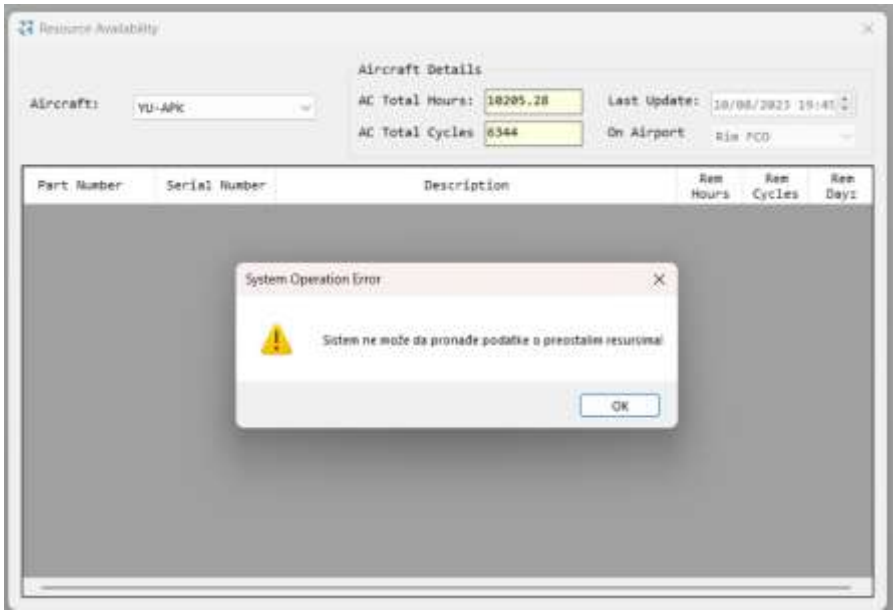
9. Aviomehaničar unosi relevantne podatke bitne za skidanje dijela sa aviona. (APUSO)
10. Aviomehaničar poziva sistem da prebaci dio sa aviona u magacin avio dijelova. (APSO)
11. Sistem prebaca dio sa aviona u magacin avionskih dijelova. (SO)
12. Sistem prikazuje podatke dijela u magacinu avionskih dijelova uz poruku „Sistem je izvršio prebacanje dijela sa aviona u magacin“. (IA)



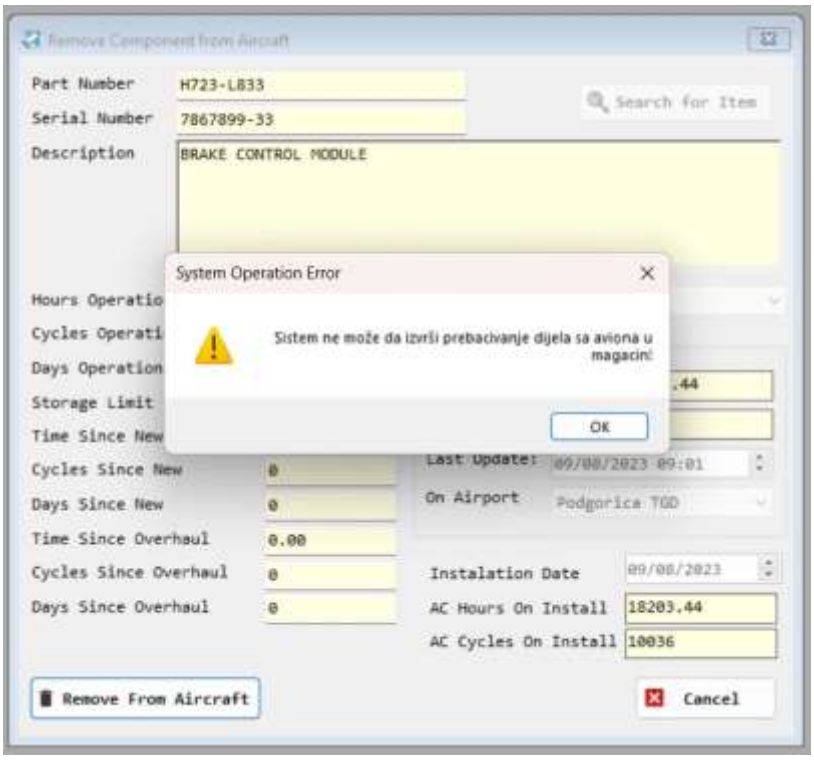
Opis akcije: Klikom na dugme „Remove From Aircraft“ aviomehaničar poziva sistemsku operaciju *PrebaciDioSaAvionaUMagacin(RotablePartsLog).*

Alternativna scenarija

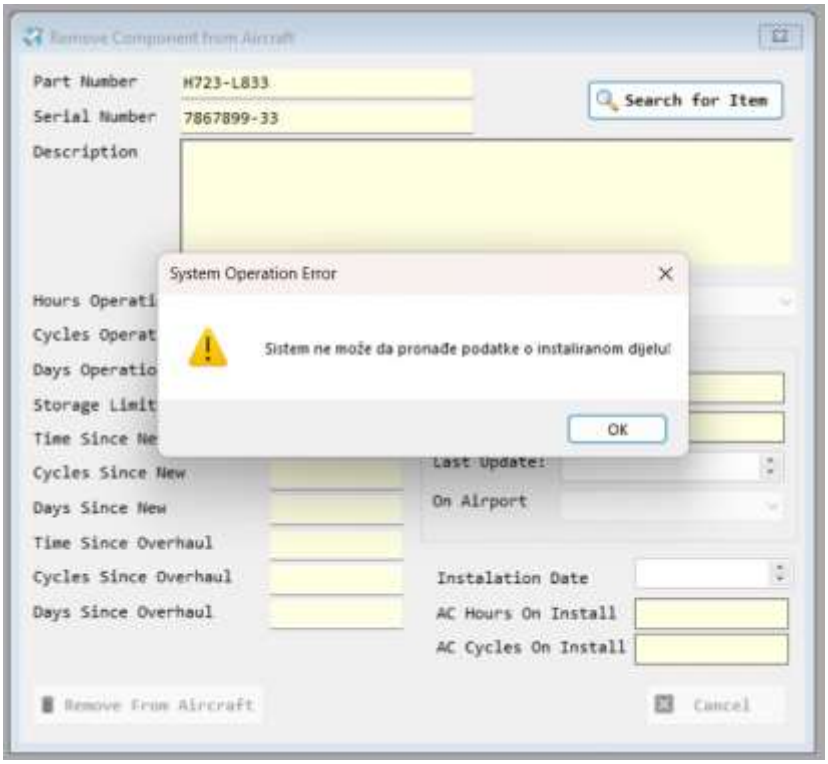
4.1 Ukoliko sistem ne može da nađe tražene podatke o instaliranim dijelovima na avionu on aviomehaničaru prikazuje poruku „Sistem ne može da pronađe podatke o preostalim resursima“. Prekida se izvršavanje scenarija. (IA)



8.1 Ukoliko sistem ne može da nađe tražene podatke o instaliranom dijelu na avionu on aviomehaničaru prikazuje poruku „Sistem ne može da pronađe podatke o instaliranom dijelu“. Prekida se izvršavanje scenarija. (IA)



12.1 Ukoliko sistem ne može da izvrši prebacivanje dijela sa aviona u magacin avionskih dijelova on aviomehaničaru prikazuje poruku „Sistem ne može da izvrši prebacivanje dijela sa aviona u magacin“.



SK10: Slučaj korišćenja – Slanje dijela iz magacina u avio servis

Naziv SK
Slanje dijela iz magacina u avio servis

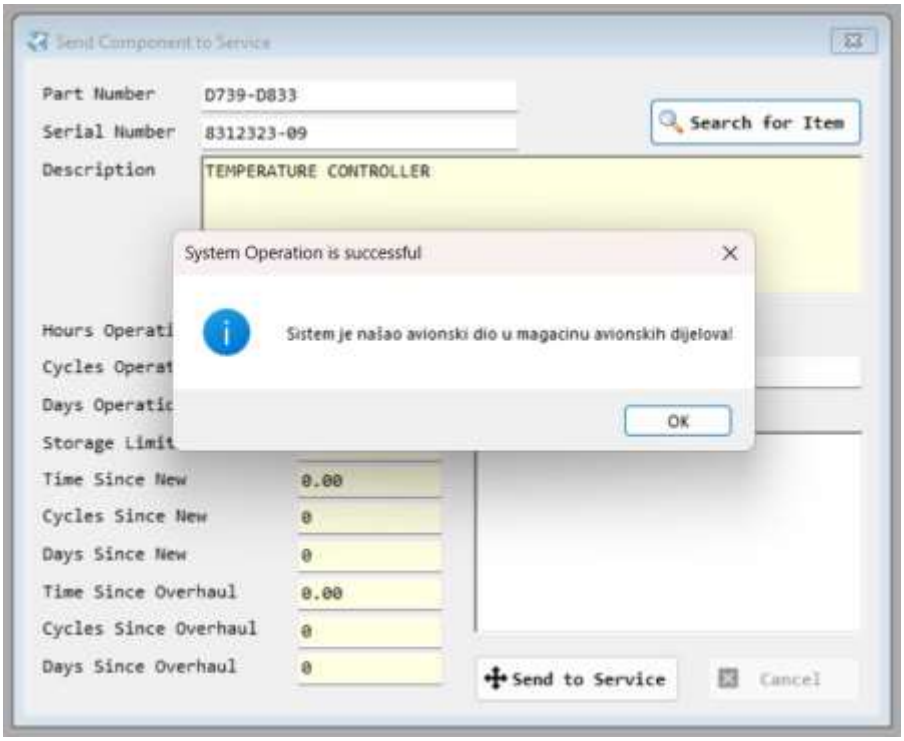
Aktori SK
Aviomehaničar

Učesnici SK
Aviomehaničar i sistem

Preduslov
Sistem je funkcionalan, aviomehaničar je ulogovan na sistem pod svojim nalogom, sistem prikazuje formu za unos podataka. Šifarnici aviona i aerodroma su inicijalizovani.

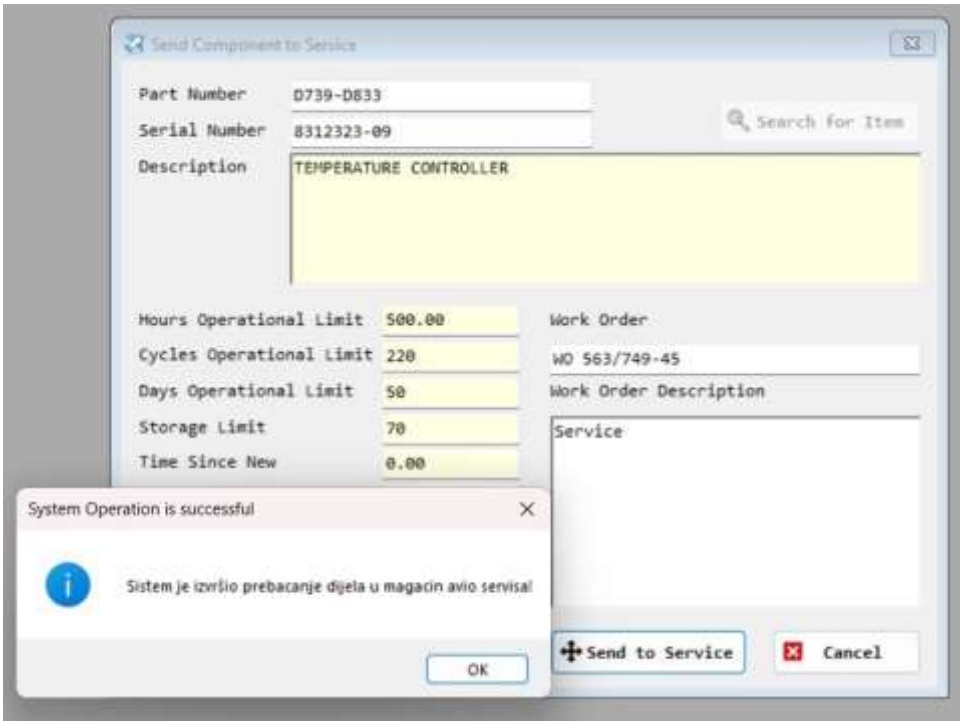
Osnovni scenario SK

1. Aviomehaničar unosi PN (Part number) i SN (Serial number) avionskog dijela radi pristupa kartonu avionskog dijela koji je lociran u magacinu. (APUSO)
2. Aviomehaničar poziva sistem da na osnovu PN (Part number) i SN (Serial number) kao parametara nađe avionski dio u magacinu avionskih dijelova. (APSO)
3. Sistem traži avionski dio u magacinu avionskih dijelova. (SO)
4. Sistem vraća podatke iz kartona avionskog dijela kao i poruku „Sistem je našao avionski dio u magacinu avionskih dijelova“. (IA)



Opis akcije: Klikom na dugme „Search for Item“ aviomehaničar poziva sistemsku operaciju *NadjiDioMagacin(RotablePartsLog).*

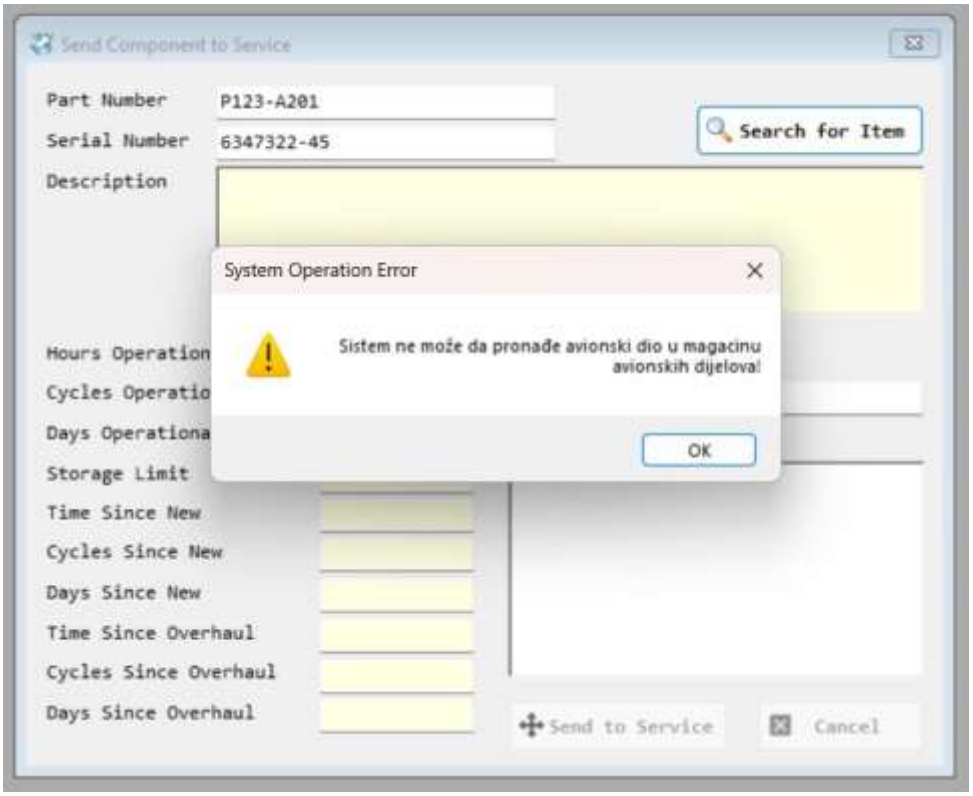
5. Aviomehaničar unosi podatke relevantne za prebacanje avionskog dijela iz magacina avionskih dijelova u magacin avio servisa. (APUSO)
6. Aviomehaničar poziva sistem da izabrani avionski dio prebaci iz magacina avionskih dijelova u magacin avio servisa. (APSO)
7. Sistem prebaca avionski dio iz magacina avionskih dijelova u magacin avio servisa. (SO)
8. Sistem prikazuje podatke o prebačenom dijelu u avio servis uz poruku „Sistem je izvršio prebacanje dijela u magacin avio servisa“. (IA)



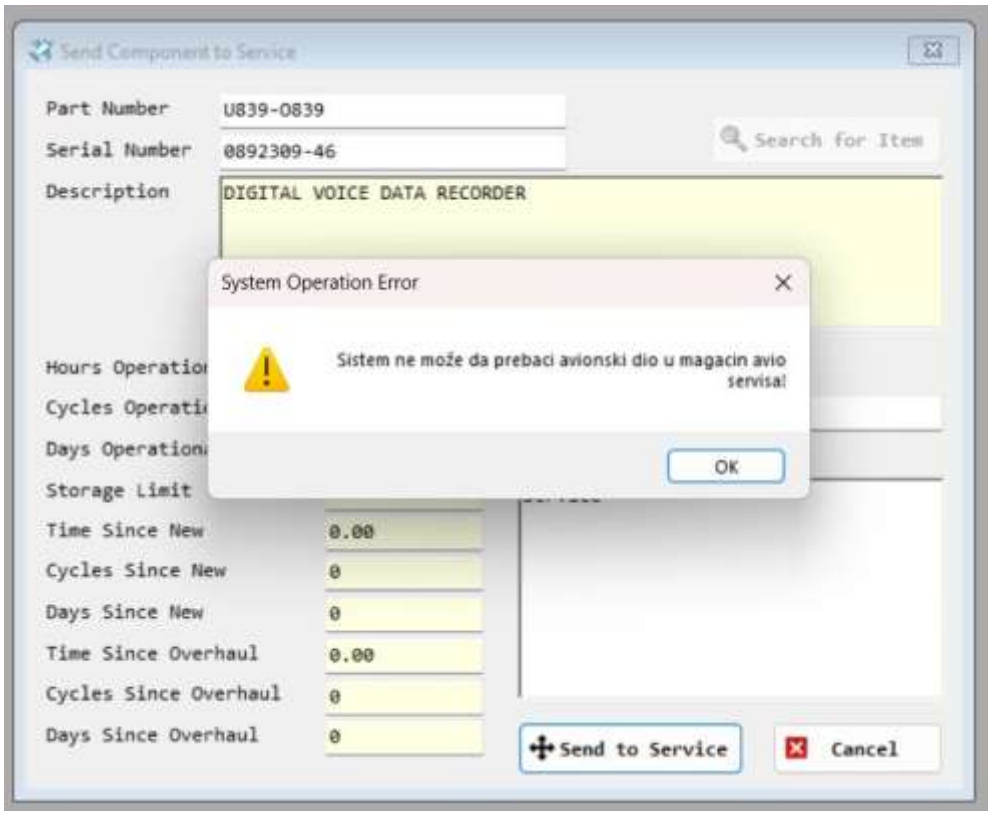
Opis akcije: Klikom na dugme „Send to Service“ aviomehaničar poziva sistemsku operaciju *PrebaciDioIzMagacinaUServis(RotablePartsLog).*

Alternativna scenarija

4.1 Ukoliko sistem ne može da nađe tražene podatke iz kartona avionskog dijela koji je lociran u magacinu on aviomehaničaru prikazuje poruku „Sistem ne može da pronađe avionski dio u magacinu avionskih dijelova“. Prekida se izvršavanje scenarija. (IA)



8.1 Ukoliko sistem ne može da izvršio prebacanje dijela iz magacina avionskih dijelova u magacin avio servisa on aviomehaničaru prikazuje poruku „Sistem ne može da prebaci avionski dio u magacin avio servisa“. (IA)



SK11: Slučaj korišćenja – Izvještavanje o preostalim resursima dijela aviona

Naziv SK
Izvještavanje o preostalim resursima dijela aviona

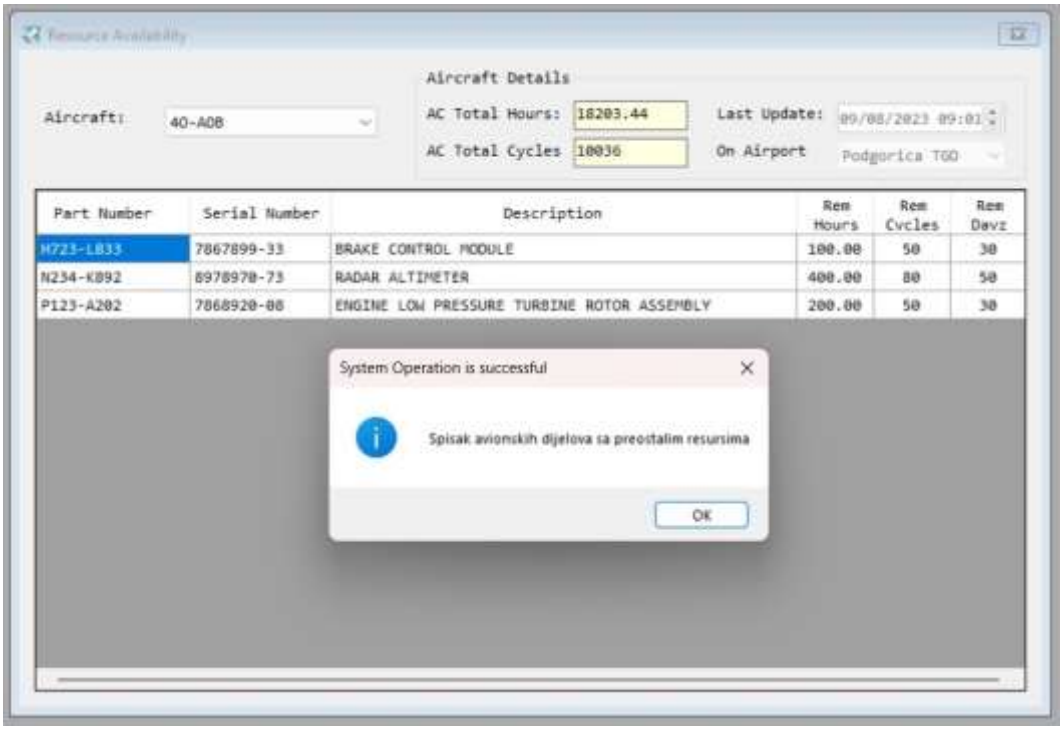
Aktori SK
Aviomehaničar

Učesnici SK
Aviomehaničar i sistem

Preduslov
Sistem je funkcionalan, aviomehaničar je ulogovan na sistem pod svojim nalogom, sistem prikazuje formu za unos podataka. Šifarnici aviona i aerodroma su inicijalizovani. Učitana lista aviona.

Osnovni scenario SK

1. Aviomehaničar bira avion radi izvještavanja o preostalim resursima avionskih dijelova instaliranim na njega. (APUSO)
2. Aviomehaničar poziva sistem da na osnovu izabranog aviona vrati spisak svih dijelova sa informacijom o preostaloj količini resursa. (APSO)
3. Sistem traži informacijom o preostaloj količini resursa avionskih dijelova instaliranih na avionu. (SO)
4. Sistem vraća tražene podatke korisnika uz poruku „Spisak avionskih dijelova sa preostalim resursima“. (IA)



Opis akcije: Izborom aviona iz ComboBox kontrole aviomehaničar poziva sistemsku operaciju *VratiResurse(ResourceAvailability).*

Alternativna scenarija

4.1 Ukoliko sistem ne može da nađe tražene podatke o preostalim resursima on aviomehaničaru prikazuje poruku „Sistem ne može da pronađe podatke o preostalim resursima“.

SK12: Slučaj korišćenja – Izvještavanje o isteku resursa avionskog dijela (resurs u satima)

Naziv SK
Izvještavanje o isteku resursa avionskog dijela (resurs u satima)

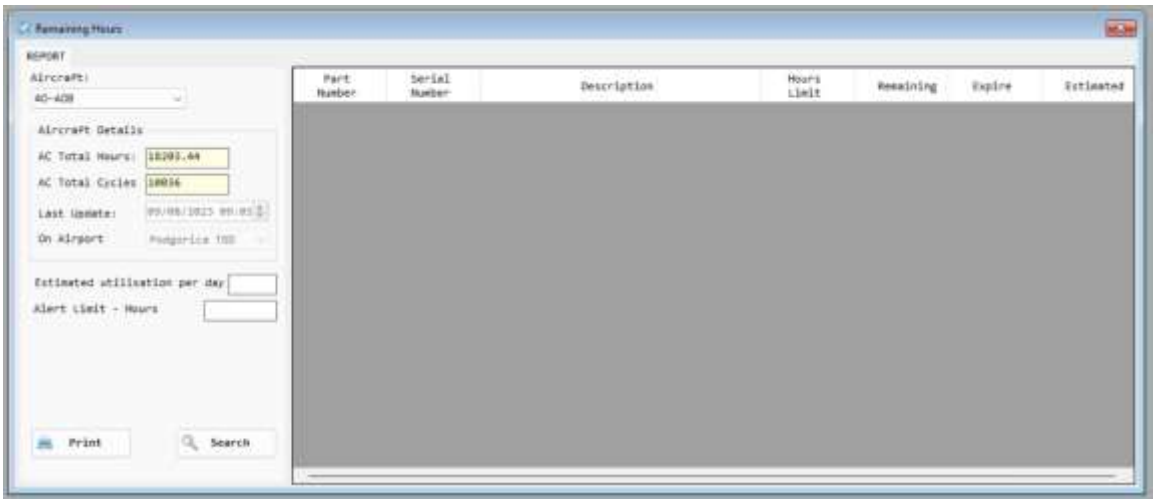
Aktori SK
Aviomehaničar

Učesnici SK
Aviomehaničar i sistem

Preduslov
Sistem je funkcionalan, aviomehaničar je prijavljen na sistem pod svojim nalogom, sistem prikazuje formu za unos podataka. Šifarnici aviona i aerodroma su inicijalizovani. Učitana je lista aviona.

Osnovni scenario SK

1. Aviomehaničar bira avion radi izvještavanja o isteku resursa avionskih dijelova instaliranih na njemu. (APUSO)
2. Aviomehaničar poziva sistem da na osnovu izabranog aviona vrati sve informacije o izabranom avionu. (APSO)
3. Sistem traži informacije o avionu. (SO)
4. Sistem vraća tražene podatke korisniku. (IA)



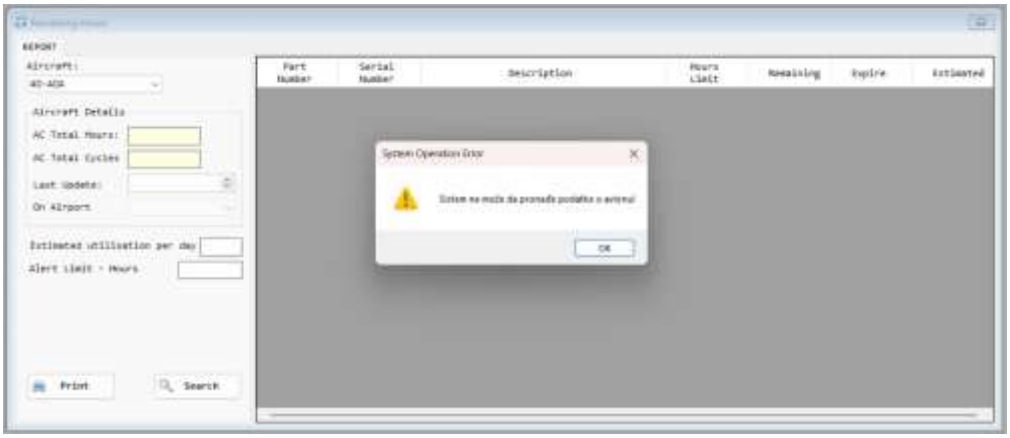
Opis akcije: Izborom aviona iz ComboBox kontrole aviomehaničar poziva sistemsku operaciju *UcitajAvion(Aircraft)*.

5. Aviomehaničar unosi informacije o dnevnom broju sati leta aviona, kao i broju sati za servis. (APUSO)
6. Aviomehaničar poziva sistem da na osnovu unešenih vrijednosti vrati sve dijelove aviona koji zadovoljavaju navedeni uslov. (APSO)
7. Sistem traži informacije o preostaloj količini resursa avionskih dijelova instaliranih na avionu. (SO)
8. Sistem vraća tražene podatke korisnika uz poruku „Sistem je našao sledeće dijelove! “. (IA)

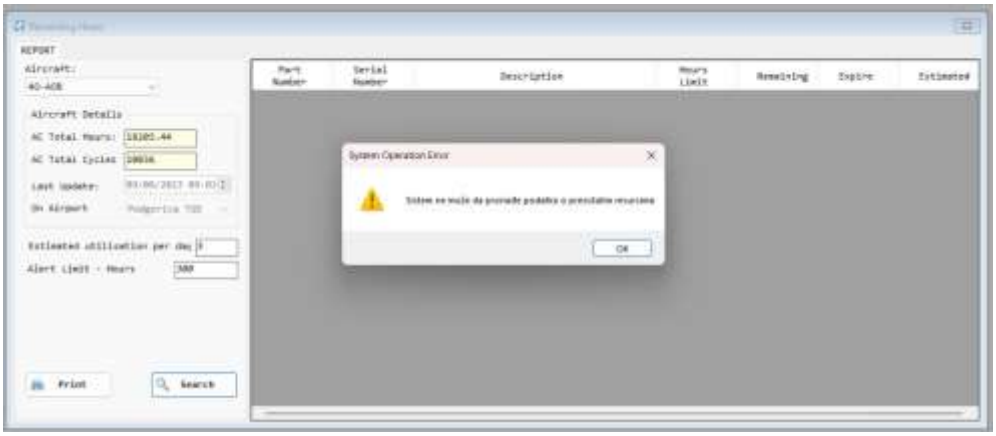
Opis akcije: Klikom na dugme „Search“ aviomehaničar poziva sistemsku operaciju *VratiPreostaleSate(RemainingHours).*

Alternativna scenarija

4.1 Ukoliko sistem ne može da nađe tražene podatke o avionu on aviomehaničaru prikazuje poruku „Sistem ne može da pronađe podatke o avionu.“. Prekida se izvršavanje scenarija. (IA)



8.1 Ukoliko sistem ne može da nađe tražene dijelove aviona on aviomehaničaru prikazuje poruku „Sistem ne može da pronađe podatke o preostalim resursima“. (IA)



SK13: Slučaj korišćenja – Izvještavanje o isteku resursa avionskog dijela (resurs u ciklusima)

Naziv SK
Izvještavanje o isteku resursa avionskog dijela (resurs u ciklusima)

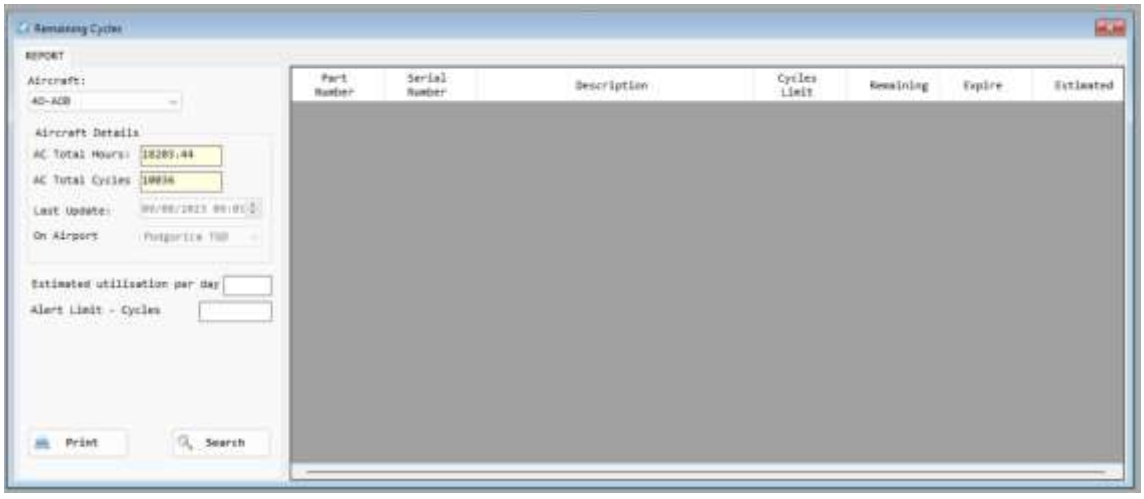
Aktori SK
Aviomehaničar

Učesnici SK
Aviomehaničar i sistem

Preduslov
Sistem je funkcionalan, aviomehaničar je prijavljen na sistem pod svojim nalogom, sistem prikazuje formu za unos podataka. Šifarnici aviona i aerodroma su inicijalizovani. Učitana je lista aviona.

Osnovni scenario SK

1. Aviomehaničar bira avion radi izvještavanja o isteku resursa avionskih dijelova instaliranih na njemu. (APUSO)
2. Aviomehaničar poziva sistem da na osnovu izabranog aviona vrati sve informacije o izabranom avionu. (APSO)
3. Sistem traži informacije o avionu. (SO)
4. Sistem vraća tražene podatke korisniku. (IA)



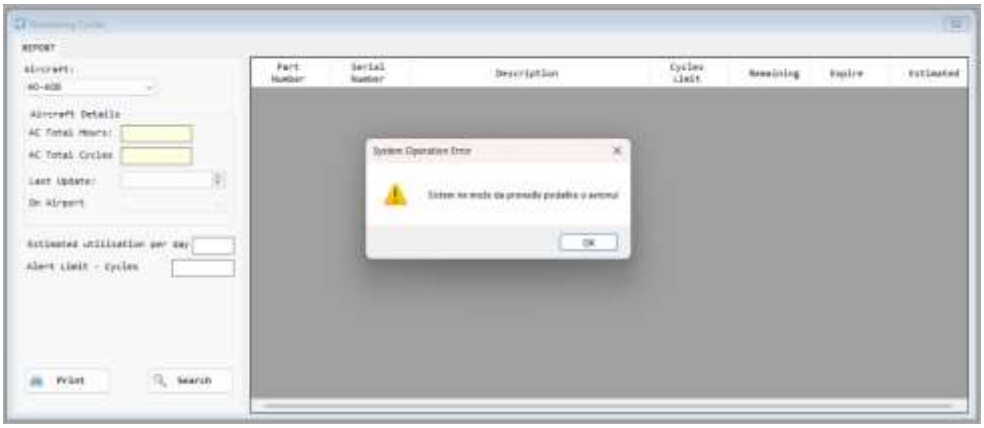
Opis akcije: Izborom aviona iz ComboBox kontrole aviomehaničar poziva sistemsku operaciju *UcitajAvion(Aircraft)*.

5. Aviomehaničar unosi informacije o dnevnom broju ciklusa leta aviona, kao i broju ciklusa za servis. (APUSO)
6. Aviomehaničar poziva sistem da na osnovu unešenih vrijednosti vrati sve dijelove aviona koji zadovoljavaju navedeni uslov. (APSO)
7. Sistem traži informacije o preostaloj količini resursa avionskih dijelova instaliranih na avionu. (SO)
8. Sistem vraća tražene podatke korisnika uz poruku „Sistem je našao sledeće dijelove!“. (IA)

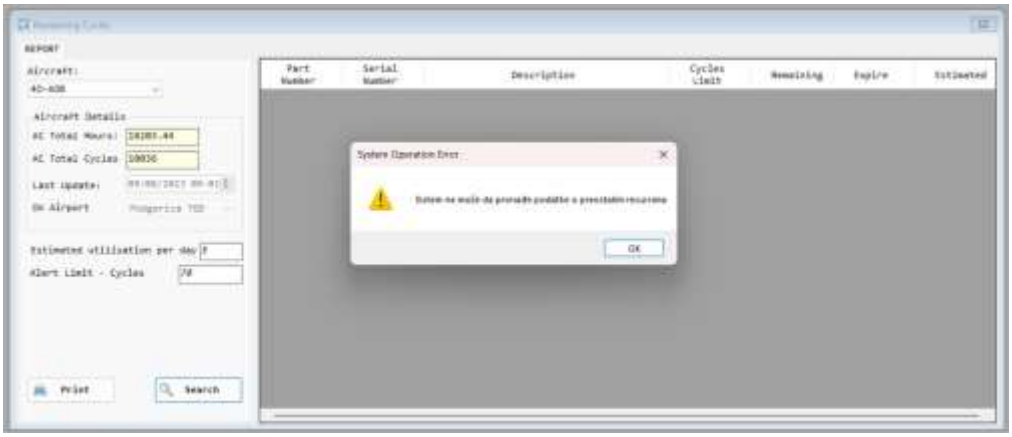
Opis akcije: Klikom na dugme „Search“ aviomehaničar poziva sistemsku operaciju *VratiPreostaleCikluse(RemainingCycles).*

Alternativna scenarija

4.1 Ukoliko sistem ne može da nađe tražene podatke o avionu on aviomehaničaru prikazuje poruku „Sistem ne može da pronađe podatke o avionu.“. Prekida se izvršavanje scenarija. (IA)



8.1 Ukoliko sistem ne može da nađe tražene dijelove aviona on aviomehaničaru prikazuje poruku „Sistem ne može da pronađe podatke o preostalim resursima“. (IA)



SK14: Slučaj korišćenja – Izvještavanje o isteku resursa avionskog dijela (resurs u danima)

Naziv SK
Izvještavanje o isteku resursa avionskog dijela (resurs u danima)

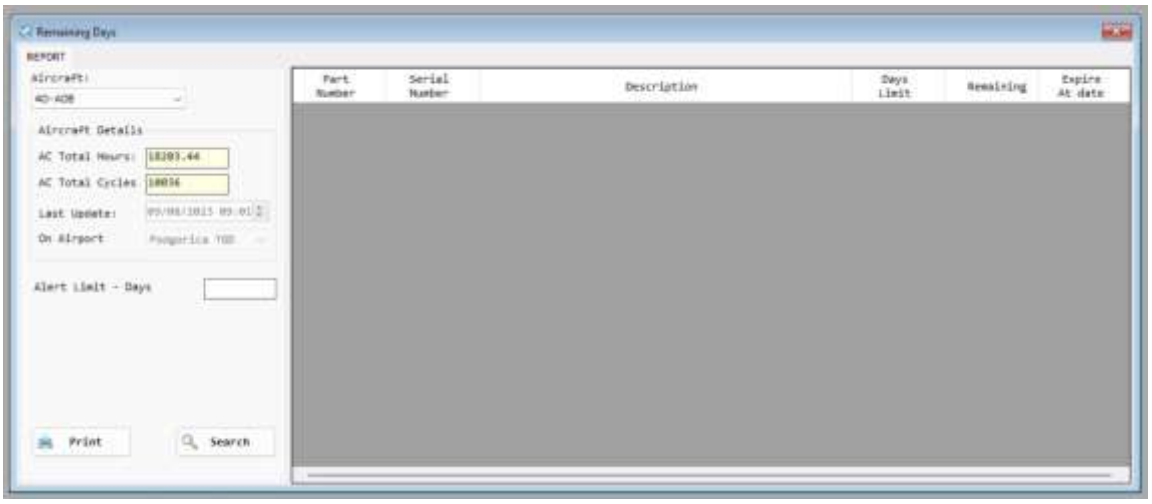
Aktori SK
Aviomehaničar

Učesnici SK
Aviomehaničar i sistem

Preduslov
Sistem je funkcionalan, aviomehaničar je prijavljen na sistem pod svojim nalogom, sistem prikazuje formu za unos podataka. Šifarnici aviona i aerodroma su inicijalizovani. Učitana je lista aviona.

Osnovni scenario SK

1. Aviomehaničar bira avion radi izvještavanja o isteku resursa avionskih dijelova instaliranih na njemu. (APUSO)
2. Aviomehaničar poziva sistem da na osnovu izabranog aviona vrati sve informacije o izabranom avionu. (APSO)
3. Sistem traži informacije o avionu. (SO)
4. Sistem vraća tražene podatke korisniku. (IA)



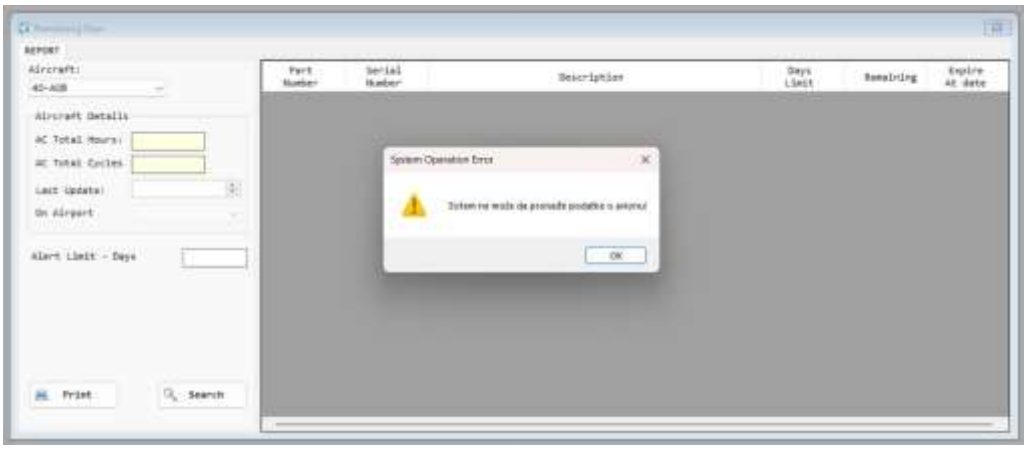
Opis akcije: Izborom aviona iz ComboBox kontrole aviomehaničar poziva sistemsku operaciju *UcitajAvion(Aircraft)*.

5. Aviomehaničar unosi informacije o broju dana za servis. (APUSO)
6. Aviomehaničar poziva sistem da na osnovu unešenih vrijednosti vrati sve dijelove aviona koji zadovoljavaju navedeni uslov. (APSO)
7. Sistem traži informacije o preostaloj količini resursa avionskih dijelova instaliranih na avionu. (SO)
8. Sistem vraća tražene podatke korisnika uz poruku „ Sistem je našao sledeće dijelove!“. (IA)

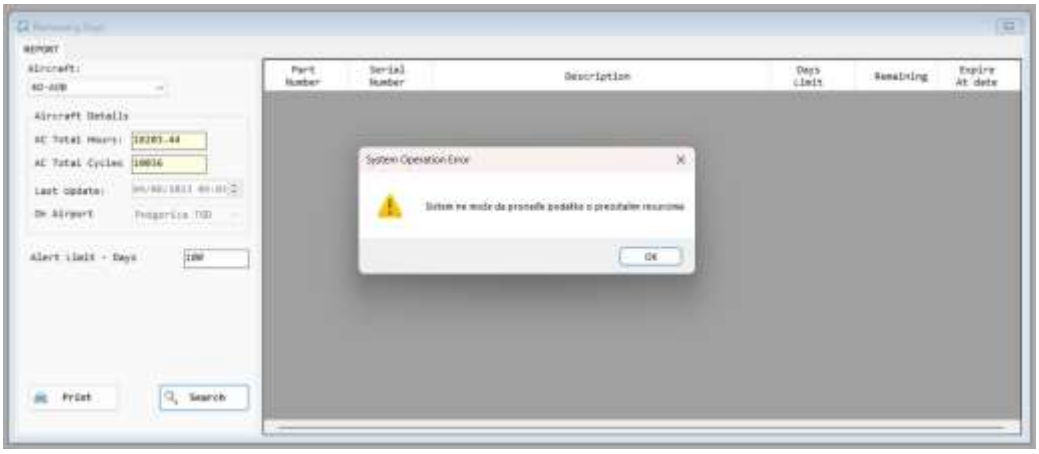
Opis akcije: Klikom na dugme „Search“ aviomehaničar poziva sistemsku operaciju *VratiPreostaleDane(RemainingDays).*

Alternativna scenarija

4.1 Ukoliko sistem ne može da nađe tražene podatke o avionu on aviomehaničaru prikazuje poruku „Sistem ne može da pronađe podatke o avionu.“. Prekida se izvršavanje scenarija. (IA)



8.1 Ukoliko sistem ne može da nađe tražene dijelove aviona on aviomehaničaru prikazuje poruku „Sistem ne može da pronađe podatke o preostalim resursima“. (IA)



SK15: Slučaj korišćenja – Servisiranje i produženje resursa dijela aviona

Naziv SK
Servisiranje i produženje resursa dijela aviona

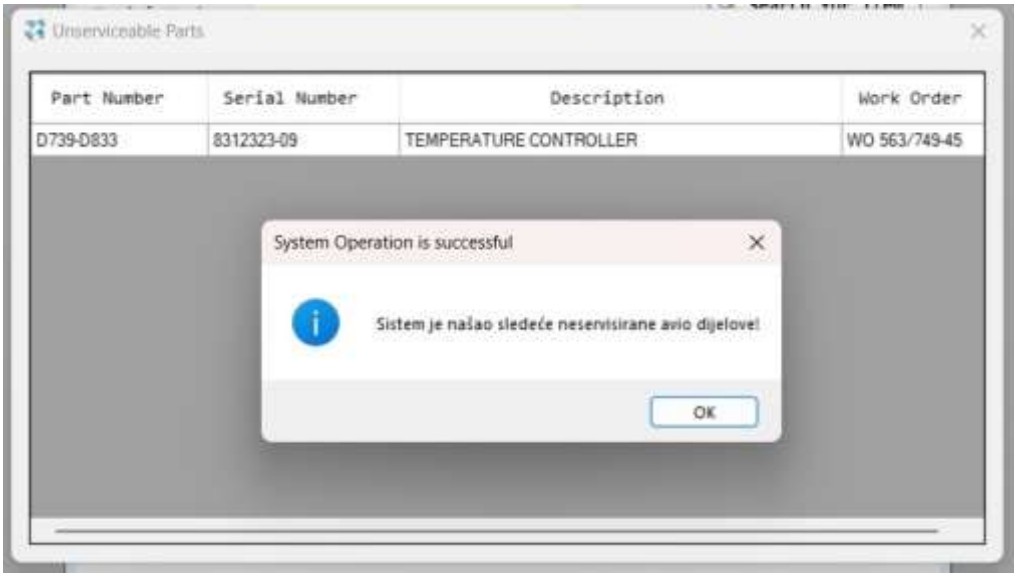
Aktori SK
Avio servis

Učesnici SK
Avio servis i sistem

Preduslov
Sistem je funkcionalan, avio servis je ulogovan na sistem pod svojim nalogom, sistem prikazuje formu za unos podataka.

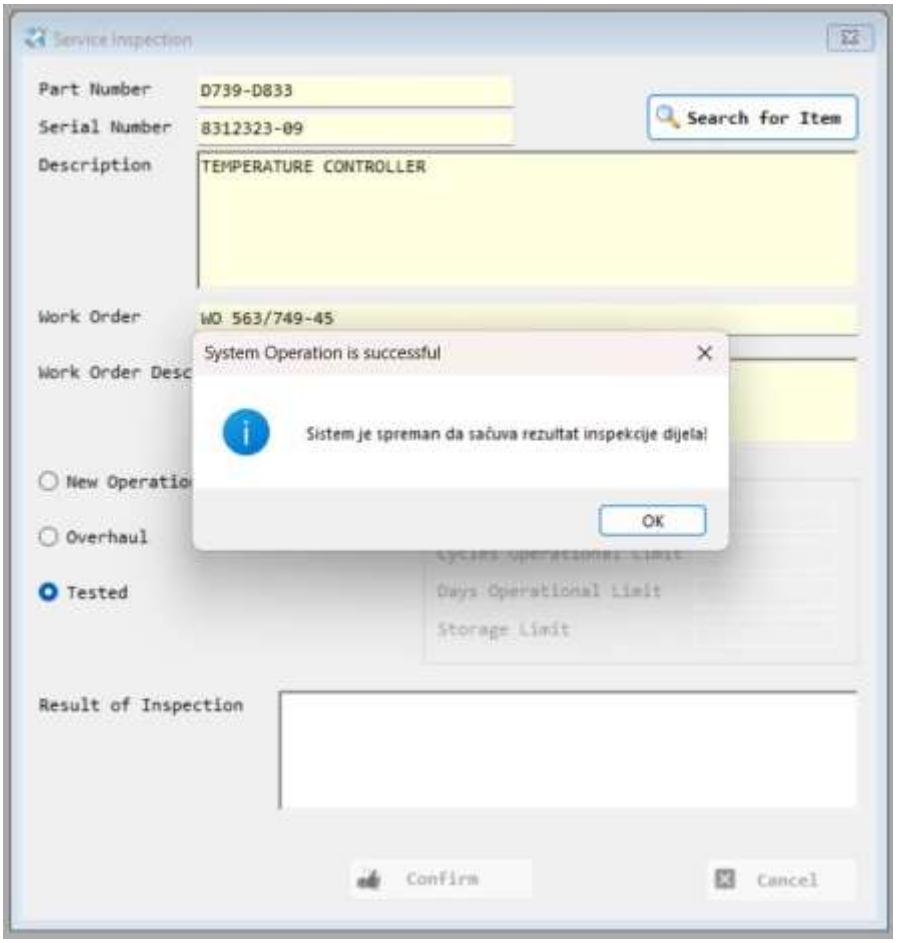
Osnovni scenario SK

1. Avio servis bira magacin neservisiranih dijelova radi testiranja i reparacije avionskog dijela. (APUSO)
2. Avio servis poziva sistem da iz magacin neservisiranih dijelova vrati sve neservisirane dijelove radi testiranja i reparacije avionskog dijela. (APSO)
3. Sistem pravi spisak svih neservisiranih dijelova aviona. (SO)
4. Sistem vraća spisak neservisiranih dijelova aviona uz poruku „Sistem je našao sledeće neservisirane avio dijelove“. (IA)



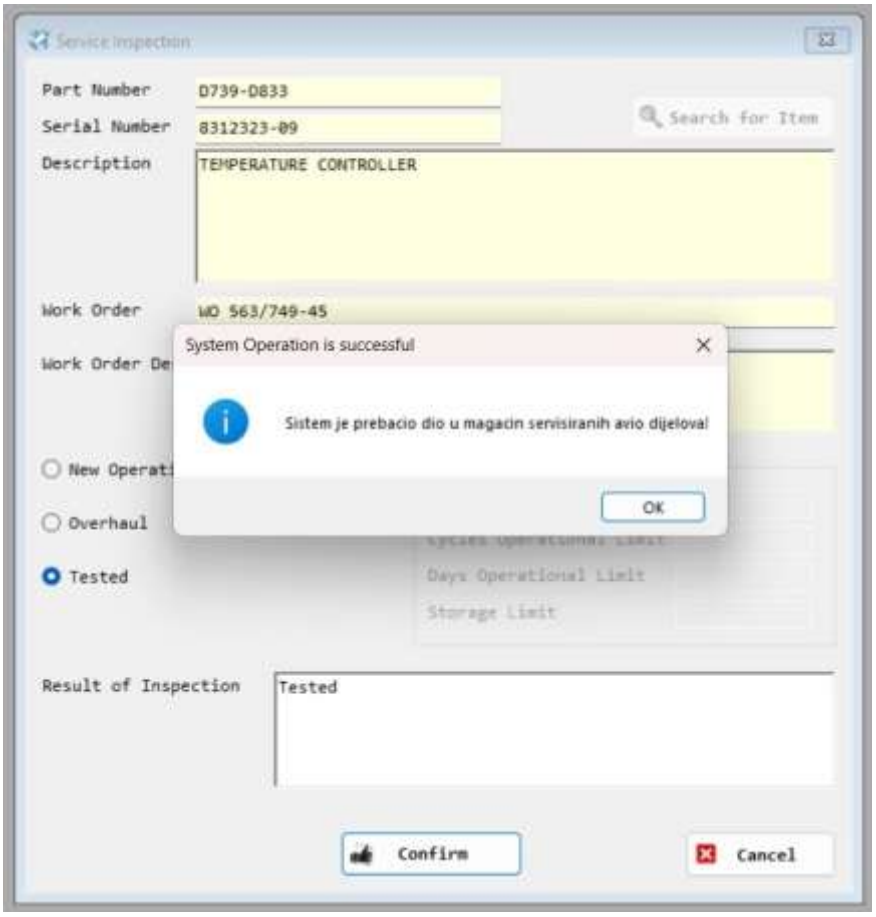
Opis akcije: Klikom na dugme „Search for Item“ avio servis poziva sistemsku operaciju *NadjiNeservisirane(UnserviceableParts).*

5. Avio servis bira PN (Part number) i SN (Serial number) avionskog dijela radi testiranja i reparacije istog. (APUSO)
6. Avio servis poziva sistem da vrati informacije o selektovanom dijelu iz liste neservisiranih djelova. (APSO)
7. Sistem izvršava akciju selektovanja dijelu iz liste neservisiranih djelova. (SO)
8. Sistem vraća Avio servisu poruku „Sistem je spreman da sačuva rezultat inspekcije dijela“. (IA)



Opis akcije: Duplim klikom na konkretan avionski dio u DataGridView kontroli avio servis poziva sistemsku operaciju *NeservisiraniDio(RotablePartsLog).*

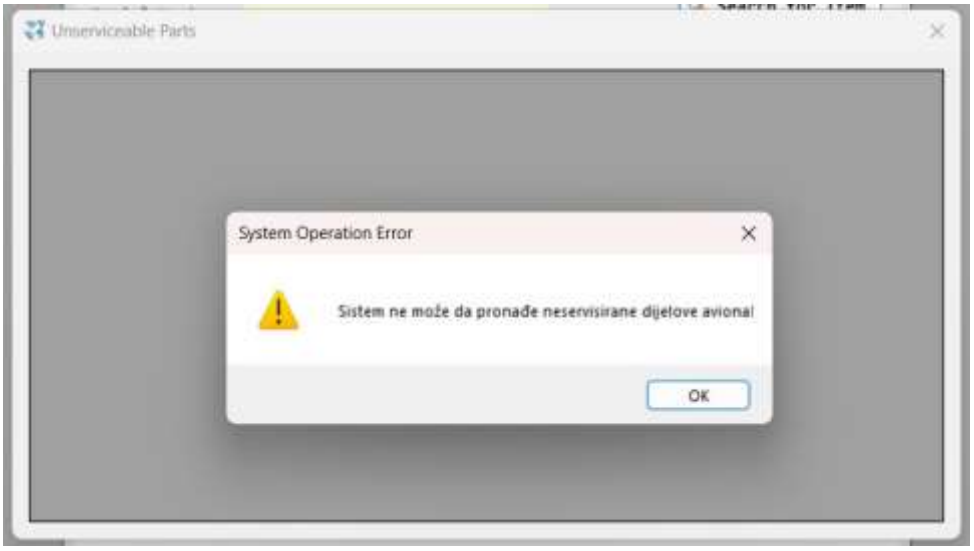
9. Nakon izvršenih testiranja avio servis izdaje sertifikat o produženju resursa avionskom dijelu ili potvrdu da se dio ne može servisirati. (APUSO)
10. Avio servis poziva sistem da podatke vezane za inspekciju avionskog dijela sačuva i prebaci dio u magacin servisiranih dijelova. (APSO)
11. Sistem izvršava akciju prebacanja dijela u magacin servisiranih dijelova. (SO)
12. Sistem prikazuje podatke o servisiranom dijelu uz „Sistem je prebacio dio u magacin servisiranih avio dijelova“. (IA)



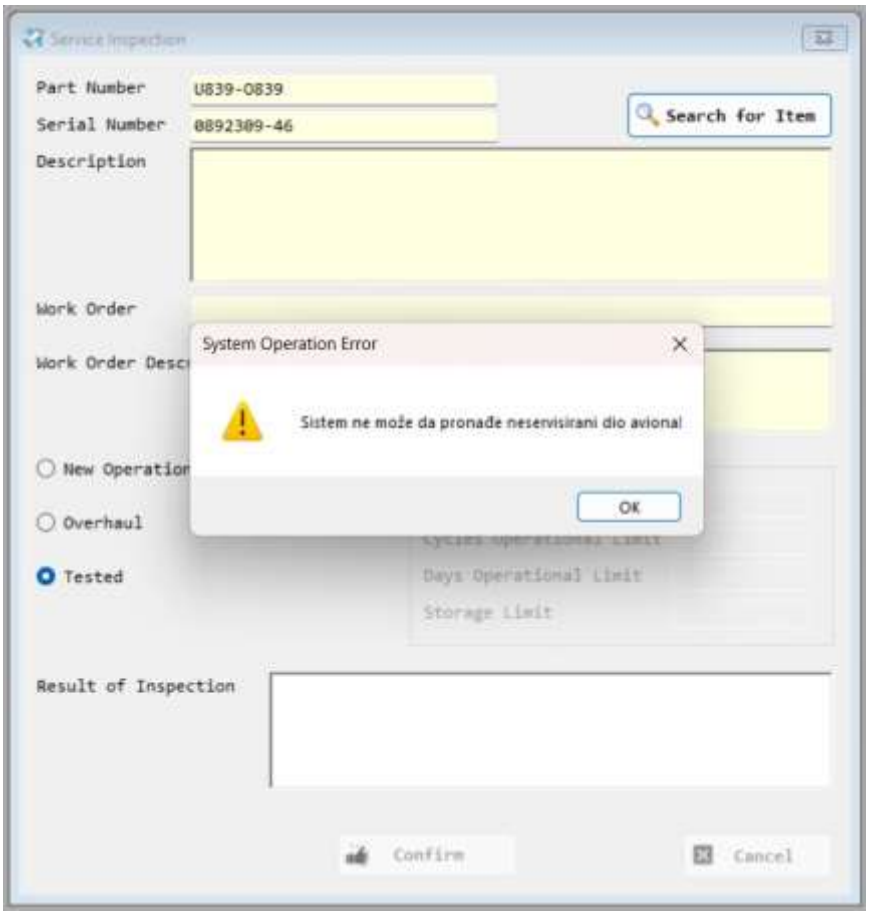
Opis akcije: Klikom na dugme „Confirm“ avio servis poziva sistemsku operaciju *RezultatInspekcije(RotablePartsService).*

Alternativna scenarija

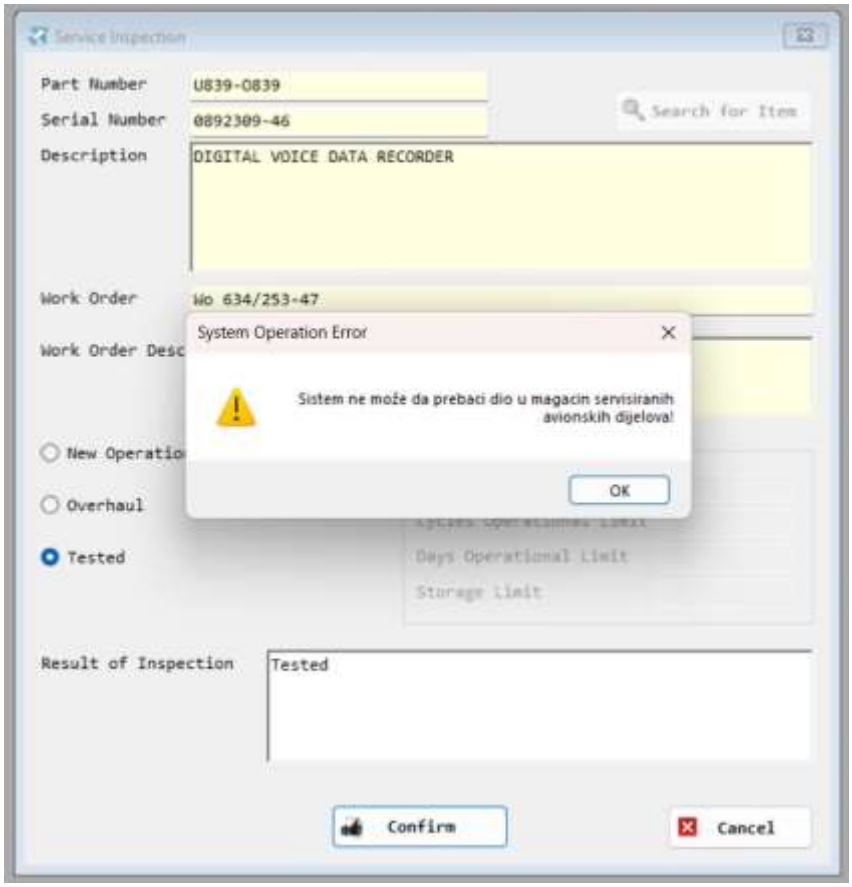
4.1 Ukoliko sistem ne može da nađe neservisirane dijelove aviona on avio servisu prikazuje poruku „Sistem ne može da pronađe neservisirane dijelove aviona“. Prekida se izvršavanje scenarija. (IA)



8.1 Ukoliko sistem ne može da nađe neservisirani dio aviona on avio servisu prikazuje poruku „Sistem ne može da pronađe neservisirani dio aviona“. Prekida se izvršavanje scenarija. (IA)



12.1 Ukoliko sistem ne može da zapamti podatke o servisiranju i prebacanju dijela u magacin servisiranih dijelova on avio servisu prikazuje poruku „Sistem ne može da prebaci dio u magacin servisiranih avionskih dijelova“.


Service Inspection
Part Number
U839-0839
Serial Number
0892309-46
Search for Item
Description
DIGITAL VOICE DATA RECORDER
Work Order
Wo 634/253-47
Work Order Desc
System Operation Error
Sistem ne može da prebaci dio u magacin servisiranih avionskih dijelova!
OK
New Operatio
Overhaul
Tested
Days Operational Limit
Storage Limit
Result of Inspection
Tested
Confirm
Cancel

SK16: Slučaj korišćenja – Vraćanje dijela nakon popravke u magacin dijelova

Naziv SK
Vraćanje dijela aviona nakon reparacije u magacin dijelova

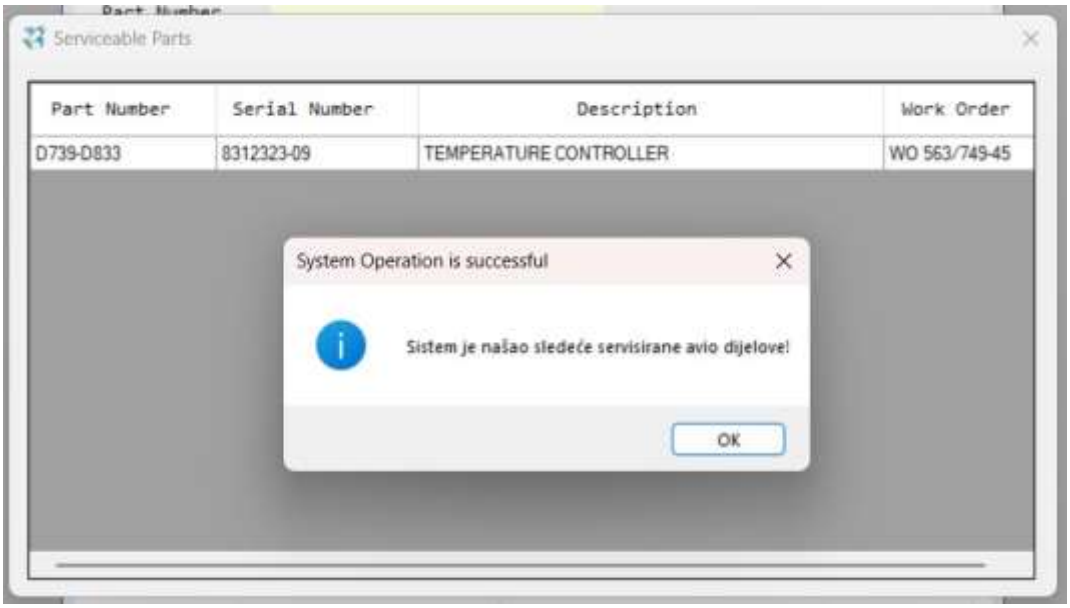
Aktori SK
Avio servis

Učesnici SK
Avio servis i sistem

Preduslov
Sistem je funkcionalan, avio servis je ulogovan na sistem pod svojim nalogom, sistem prikazuje formu za unos podataka.

Osnovni scenario SK

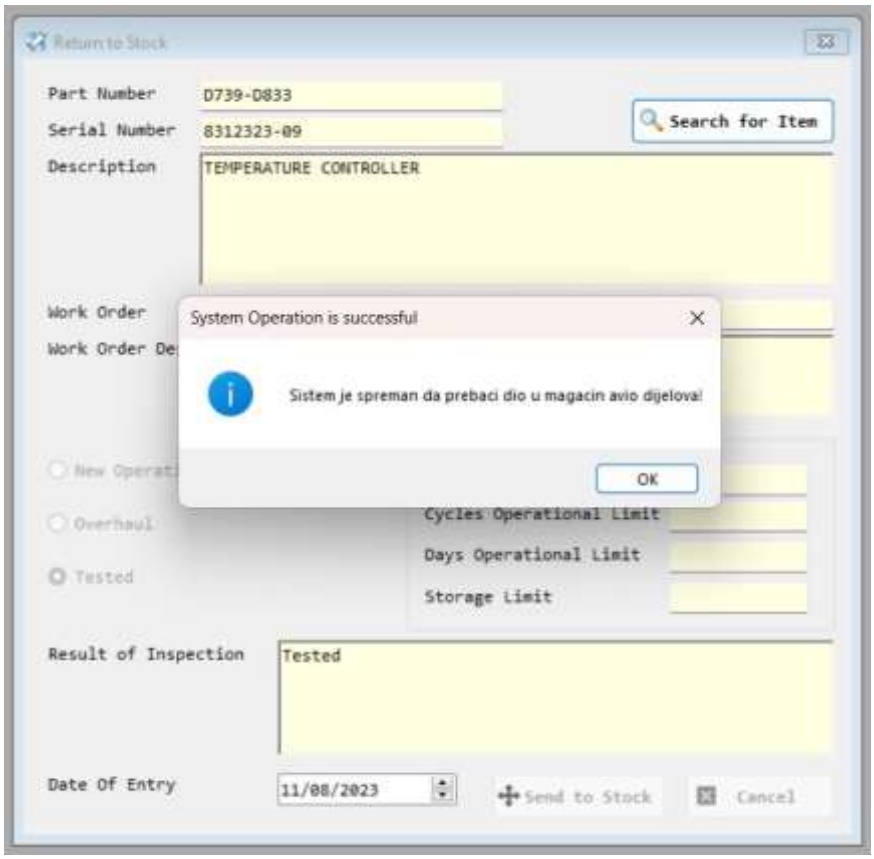
1. Avio servis bira magacin servisiranih dijelova radi vraćanja avionskog dijela u magacin pošiljaoca. (APUSO)
2. Avio servis poziva sistem da iz magacin servisiranih dijelova vrati sve servisirane dijelove radi vraćanja avionskog dijela u magacin pošiljaoca. (APSO)
3. Sistem pravi spisak svih servisiranih dijelova aviona. (SO)
4. Sistem vraća spisak servisiranih dijelova aviona uz poruku „Sistem je našao sledeće servisirane avio dijelove“. (IA)



Opis akcije: Klikom na dugme „Search for Item“ avio servis poziva sistemsku operaciju *NadjiServisirane(ServiceableParts).*

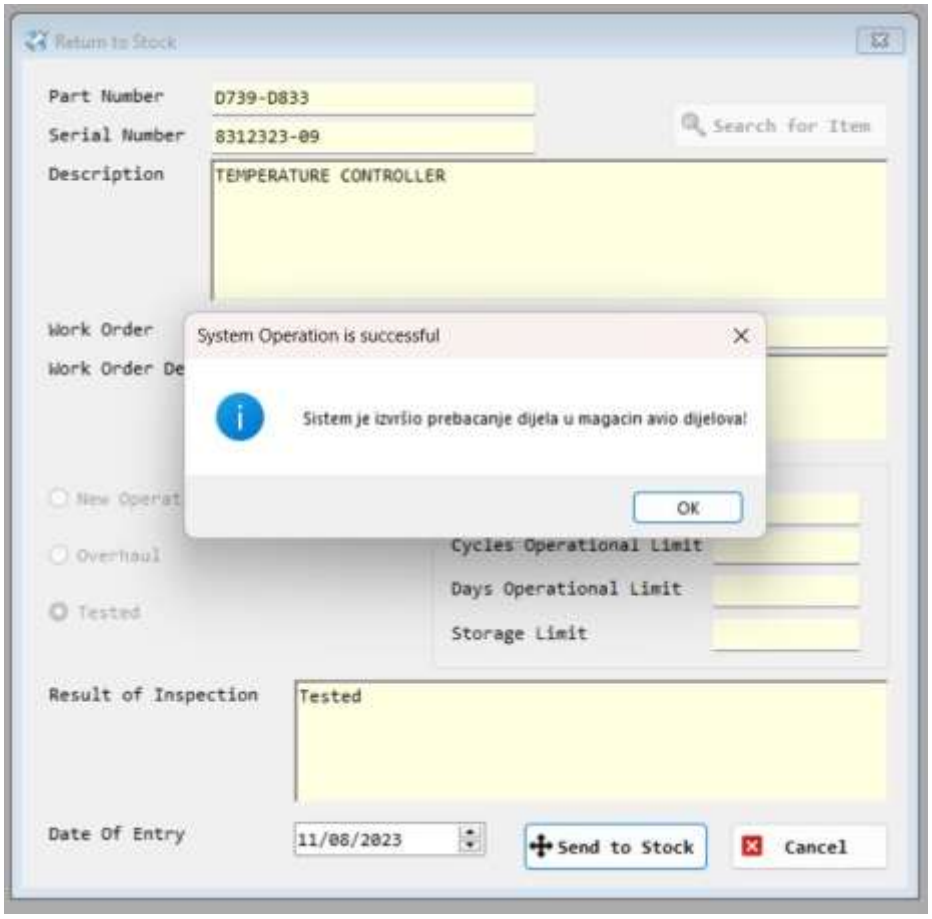
5. Avio servis bira PN (Part number) i SN (Serial number) avionskog dijela radi slanja istog u magacin avionskih dijelova. (APUSO)

6. Avio servis poziva sistem da na osnovu PN (Part number) i SN (Serial number) kao parametara vrati informacije o selektovanom dijelu iz liste servisiranih djelova. (APSO)
7. Sistem izvršava akciju selektovanja dijelu iz liste servisiranih djelova. (SO)
8. Sistem vraća avio servisu poruku „Sistem je spreman da prebaci dio u magacin avio dijelova“. (IA)



Opis akcije: Duplim klikom na konkretan avionski dio u DataGridView kontroli avio servis poziva sistemsku operaciju *ServisiraniDio(RotablePartsLog)*.

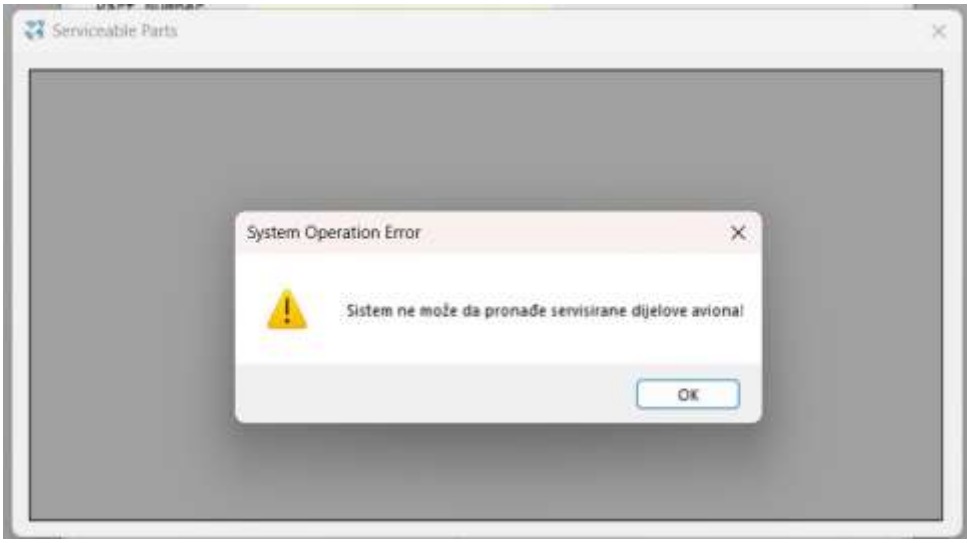
9. Avio servis unosi relevantne podatke bitne za prebacanje avionskog dijela iz magacina avio servisa u magacin avio djelova. (APUSO)
10. Avio servis poziva sistem da prebaci dio u magacin avionskih dijelova. (APSO)
11. Sistem izvršava akciju prebacanja dijela u magacin avionskih dijelova. (SO)
12. Sistem prikazuje podatke o dijelu koji se vraća u magacin avio dijelova uz poruku „Sistem je izvršio prebacanje dijela u magacin avio dijelova“. (IA)



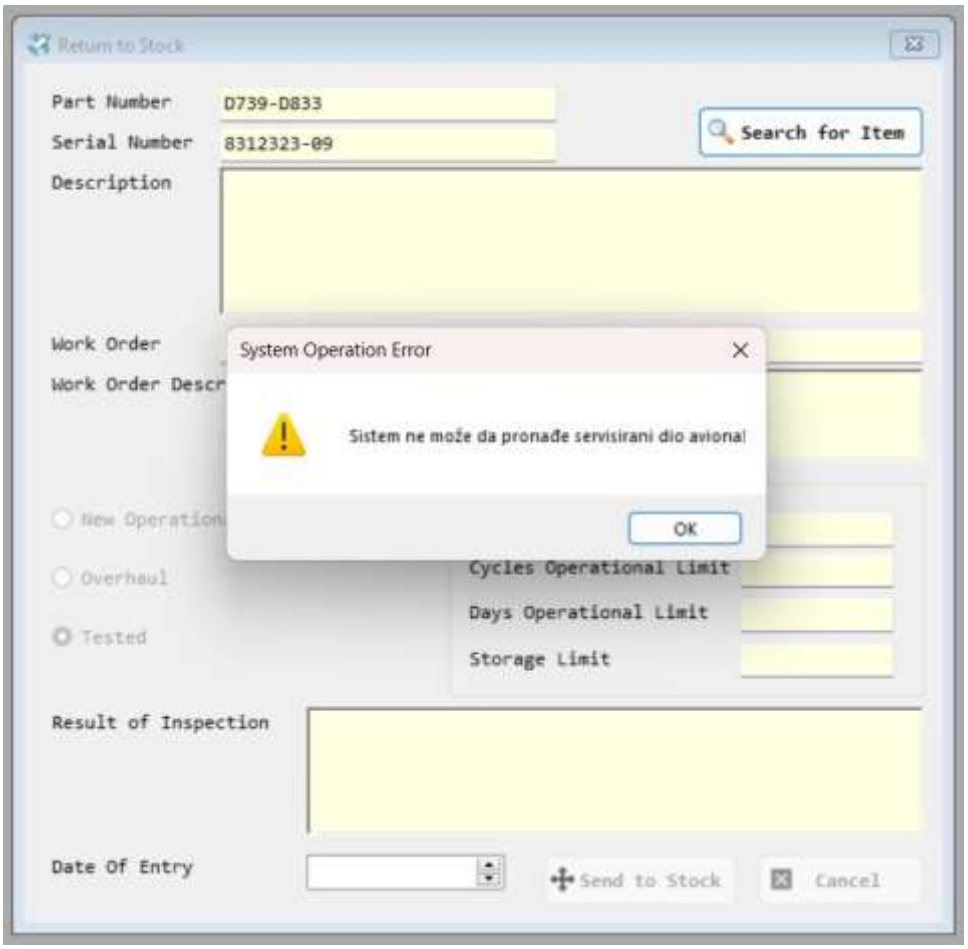
Opis akcije: Klikom na dugme „Send to Stock“ avio servis poziva sistemsku operaciju *PrebaciDioIzServisaUMagacin(RotablePartsLog).*

Alternativna scenarija

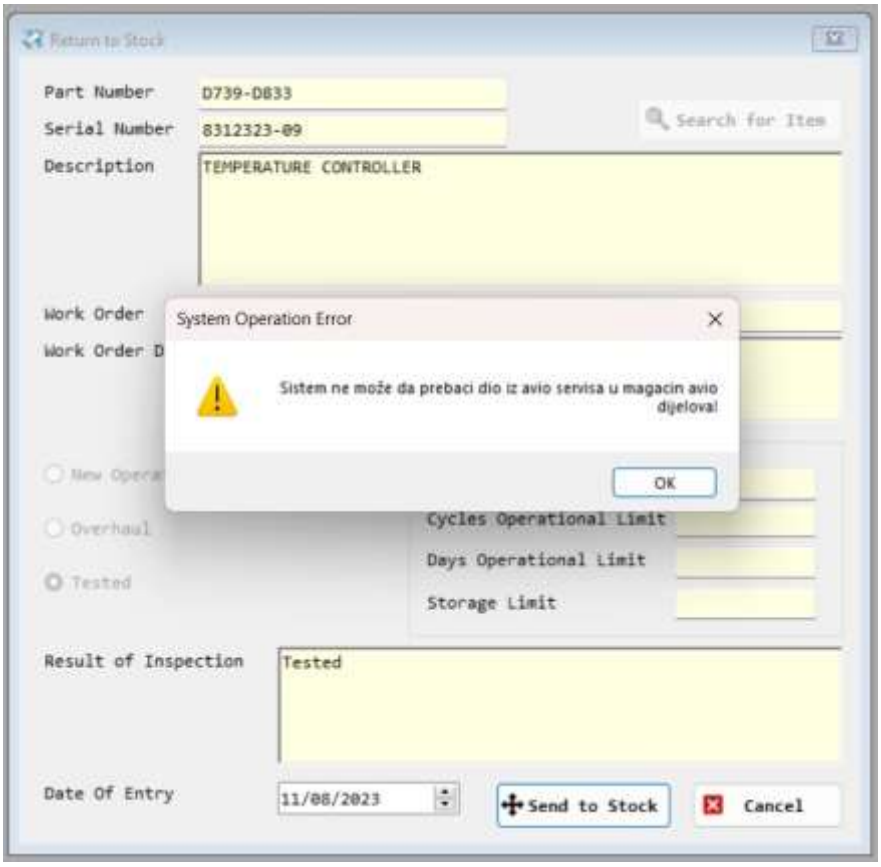
4.1 Ukoliko sistem ne može da nađe servisirane dijelove avionu on avio servisu prikazuje poruku „Sistem ne može da pronađe servisirane dijelove aviona“. Prekida se izvršavanje scenarija. (IA)



8.1 Ukoliko sistem ne može da nađe servisirani dio avionu on avio servisu prikazuje poruku „Sistem ne može da pronađe servisirani dio aviona“. Prekida se izvršavanje scenarija. (IA)



12.1 Ukoliko sistem ne može da izvrši prebacivanje dijela iz avio servisa u magacin avionskih dijelova on avio servisu prikazuje poruku „Sistem ne može da prebaci dio iz avio servisa u magacin avio dijelova“.


Return to Stock
Part Number
D739-D833
Serial Number
8312323-09
Search for Item
Description
TEMPERATURE CONTROLLER
Work Order
Work Order D
System Operation Error
Sistem ne može da prebaci dio iz avio servisa u magacin avio dijelova!
OK
New Opera
Overhaul
Tested
Cycles Operational Limit
Days Operational Limit
Storage Limit
Result of Inspection
Tested
Date Of Entry
11/08/2023
Send to Stock
Cancel

SK17: Slučaj korišćenja – Prikaz kretanja dijela kroz system

Naziv SK
Prikaz kretanja dijela kroz sistem

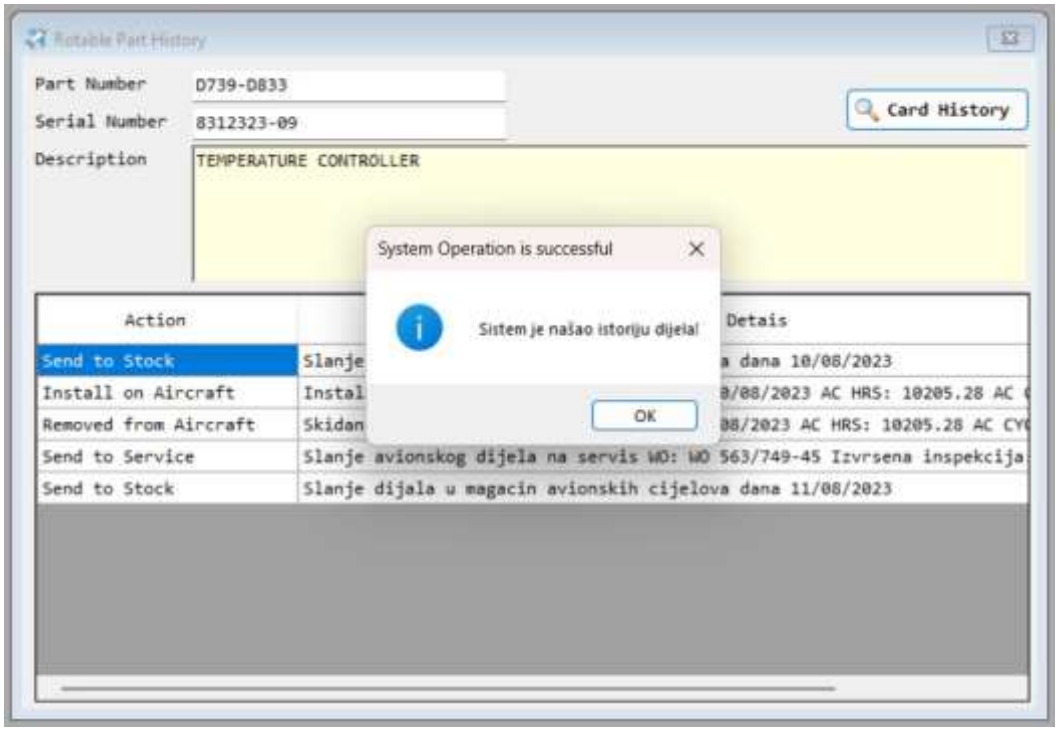
Aktori SK
Aviomehaničar

Učesnici SK
Aviomehaničar i sistem

Preduslov
Sistem je funkcionalan, aviomehaničar je prijavljen na sistem pod svojim nalogom, sistem prikazuje formu za unos podataka. Šifarnici aviona i aerodroma su inicijalizovani.

Osnovni scenario SK

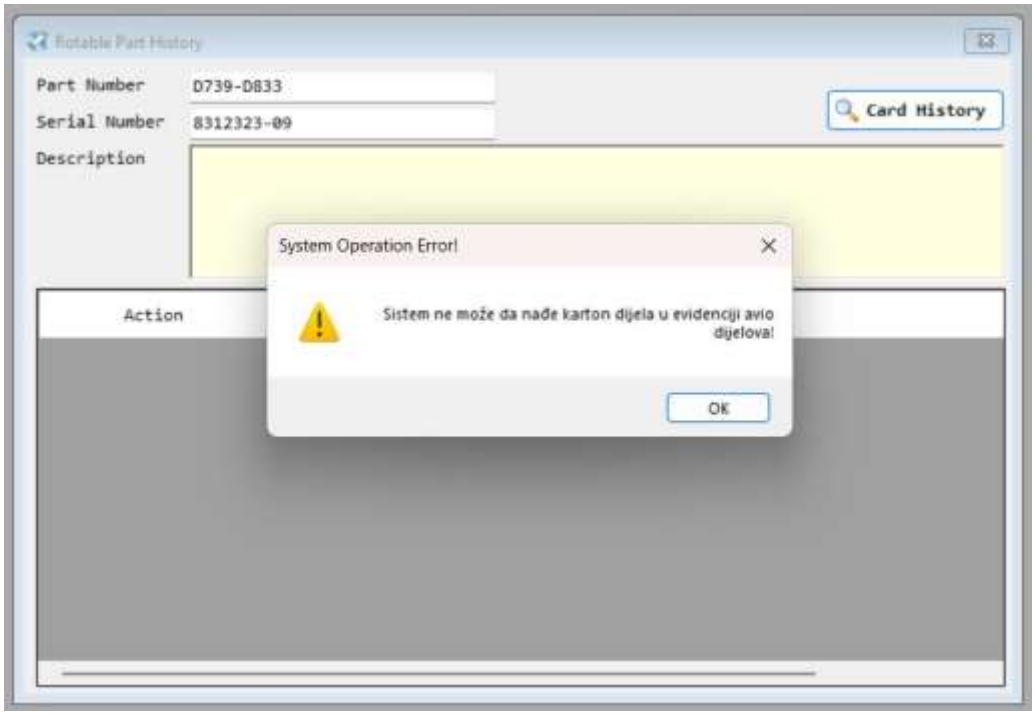
1. Aviomehaničar unosi PN (Part number) i SN (Serial number) avionskog dijela radi pristupa kartonu avionskog dijela. (APUSO)
2. Aviomehaničar poziva sistem da na osnovu PN (Part number) i SN (Serial number) kao parametara nađe karton kretanja avionskog dijela iz evidencije avionskih dijelova. (APSO)
3. Sistem traži karton kretanja avionskog dijela po zadatom kriterijumu. (SO)
4. Sistem vraća podatke kretanja avionskog dijela kao i poruku „Sistem je našao istoriju dijela!“. (IA)



Opis akcije: Klikom na dugme „Card History“ aviomehaničar poziva sistemsku operaciju *VratiIstorijuDijela(RotablePartsHistory)*

Alternativna scenarija

4.1 Ukoliko sistem ne može da nađe tražene podatke o avionskom dijelu sistem šalje aviomehaničaru poruku „Sistem ne može da nađe karton dijela u evidenciji avio dijelova!“. (IA)



4.3.1.2 Projektovanje kontrolera korisničkog interfejsa

Kontroler korisničkog interfejsa je odgovoran da:

1. prihvati podatke koje šalje ekranska forma
2. konvertuje podatke (koji se nalaze u grafičkim elementima) u objekat koji predstavlja ulazni argument SO koja će biti pozvana
3. šalje zahtev za izvršenje sistemske operacije do aplikacionog servera (softverskog sistema)
4. prihvata objekat (izlaz) softverskog sistema nastao kao rezultat izvršenja sistemske operacije
5. konvertuje objekat u podatke grafičkih elemenata

4.3.2 Projektovanje aplikacione logike

Aplikaciona logika opisuje strukturu i ponašanje sistema. Aplikacioni server se sastoji iz:

1. Kontrolera aplikacione logike – treba da podigne serverski soket koji će da osluškuje mrežu. Služi za komunikaciju sa klijentom i odgovoran je da prihvati zahtev za izvršenje sistemske operacije od klijenta i prosledi ga do poslovne logike koja je odgovorna za izvršenje SO
2. Poslovna logika – opisana je strukturom (domenske klase) i ponašanjem (sistemske operacije)
3. Broker baze podataka – služi za komunikaciju između poslovne logike i baze podataka

4.3.2.1 Kontroler aplikacione logike

Dio za komunikaciju podiže serverski soket koji osluškuje mrežu. Kada klijentski soket uspostavi konekciju sa serverskim soketom, tada server generiše nit koja će uspostaviti dvosmernu komunikaciju sa klijentom.

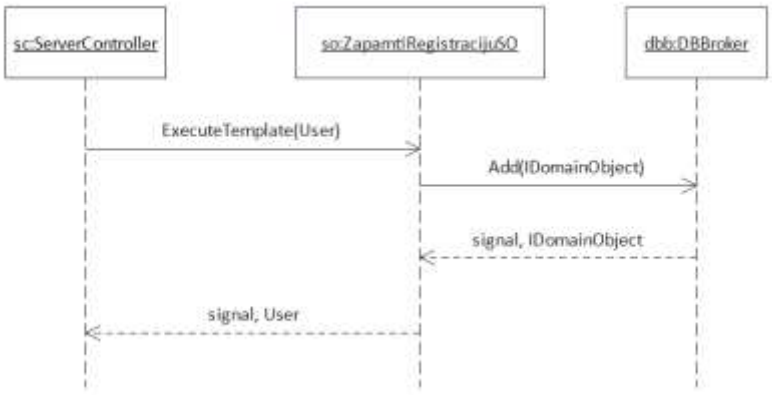
Softverski sistem realizovan je kao klijent-server aplikacija. Na serverskoj strani je nit Thread koja sadrži objekat klase Socket. Nit konstantno poziva metodu accept koja čeka da se pokrene klijentska aplikacija koja, kad se to desi, će pokušati da se poveže na server. Slanje i primanje podataka od klijenta se ostvaruje preko soketa, metoda accept kreira objekat klase Socket. Klijent šalje zahtev za izvršenje neke od SO do odgovarajuće niti (koju smo nazvali ClientHandler), koja je povezana sa tim klijentom. ClientHandler prima zahtev i dalje ga preusmerava do klasa koje su odgovorne za izvršenje SO. Nakon izvršenja SO rezultat se vraća do aplikacione logike, odnosno do klase ClientHandler na serverskoj strani koja taj rezultat šalje nazad do klijenta putem soketa.

4.3.2.2 Poslovna logika

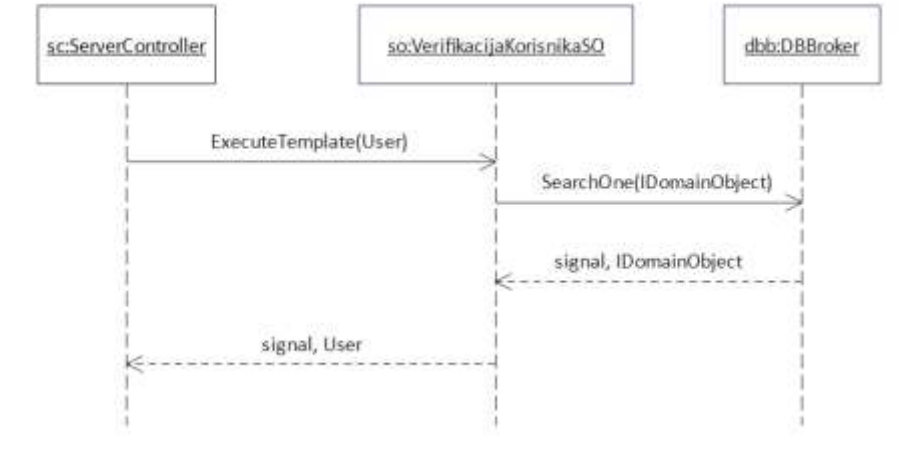
Za svaki od prethodno definisanih ugovora pravimo sistemsku operaciju, što zapravo predstavlja projektovanje ponašanja. Klasa **SystemOperationBase** koja predstavlja apstraktnu klasu koja sadrži metodu **ExecuteTemplate**, koja predstavlja šablon izvršavanja svake operacije nad bazom podataka, a kao parametar prima objekat koji implementira interfejs IDomainObject. U toj metodi se poziva metoda **Execute()**, koja je apstraktna i koje će svaka klasa sistemske operacije implementirati. Provera preduslova se izvršava na klijentskoj strain ukoliko postoji, a postuslovi se očitavaju u okviru Response objekta koji šalje server klijentu i na osnovu koga klijent zaključuje da li je operacija uspešno izvršena na serverskoj strani ili je došlo do greške.

Za svaku sistemsku operaciju treba napraviti konceptualna rešenja koja su direktno povezana sa logikom problema. Za svaki ugovor projektuje se konceptualno rešenje.

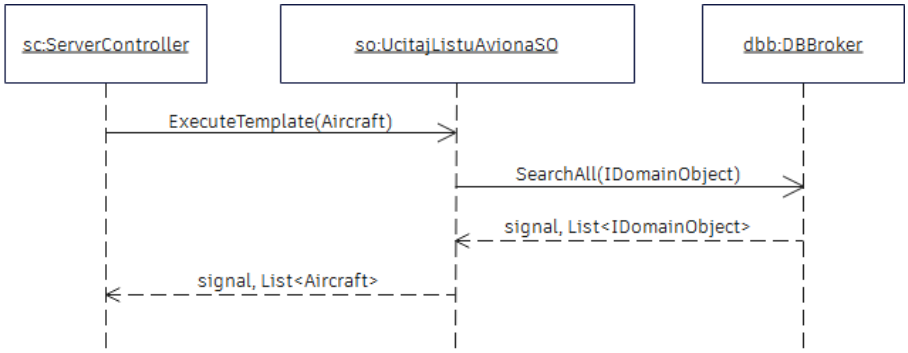
Ugovor UG1: ZapamtiRegistraciju(User) Signal;
Veza sa SK: SK1
Preduslovi: Vrijednosna i strukturna ograničenja nad objektom *User* moraju biti zadovoljena.
Postuslovi: Podaci o korisniku su zapamćeni.



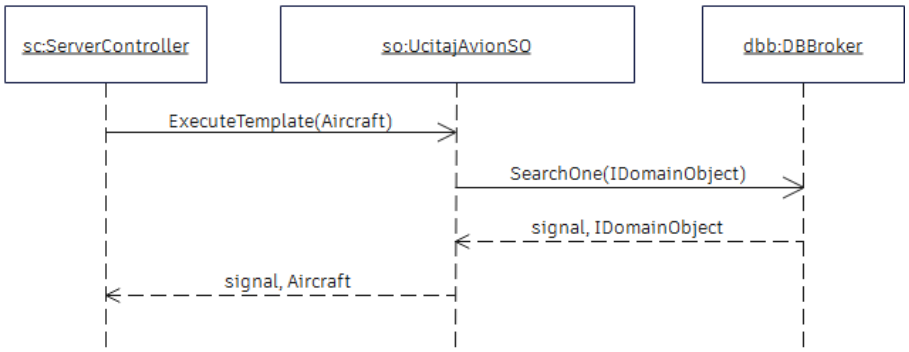
Ugovor UG2: VerifikacijaKorisnika(User) Signal;
Veza sa SK: SK2
Preduslovi:
Postuslovi:



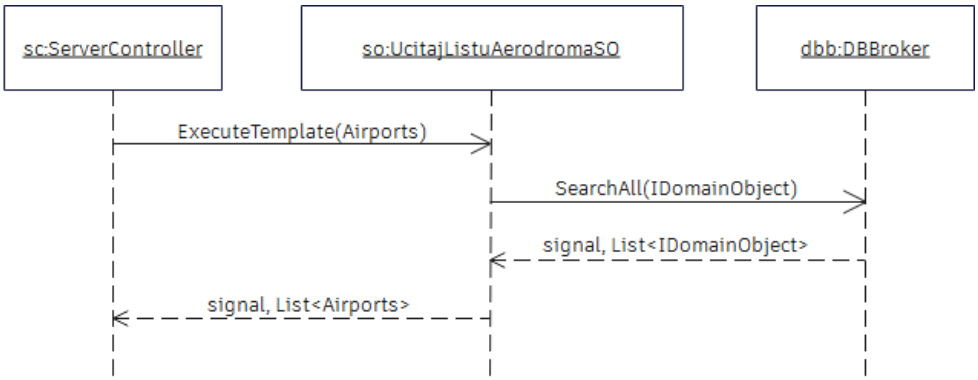
Ugovor UG3: UcitajListuAviona(Aircraft) Signal;
Veza sa SK: SK3, SK4, SK5, SK8, SK9, SK12, SK13, SK14
Preduslovi:
Postuslovi:



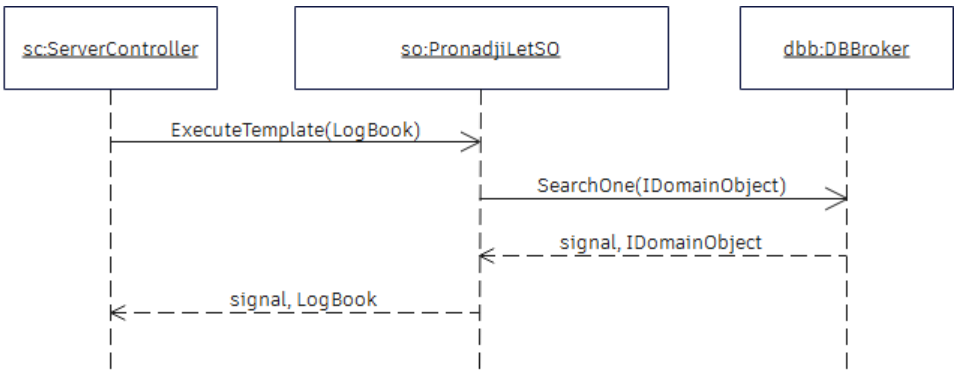
Ugovor UG4: UcitajAvion(Aircraft) Signal;
Veza sa SK: SK4, SK8
Preduslovi:
Postuslovi:



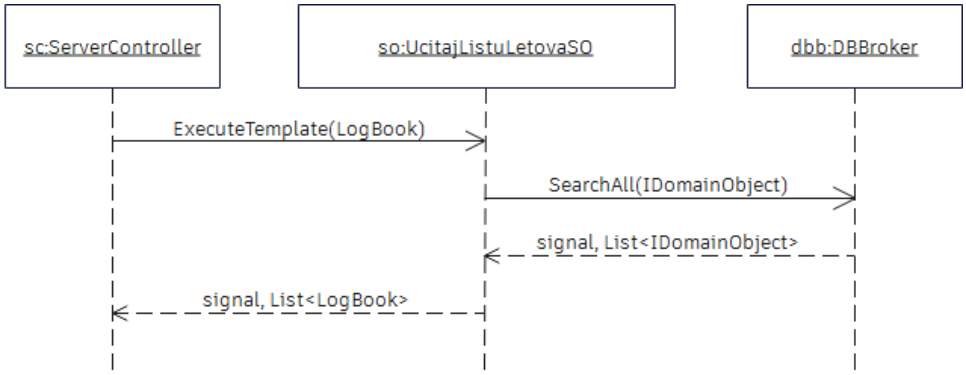
Ugovor UG5: UcitajListuAerodroma(Airports) Signal;
Veza sa SK: SK3, SK4, SK5, SK8, SK9, SK12, SK13, SK14
Preduslovi:
Postuslovi:



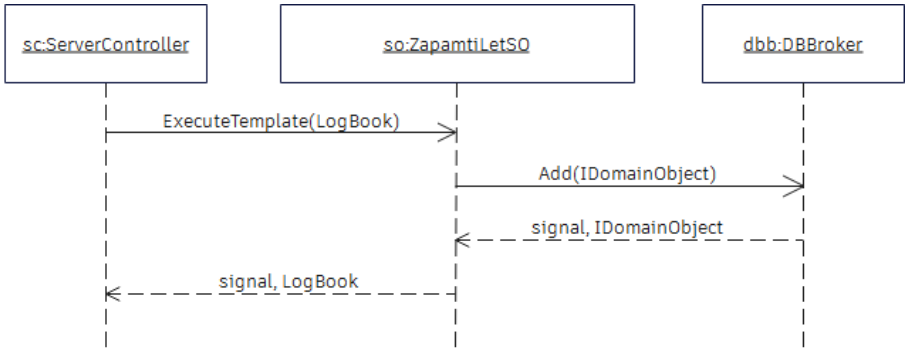
Ugovor UG6: PronadjiLet(LogBook) Signal;
Veza sa SK: SK5
Preduslovi:
Postuslovi:



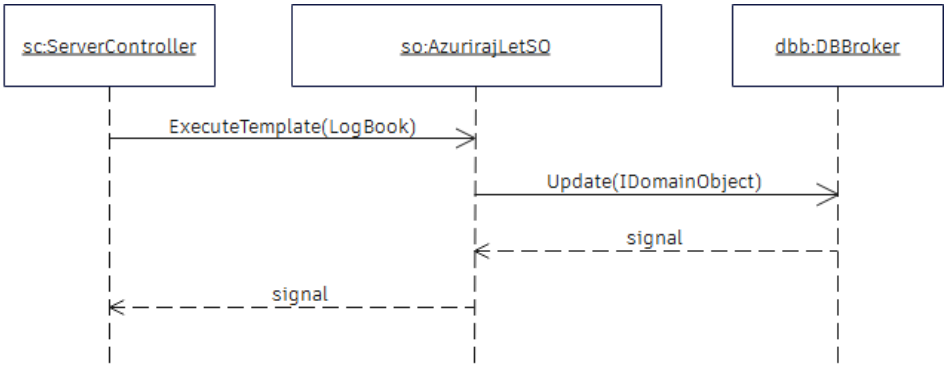
Ugovor UG7: UcitajListuLetova(LogBook) Signal;
Veza sa SK: SK5
Preduslovi:
Postuslovi:



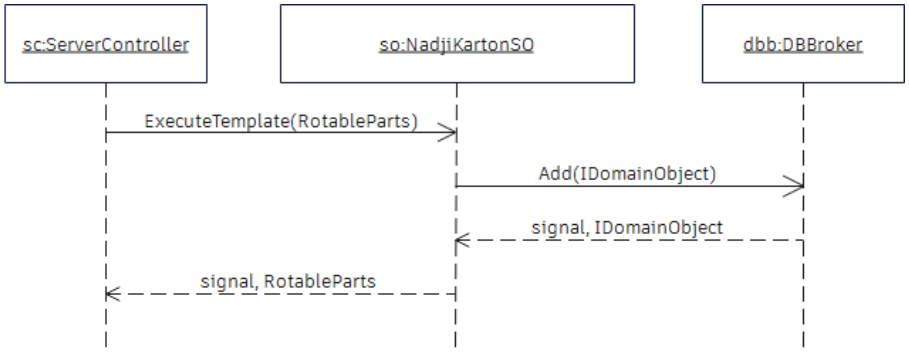
Ugovor UG8: ZapamtiLet(LogBook) Signal;
Veza sa SK: SK4
Preduslovi: Vrijednosna i strukturna ograničenja nad objektom *LogBook* moraju biti zadovoljena
Postuslovi: Podaci o letu su zapamćeni



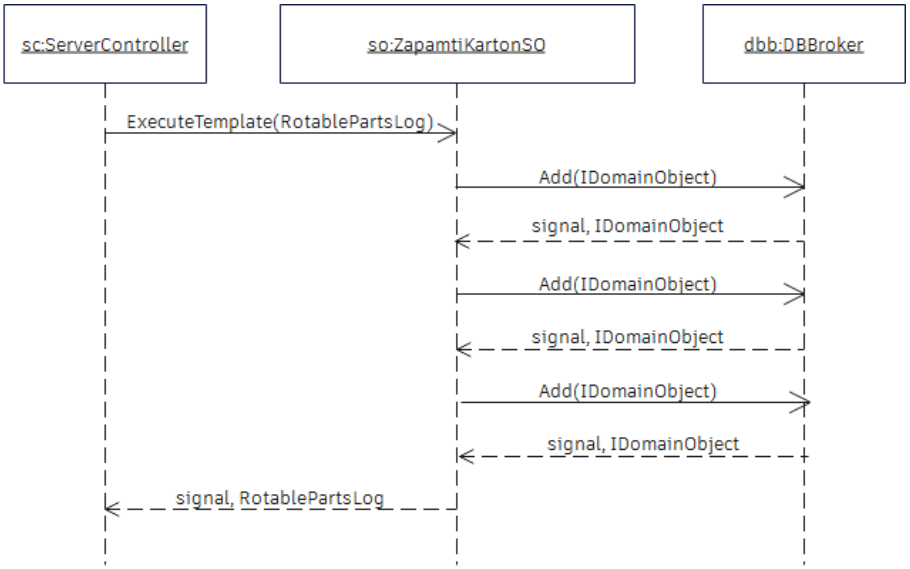
Ugovor UG9: AzurirajLet(LogBook) Signal;
Veza sa SK: SK5
Preduslovi: Vrijednosna i strukturna ograničenja nad objektom *LogBook* moraju biti zadovoljena.
Postuslovi: Podaci o ažuriranom letu su zapamćeni.



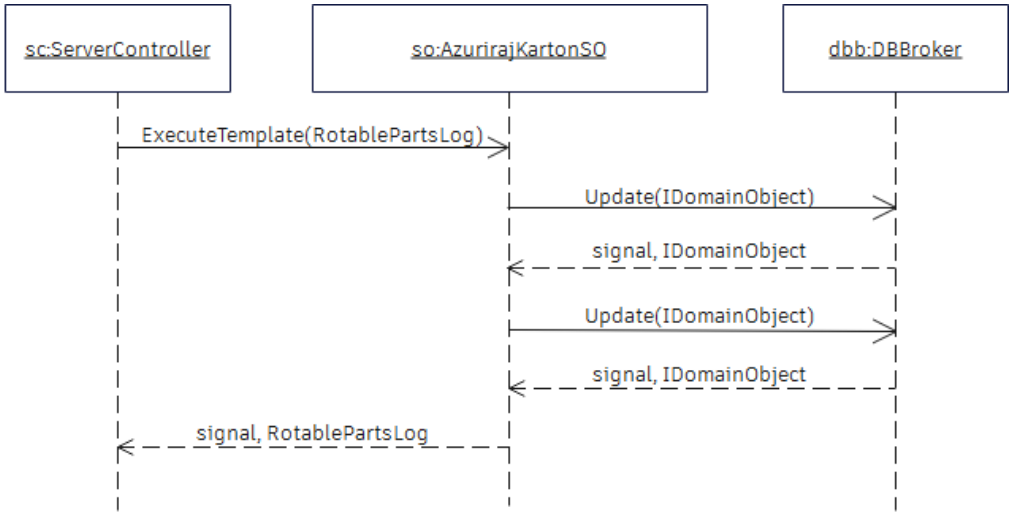
Ugovor UG10: NadjiKarton(RotableParts) Signal;
Veza sa SK: SK6, SK17
Preduslovi:
Postuslovi:



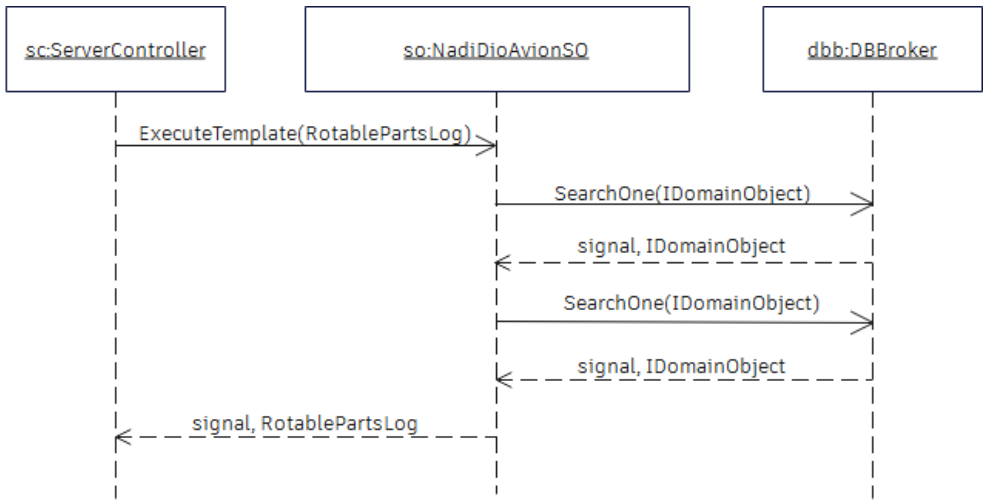
Ugovor UG11: ZapamtiKarton(RotablePartsLog) Signal;
Veza sa SK: SK6
Preduslovi: Vrijednosna i strukturna ograničenja nad objektima *RotableParts, RotablePartsLog, RotablePartsStock* moraju biti zadovoljena.
Postuslovi: Podaci iz kartona avio dijela su zapamćeni.



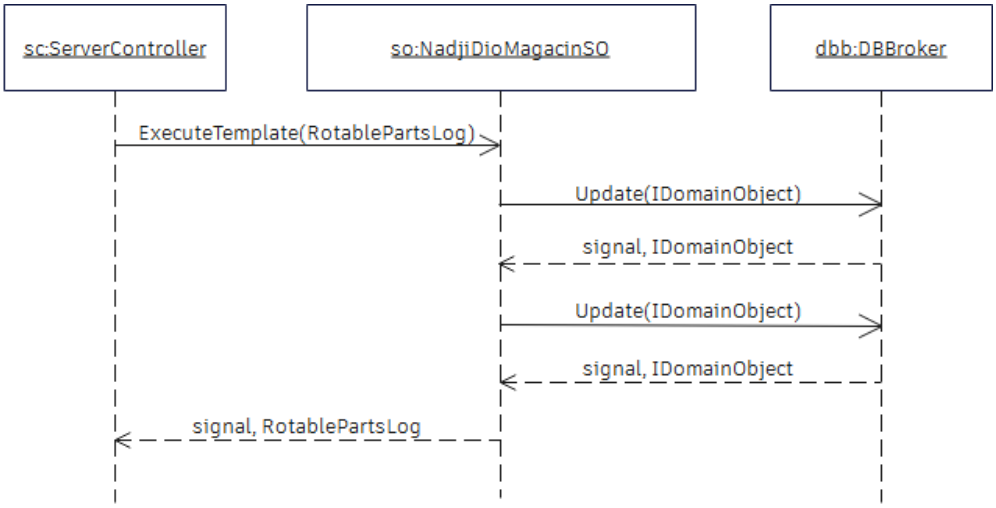
Ugovor UG12: AzurirajKarton(RotablePartsLog) Signal;
Veza sa SK: SK7
Preduslovi: Vrijednosna i strukturna ograničenja nad objektima *RotableParts, RotablePartsStock* moraju biti zadovoljena.
Postuslovi: Ažurirani podaci iz kartona avio dijela su zapamćeni.



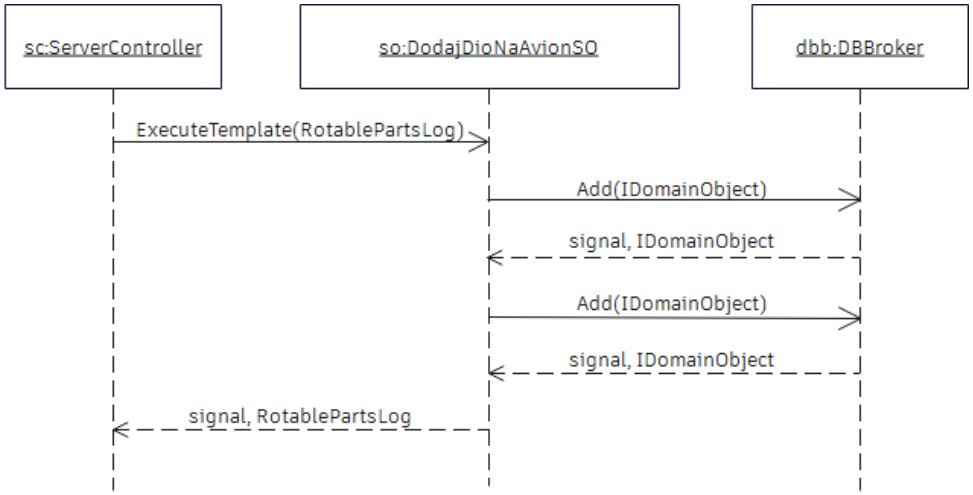
Ugovor UG13: NadjiDioAvion(RotablePartsLog) Signal;
Veza sa SK: SK9
Preduslovi:
Postuslovi:



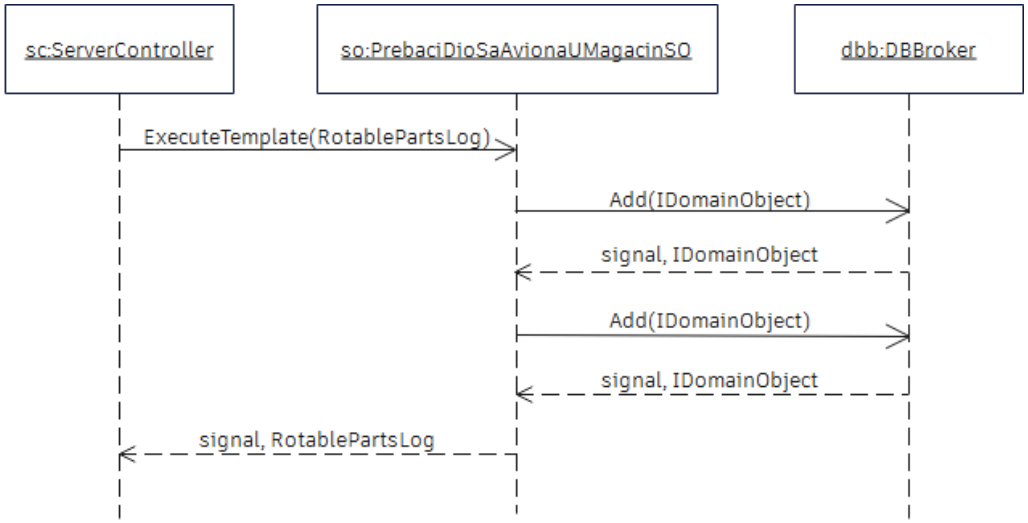
Ugovor UG14: NadjiDioMagacin(RotablePartsLog) Signal;
Veza sa SK: SK7, SK8, SK10
Preduslovi:
Postuslovi:



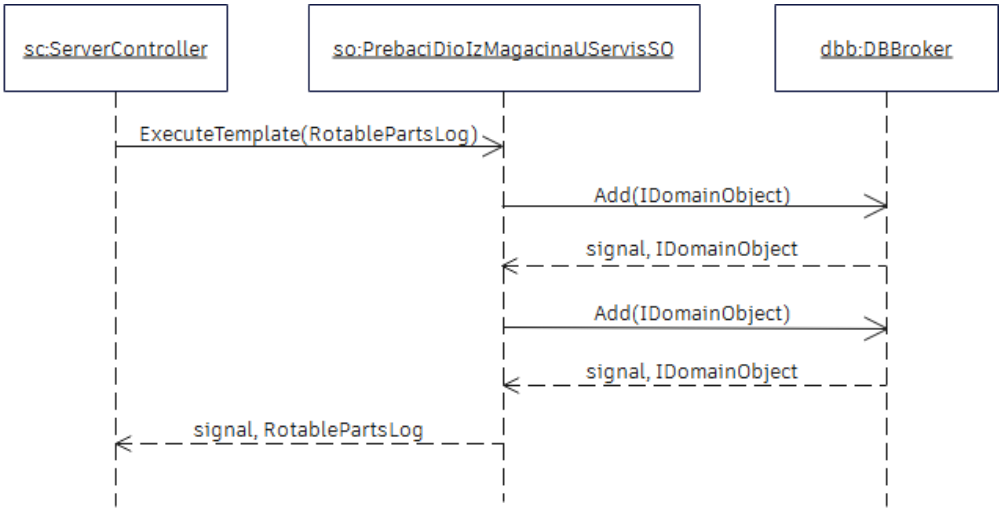
Ugovor UG15: DodajDioNaAvion(RotablePartsLog) Signal;
Veza sa SK: SK8
Preduslovi: Vrijednosna i strukturna ograničenja nad objektima *RotablePartsAircraft*, *RotablePartsLog* moraju biti zadovoljena.
Postuslovi: Podaci o instalaciji dijela na avion su zapamćeni.



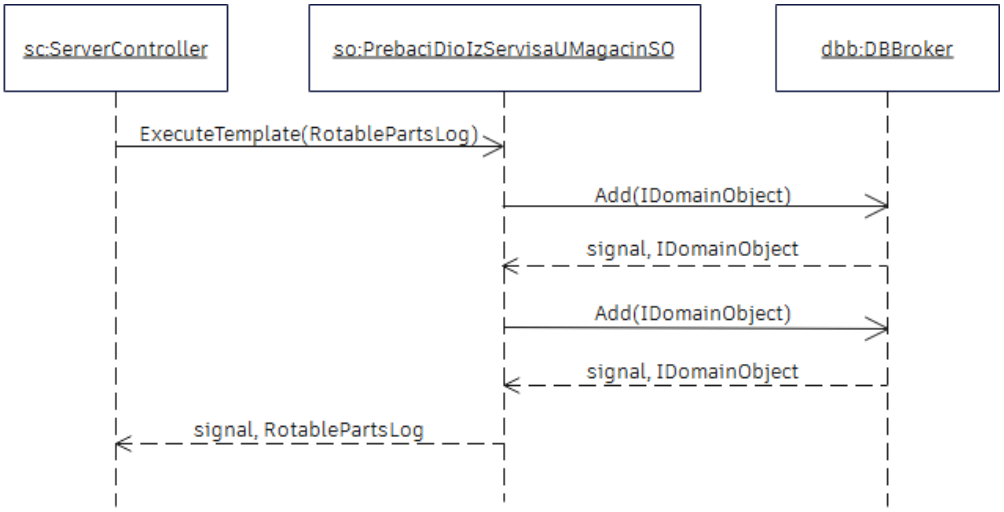
Ugovor UG16: PrebaciDioSaAvionaUMagacin(RotablePartsLog) Signal;
Veza sa SK: SK9
Preduslovi: Vrijednosna i strukturna ograničenja nad objektima *RotablePartsStock*, *RotablePartsLog* moraju biti zadovoljena.
Postuslovi: Podaci o prebacanju dijela sa aviona u magacin su zapamćeni.



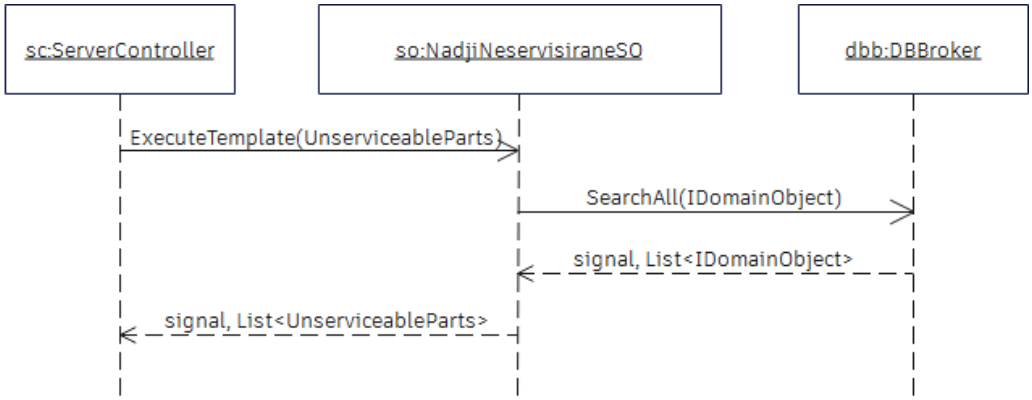
Ugovor UG17: PrebaciDioIzMagacinaUServis(RotablePartsLog) Signal;
Veza sa SK: SK10
Preduslovi: Vrijednosna i strukturna ograničenja nad objektima *RotablePartsService*, *RotablePartsLog* moraju biti zadovoljena.
Postuslovi: Podaci o prebacanju dijela iz magacina u servis su zapamćeni.



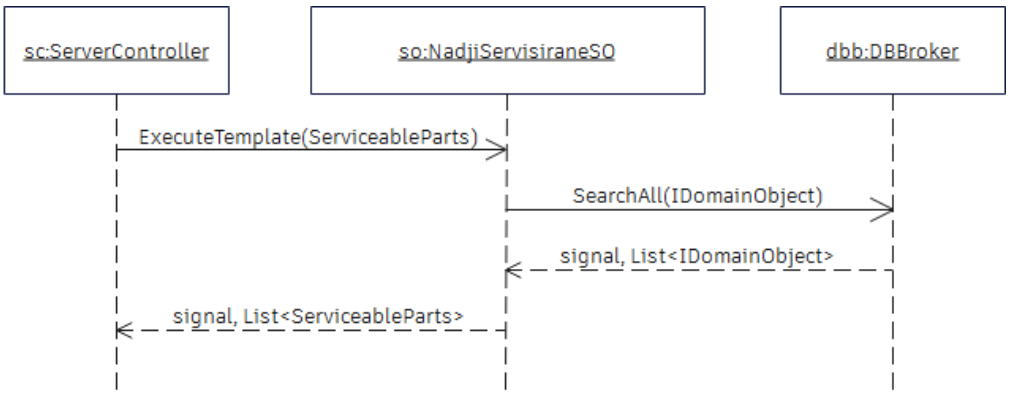
Ugovor UG18: PrebaciDioIzServisaUMagacin(RotablePartsLog) Signal;
Veza sa SK: SK16
Preduslovi: Vrijednosna i strukturna ograničenja nad objektima *RotablePartsStock*, *RotablePartsLog* moraju biti zadovoljena.
Postuslovi: Podaci o prebacanju dijela iz servisa u magacin avio dijelova su zapamćeni.



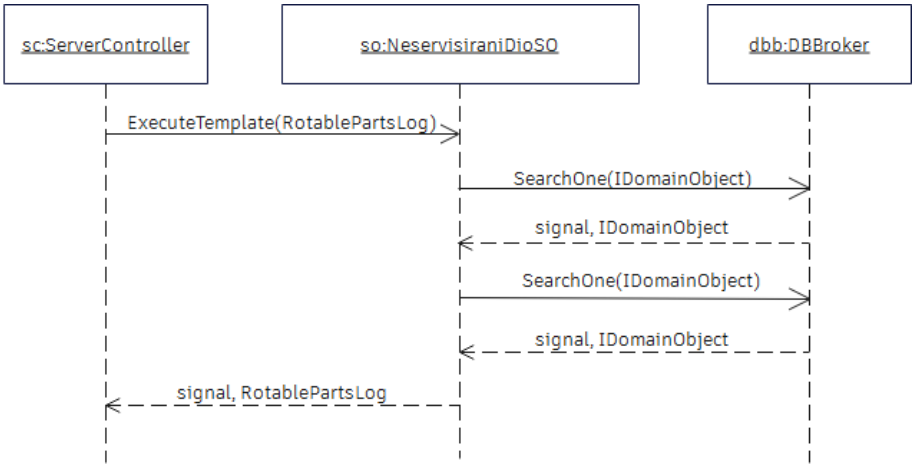
Ugovor UG19: NadjiNeservisirane(UnserviseableParts) Signal;
Veza sa SK: SK15
Preduslovi:
Postuslovi:



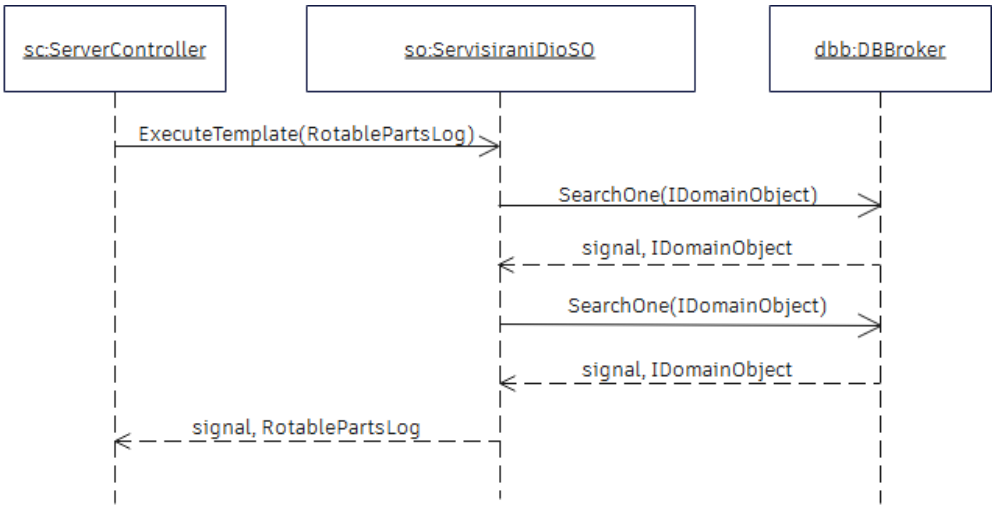
Ugovor UG20: NadjiServisirane(ServiseableParts) Signal;
Veza sa SK: SK16
Preduslovi:
Postuslovi:



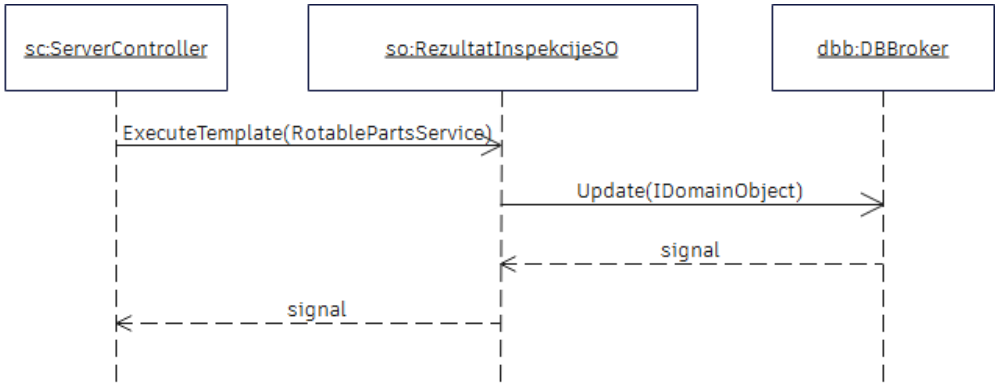
Ugovor UG21: NeservisiraniDio(RotablePartsLog) Signal;
Veza sa SK: SK15
Preduslovi:
Postuslovi:



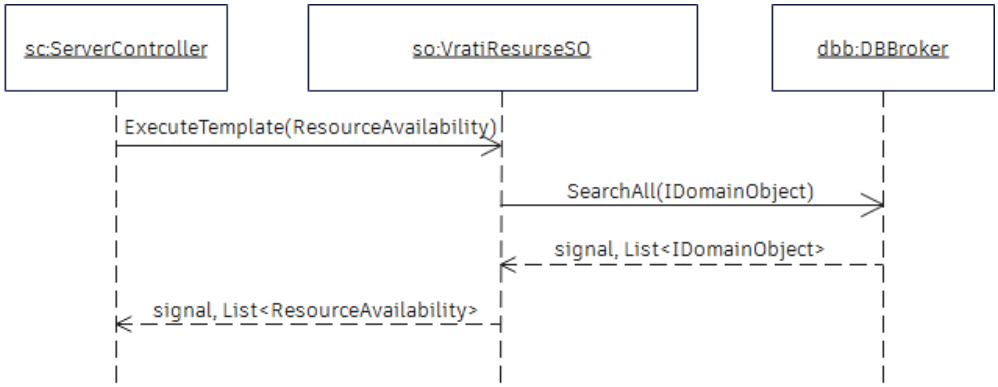
Ugovor UG22: ServisiraniDio(RotablePartsLog) Signal;
Veza sa SK: SK16
Preduslovi:
Postuslovi:



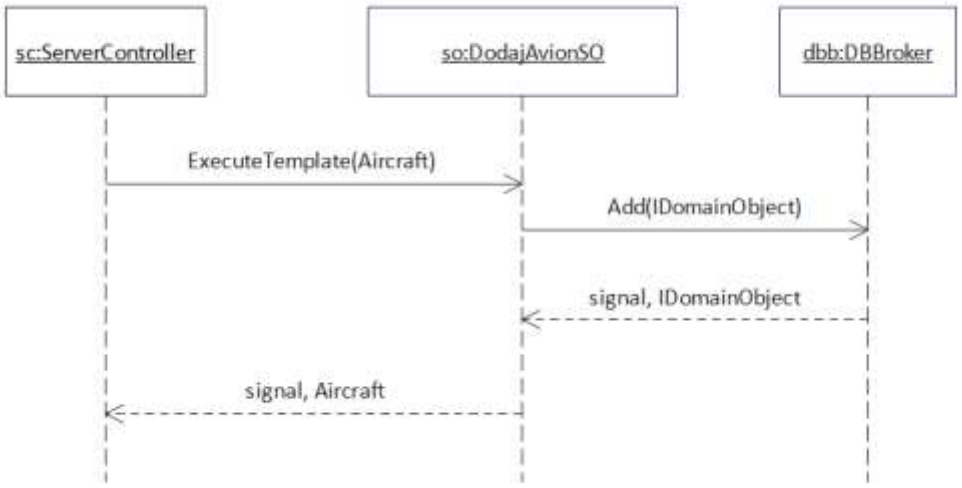
Ugovor UG23: RezultatInspekcije(RotablePartsService) Signal;
Veza sa SK: SK15
Preduslovi:
Postuslovi:



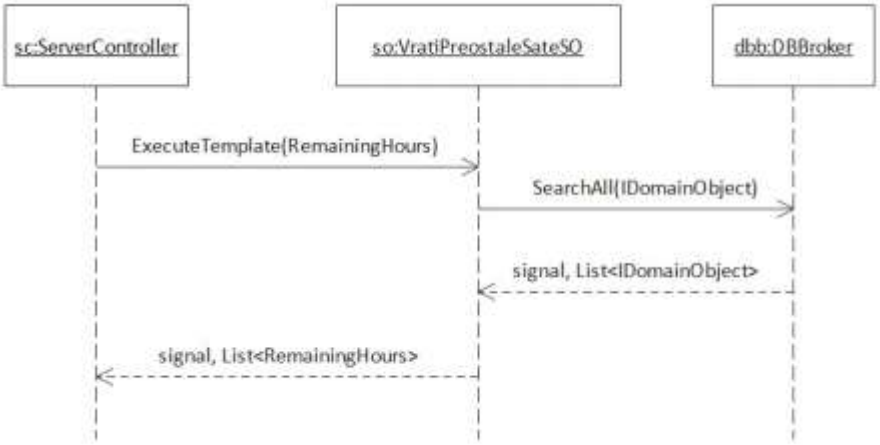
Ugovor UG24: VratiResurse(ResourceAvailability) Signal;
Veza sa SK: SK9, SK11
Preduslovi:
Postuslovi:



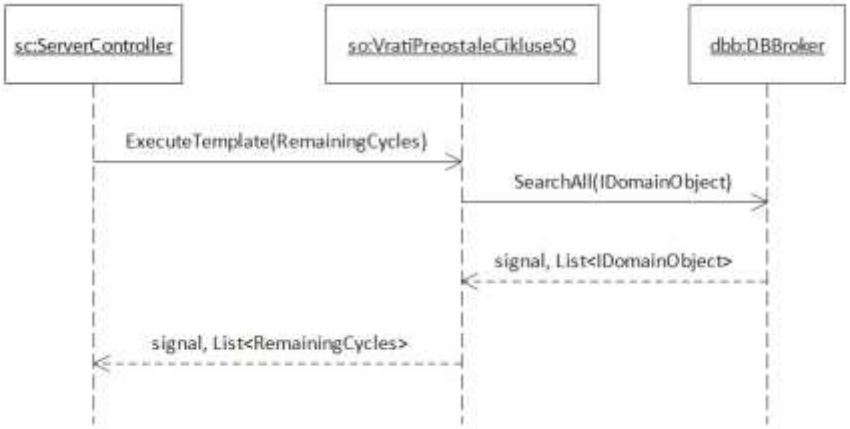
Ugovor UG25: DodajAvion(Aircraft) Signal;
Veza sa SK: SK3
Preduslovi: Vrijednosna i strukturna ograničenja nad objektom *Aircraf* moraju biti zadovoljena.
Postuslovi: Podaci o avionu su zapamćeni.



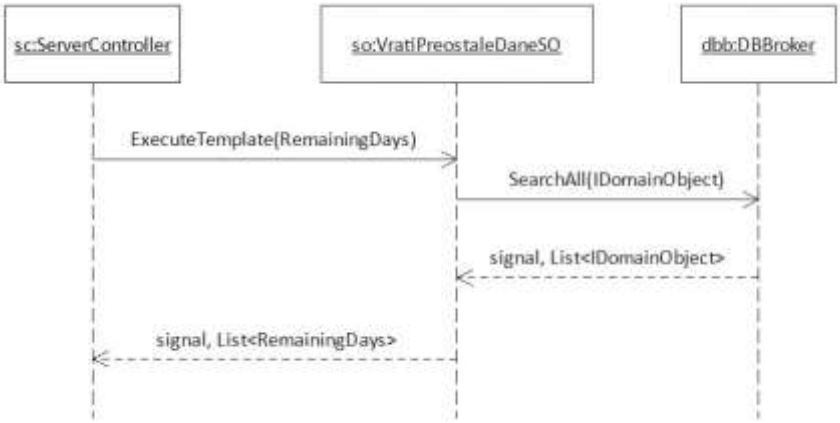
Ugovor UG26: VratiPreostaleSate(RemainingHours) Signal;
Veza sa SK: SK12
Preduslovi:
Postuslovi:



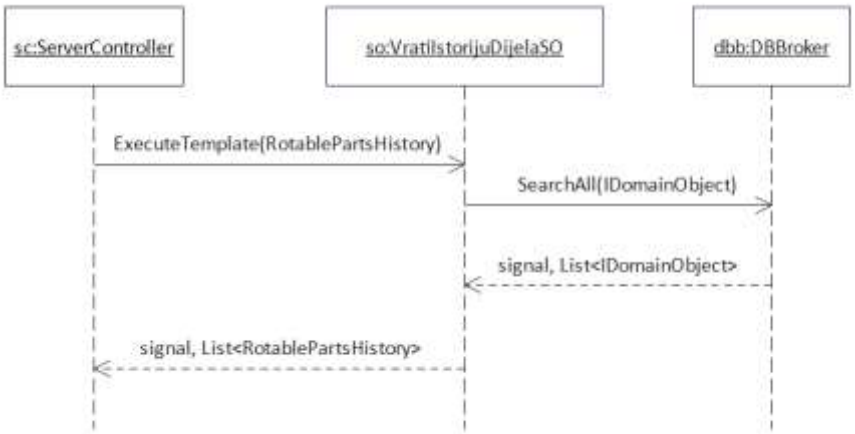
Ugovor UG27: VratiPreostaleCikluse(RemainingCycles) Signal;
Veza sa SK: SK13
Preduslovi:
Postuslovi:



Ugovor UG28: VratiPreostaleDane(RemainingDays) Signal;
Veza sa SK: SK14
Preduslovi:
Postuslovi:

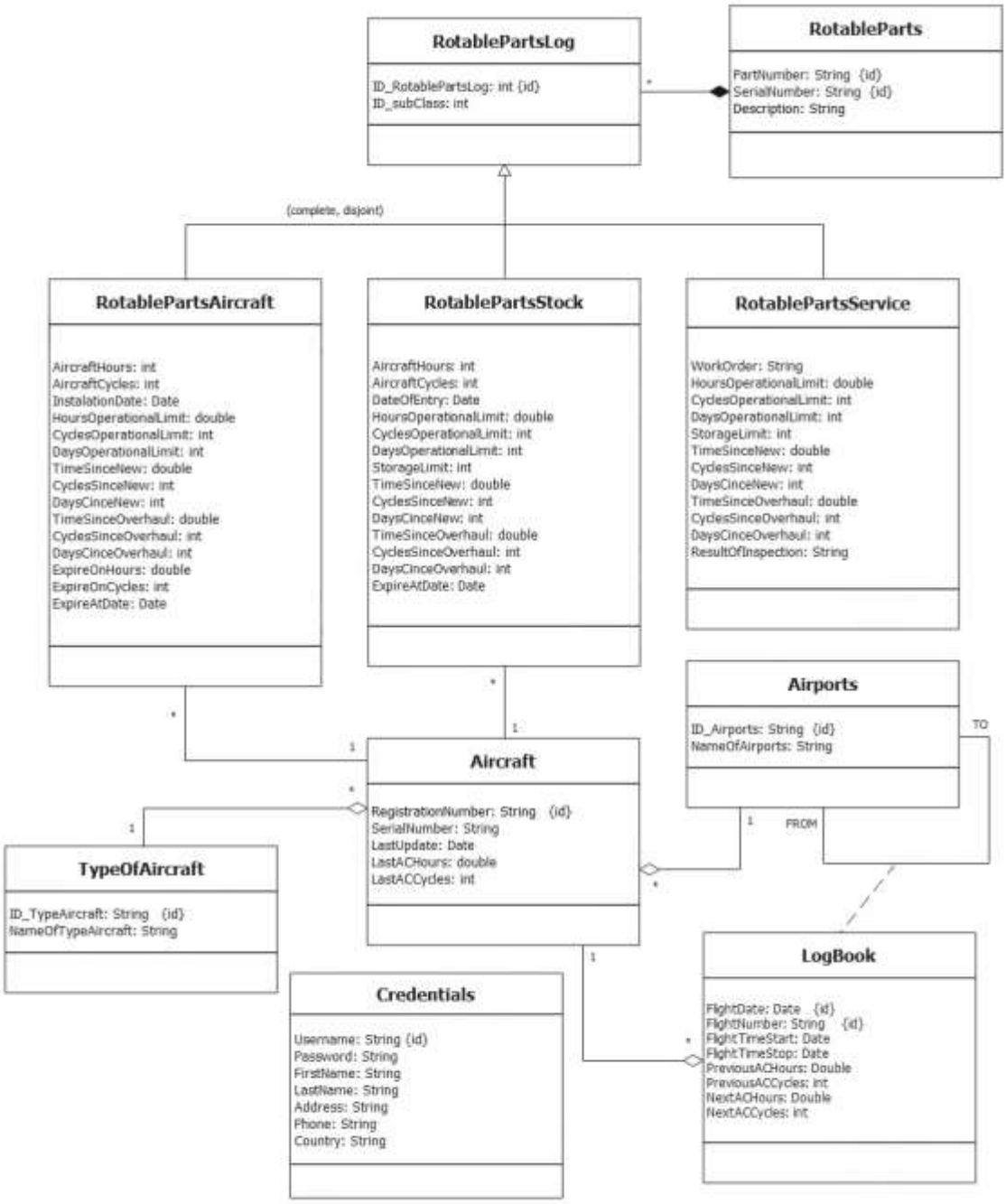


Ugovor UG29: VratiIstorijuDijela(RotablePartsHistory) Signal;
Veza sa SK: SK17
Preduslovi:
Postuslovi:



4.3.3 Projektovanje strukture softverskog sistema (domenske klase)

Na osnovu konceptualnih klasa kreiraju se softverske klase.



Identifikovane su sledeće klase:

```
public class Aircraft : IDomainObject
{
   public string RegistrationNumber { get; set; }
   public string SerialNumber { get; set; }
   public DateTime LastUpdate { get; set; }
   public Airport Airport { get; set; }
   public Decimal LastACHours { get; set; }
   public Decimal LastACCycles { get; set; }

   public override string ToString()
   {
      return RegistrationNumber;
   }

   public override bool Equals(object obj)
   {
      if (obj is Aircraft a) return a.RegistrationNumber == RegistrationNumber;
      return false;
   }

}
```

Domenska klasa 1: *Aircraft*

```
public class Airport: IDomainObject
{
   public Decimal ID_Airport { get; set; }
   public string NameOfAirports { get; set; }

   public override bool Equals(object obj)
   {
      if (obj is Airport a) return a.ID_Airport == ID_Airport;
      return false;
   }

    public override string ToString()
   {
      return NameOfAirports;
   }
}
```

Domenska klasa 2: *Airport*

```
public class RotableParts : IDomainObject
{
   public Decimal ID_RotableParts { get; set; }
   public string PartNumber { get; set; }
   public string SerialNumber { get; set; }
   public string Description { get; set; }
}
```

Domenska klasa 3: *RotableParts*

```
public class LogBook : IDomainObject
{
    public Decimal ID_LogBook{ get; set; }
    public DateTime FlightDate { get; set; }
    public string FlightNumber { get; set; }
    public Airport Airport_FROM { get; set; }
    public Airport Airport_TO { get; set; }
    public Aircraft Aircraft { get; set; }
    public DateTime FlightTimeStart { get; set; }
    public DateTime FlightTimeStop { get; set; }
    public Decimal PreviousACHours { get; set; }
    public Decimal PreviousACCycles { get; set; }
    public Decimal NextACHours { get; set; }
    public Decimal NextACCycles { get; set; }
}
```

Domenska klasa 4: *LogBook*

```
public class RotablePartsLog : IDomainObject
{
    public Decimal ID_RotablePartsLog { get; set; }
    public RotableParts RotableParts { get; set; }
    public int SubClass { get; set; }
    public Object RotablePartsSubClass { get; set; }
}
```

Domenska klasa 5: *RotablePartsLog*

```
public class RotablePartsAircraft : IDomainObject
{
    public RotablePartsLog RotablePartsLog { get; set; }
    public RotableParts RotableParts { get; set; }
    public Aircraft Aircraft { get; set; }
    public Decimal AircraftHours { get; set; }
    public Decimal AircraftCycles { get; set; }
    public DateTime InstalationDate { get; set; }
    public Decimal HoursOperationalLimit { get; set; }
    public Decimal CyclesOperationalLimit { get; set; }
    public Decimal DaysOperationalLimit { get; set; }
    public Decimal StorageLimit { get; set; }
    public Decimal TimeSinceNew { get; set; }
    public Decimal CyclesSinceNew { get; set; }
    public Decimal DaysSinceNew { get; set; }
    public Decimal TimeSinceOverhaul { get; set; }
    public Decimal CyclesSinceOverhaul { get; set; }
    public Decimal DaysSinceOverhaul { get; set; }
    public Decimal ExpireOnHours { get; set; }
    public Decimal ExpireOnCycles { get; set; }
    public DateTime ExpireAtDate { get; set; }
}
```

Domenska klasa 6: *RotablePartsAircraft*

```
public class RotablePartsStock : IDomainObject
{
    public RotablePartsLog RotablePartsLog { get; set; }
    public RotableParts RotableParts { get; set; }
    public Aircraft Aircraft { get; set; }
    public Decimal AircraftHours { get; set; }
    public Decimal AircraftCycles { get; set; }
    public DateTime DateOfEntry { get; set; }
    public Decimal HoursOperationalLimit { get; set; }
    public Decimal CyclesOperationalLimit { get; set; }
    public Decimal DaysOperationalLimit { get; set; }
    public Decimal StorageLimit { get; set; }
    public Decimal TimeSinceNew { get; set; }
    public Decimal CyclesSinceNew { get; set; }
    public Decimal DaysSinceNew { get; set; }
    public Decimal TimeSinceOverhaul { get; set; }
    public Decimal CyclesSinceOverhaul { get; set; }
    public Decimal DaysSinceOverhaul { get; set; }
    public DateTime ExpireAtDate { get; set; }
    public Boolean IsInitial { get; set; }
}
```

Domenska klasa 7: *RotablePartsStock*

```
public class RotablePartsService : IDomainObject
{
    public RotablePartsLog RotablePartsLog { get; set; }
    public RotableParts RotableParts { get; set; }
    public string WorkOrder { get; set; }
    public string WorkOrderDescription { get; set; }
    public Decimal HoursOperationalLimit { get; set; }
    public Decimal CyclesOperationalLimit { get; set; }
    public Decimal DaysOperationalLimit { get; set; }
    public Decimal StorageLimit { get; set; }
    public Decimal TimeSinceNew { get; set; }
    public Decimal CyclesSinceNew { get; set; }
    public Decimal DaysSinceNew { get; set; }
    public Decimal TimeSinceOverhaul { get; set; }
    public Decimal CyclesSinceOverhaul { get; set; }
    public Decimal DaysSinceOverhaul { get; set; }
    public Decimal ID_ResultOfInspection { get; set; }
    public Decimal NewHoursOperationalLimit { get; set; }
    public Decimal NewCyclesOperationalLimit { get; set; }
    public Decimal NewDaysOperationalLimit { get; set; }
    public Decimal NewStorageLimit { get; set; }
    public string Description { get; set; }
}
```

Domenska klasa 8: *RotablePartsService*

```
public class User : IDomainObject
   {
      public string Username { get; set; }
      public string Password { get; set; }
      public string Firstname { get; set; }
      public string Lastname { get; set; }
      public string Address { get; set; }
      public string Phone { get; set; }
      public string Country { get; set; }
      public string IPAddress { get; set; }
      public string LogTime { get; set; }

      public override bool Equals(object obj)
      {
         if (obj is User u) return u.Username == Username && u.Password == Password;
         return false;
      }

   }
```

Domenska klasa 9: ***User***

```
   public interface IDomainObject
   {
      List<string> TableName { get; }
      int TableNameIndex { get; set; }
      List<string> SelectFields { get; }
      int SelectFieldsIndex { get; set; }
      List<string> Condition { get; }
      int ConditionIndex { get; set; }
      string InsertValues { get; }
      string UpdateValues { get; }
      List<IDomainObject> ReadMultipleRow(SqlDataReader reader);
      IDomainObject ReadSingleRow(SqlDataReader reader);
      string SelectOrderBy { get; }
   }
```

Domenska klasa 10: ***Opšta domenska klasa***

Pored njih dodate su i sledeće klase:

```
public enum Operation
   {
      GetAircrafts,
      GetAircraft,
      GetAirports,
      AddLogBook,
      GetLogBook,
      GetFlight,
      UpdateLogBook,
      SearchRotableParts,
      AddRotableParts,
      UpdateRotableParts,
      SearchRotablePartsStock,
      InstallToAircraft,
      SearchResourceAvailability,
      SearchRotablePartsAircraft,
      SendFromAircraftToStock,
      SendToService,
      SearchRotablePartsService,
      SearchUnservicable,
      ServiceInspection,
      SearchServiceable,
      SendFromServiceToStock,
      Login,
      EndConnection,
      AddUser,
      SearchRemainingHours,
      SearchRemainingCycles,
      SearchRemainingDays,
      SearchRotablePartHistory,
      AddAircraft
   }
```

Klasa 1: Common enum *Operation*

Operation – Služi za čuvanje operacija koje se šalju od klijenta ka serveru.

```
   public class Request
   {
      public Operation Operation { get; set; }
      public object RequestObject { get; set; }

      public Request(Operation operation)
      {
         this.Operation = operation;

      }
      public Request(Operation operation, object requestObject)
      {
         this.Operation = operation;
         this.RequestObject = requestObject;

      }
   }
```

Klasa 2: Common klasa *Request*

Request – Služi za slanje objekta od klijenta ka serveru. Sadrži jedan Object atribut koji predstavlja objekat nad kojim treba izvršiti zahtijevanu operaciju i atribut koji predstavlja operaciju koja treba da se izvrši.

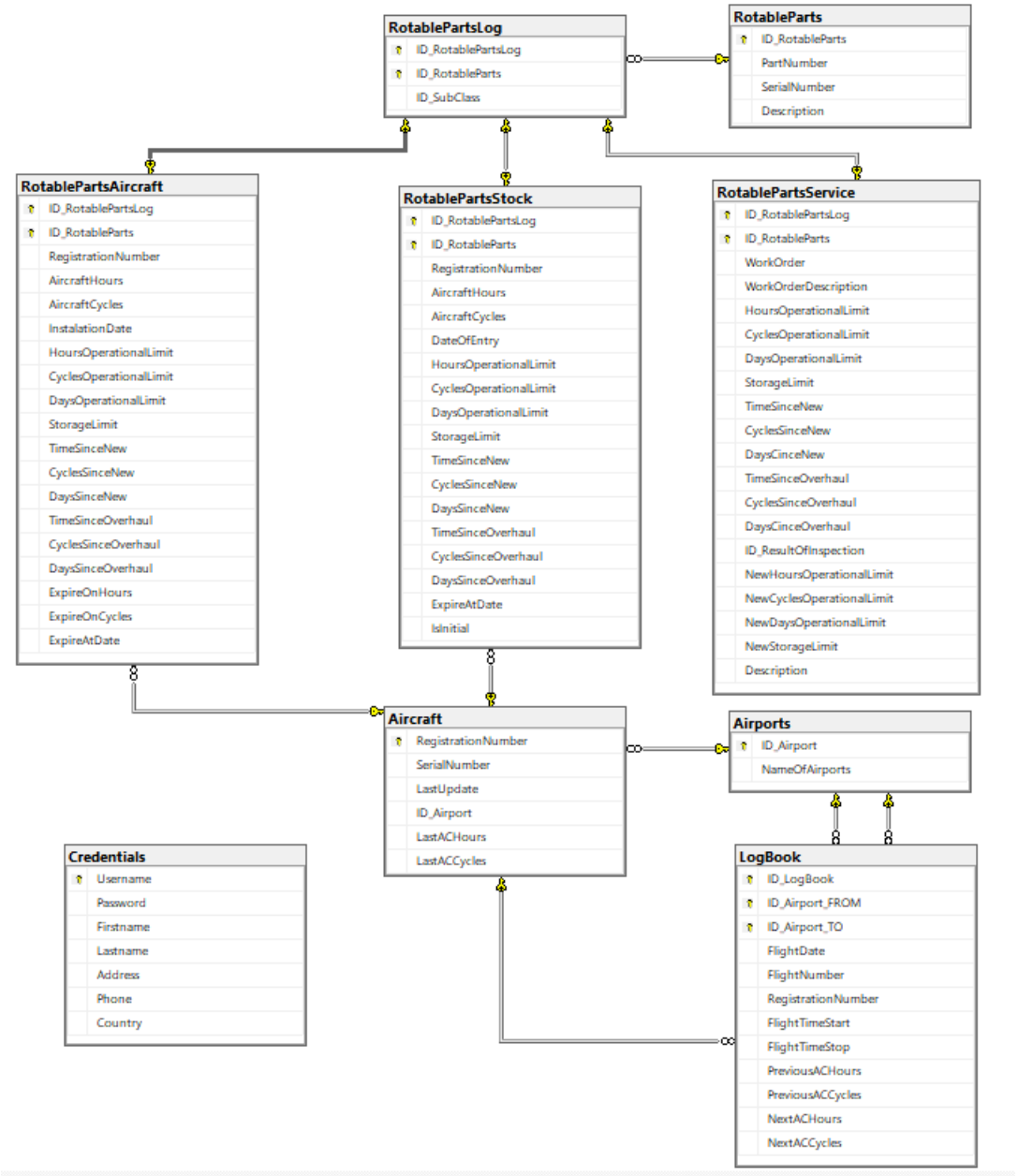
```
public class Response
{
    public string Message { get; set; }
    public object ResponseObject { get; set; }
    public bool IsSuccessful { get; set; }
}
```

Klasa 3: Common klasa *Response*

Response – Služi za slanje objekta od servera ka klijentu. Sadrži jedan Object atribut koji predstavlja rezultat izvršene operacije, jedan Message atribut koji predstavlja poruku izuzetaka koji će se generisati na klijentskoj strani ako atribut IsSuccessful ima False vrijednost.

4.3.4 Projektovanje skladišta podataka

Na osnovu relacionog modela i ograničenja projektovane su tabele baze podataka koje koristi naš softverski sistem.



	Column Name	Data Type	Allow Nulls
🔑	RegistrationNumber	varchar(10)	☐
	SerialNumber	varchar(50)	☐
	LastUpdate	datetime	☐
	ID_Airport	numeric(18, 0)	☐
	LastACHours	decimal(18, 2)	☐
	LastACCycles	decimal(18, 0)	☐

Slika 11: Tabela *Aircraft*

	Column Name	Data Type	Allow Nulls
🔑	ID_Airport	numeric(18, 0)	☐
	NameOfAirports	varchar(50)	☐

Slika 12: Tabela *Airport*

	Column Name	Data Type	Allow Nulls
🔑	Username	varchar(50)	☐
	Password	varchar(64)	☑
	Firstname	varchar(50)	☑
	Lastname	varchar(50)	☑
	Address	varchar(50)	☑
	Phone	varchar(50)	☑
	Country	varchar(50)	☑

Slika 13: Tabela *Credentials*

	Column Name	Data Type	Allow Nulls
🔑	ID_LogBook	numeric(18, 0)	☐
🔑	ID_Airport_FROM	numeric(18, 0)	☐
🔑	ID_Airport_TO	numeric(18, 0)	☐
	FlightDate	date	☐
	FlightNumber	varchar(10)	☐
	RegistrationNumber	varchar(10)	☐
	FlightTimeStart	datetime	☐
	FlightTimeStop	datetime	☐
	PreviousACHours	numeric(18, 2)	☐
	PreviousACCycles	numeric(18, 0)	☐
	NextACHours	numeric(18, 2)	☐
	NextACCycles	numeric(18, 0)	☐

Slika 14: Tabela *LogBook*

	Column Name	Data Type	Allow Nulls
🔑	ID_RotableParts	numeric(18, 0)	☐
	PartNumber	varchar(20)	☐
	SerialNumber	varchar(20)	☐
	Description	varchar(MAX)	☐

Slika 15: Tabela *RotableParts*

	Column Name	Data Type	Allow Nulls
🔑	ID_RotablePartsLog	numeric(18, 0)	☐
🔑	ID_RotableParts	numeric(18, 0)	☐
	ID_SubClass	int	☐

Slika 16: Tabela *RotablePartsLog*

	Column Name	Data Type	Allow Nulls
🔑	ID_RotablePartsLog	numeric(18, 0)	☐
🔑	ID_RotableParts	numeric(18, 0)	☐
	RegistrationNumber	varchar(10)	☐
	AircraftHours	numeric(18, 2)	☐
	AircraftCycles	numeric(18, 0)	☐
	InstalationDate	date	☐
	HoursOperationalLimit	numeric(18, 2)	☑
	CyclesOperationalLimit	numeric(18, 0)	☑
	DaysOperationalLimit	numeric(18, 0)	☑
	StorageLimit	numeric(18, 0)	☑
	TimeSinceNew	numeric(18, 2)	☑
	CyclesSinceNew	numeric(18, 0)	☑
	DaysSinceNew	numeric(18, 0)	☑
	TimeSinceOverhaul	numeric(18, 2)	☑
	CyclesSinceOverhaul	numeric(18, 0)	☑
	DaysSinceOverhaul	numeric(18, 0)	☑
	ExpireOnHours	numeric(18, 2)	☑
	ExpireOnCycles	numeric(18, 0)	☑
	ExpireAtDate	date	☑

Slika 17: Tabela *RotablePartsAircraft*

	Column Name	Data Type	Allow Nulls
🔑	ID_RotablePartsLog	numeric(18, 0)	☐
🔑	ID_RotableParts	numeric(18, 0)	☐
	RegistrationNumber	varchar(10)	☑
	AircraftHours	numeric(18, 2)	☑
	AircraftCycles	numeric(18, 0)	☑
	DateOfEntry	date	☐
	HoursOperationalLimit	numeric(18, 2)	☑
	CyclesOperationalLimit	numeric(18, 0)	☑
	DaysOperationalLimit	numeric(18, 0)	☑
	StorageLimit	numeric(18, 0)	☑
	TimeSinceNew	numeric(18, 2)	☑
	CyclesSinceNew	numeric(18, 0)	☑
	DaysSinceNew	numeric(18, 0)	☑
	TimeSinceOverhaul	numeric(18, 2)	☑
	CyclesSinceOverhaul	numeric(18, 0)	☑
	DaysSinceOverhaul	numeric(18, 0)	☑
	ExpireAtDate	date	☑
	IsInitial	bit	☑

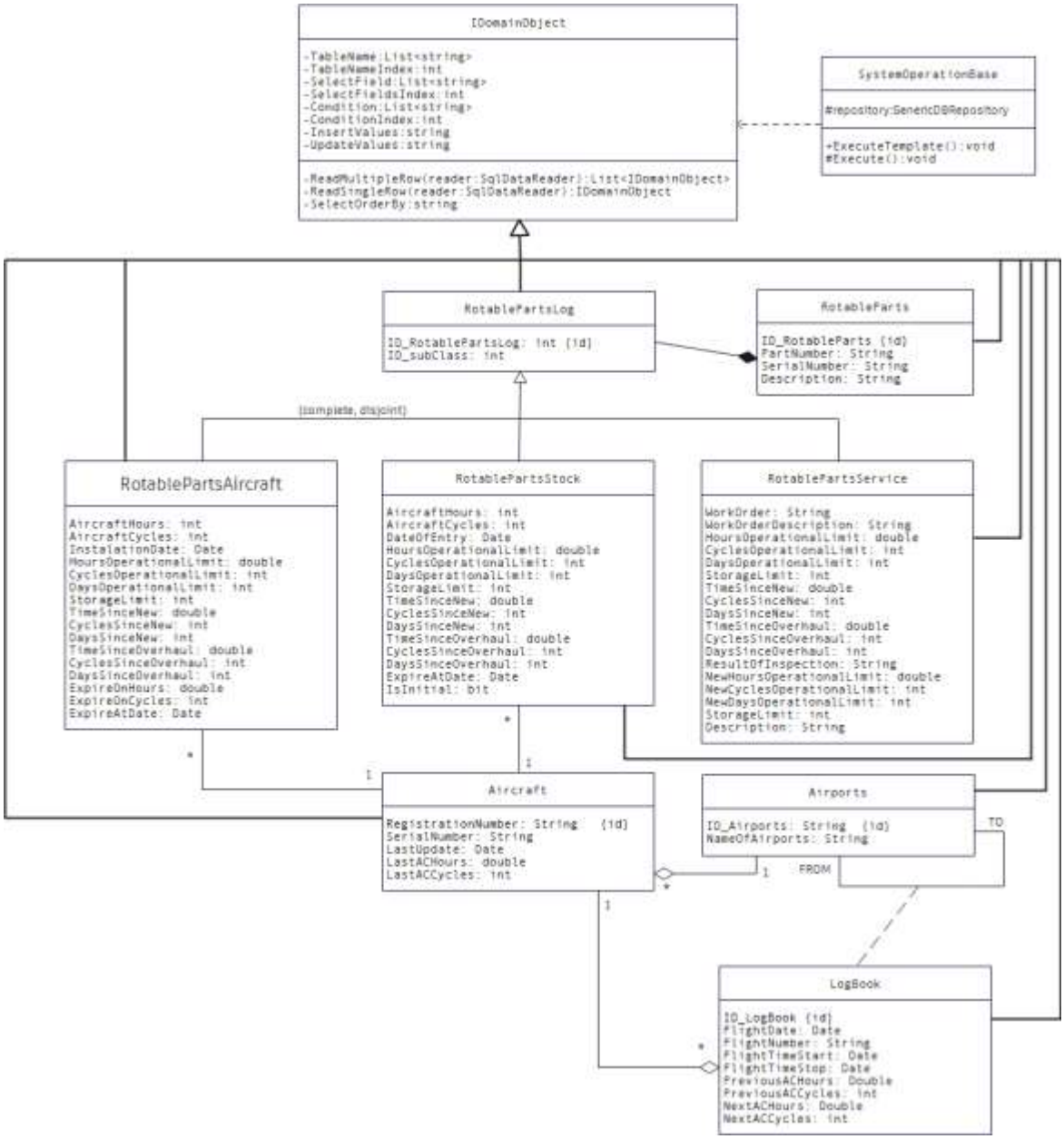
Slika 18: Tabela *RotablePartsStock*

	Column Name	Data Type	Allow Nulls
🔑	ID_RotablePartsLog	numeric(18, 0)	☐
🔑	ID_RotableParts	numeric(18, 0)	☐
	WorkOrder	varchar(50)	☐
	WorkOrderDescription	varchar(MAX)	☐
	HoursOperationalLimit	numeric(18, 2)	☑
	CyclesOperationalLimit	numeric(18, 0)	☑
	DaysOperationalLimit	numeric(18, 0)	☑
	StorageLimit	numeric(18, 0)	☑
	TimeSinceNew	numeric(18, 2)	☑
	CyclesSinceNew	numeric(18, 0)	☑
	DaysCinceNew	numeric(18, 0)	☑
	TimeSinceOverhaul	numeric(18, 2)	☑
	CyclesSinceOverhaul	numeric(18, 0)	☑
	DaysCinceOverhaul	numeric(18, 0)	☑
	ID_ResultOfInspection	numeric(18, 0)	☑
	NewHoursOperationalLimit	numeric(18, 2)	☑
	NewCyclesOperationalLimit	numeric(18, 0)	☑
	NewDaysOperationalLimit	numeric(18, 0)	☑
	NewStorageLimit	numeric(18, 0)	☑
	Description	varchar(MAX)	☑

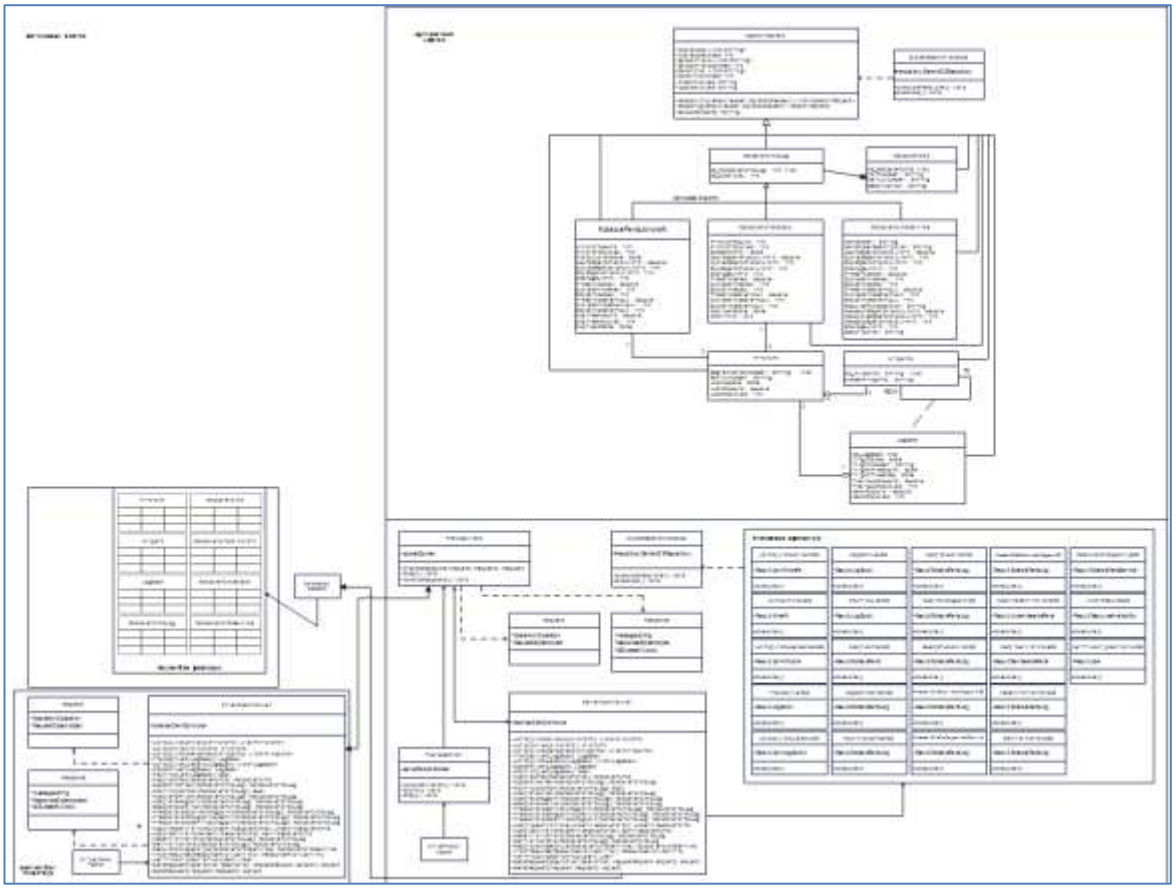
Slika 19: Tabela *RotablePartsService*

Za komunikaciju sa bazom podataka pravimo generičku klasu SystemOperationBase koja ima apstraktni metodu Execute() koju će implementirati svaka klasa sistemske operacije koja će se izvršavati. Ona se služi klasom DBBroker koja je implementirana pomoću Singleton.

Kao rezultat projektovanja klase SystemOperationBase, opšteg domenskog objekta IDomainObject i domenskih objekata dobijamo sledeći dijagram klasa:



Slika 20: Dijagram klasa opšte sistemske operacije, opšteg domenskog objekta i ostalih domenskih klasa

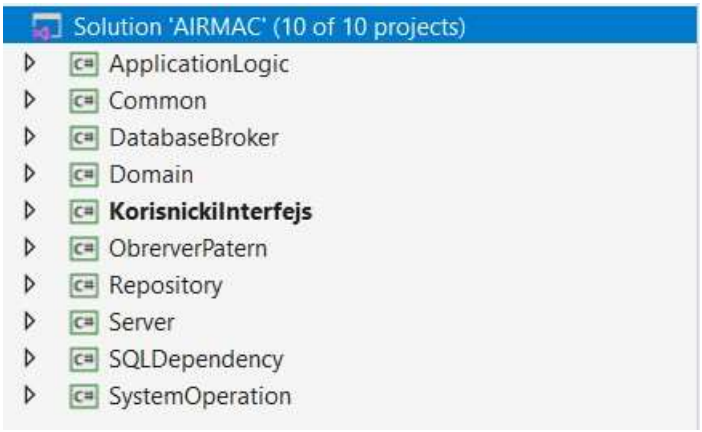


Slika 21: Konačna arhitektura softverskog sistema

4.4 FAZA IMPLEMENTACIJE

Softverski sistem je razvijen u programskom jeziku C#, razvojno okruženje Microsoft Visual Studio 2019. Kao sistem za upravljanje bazom podataka korišćen je MS SQL Express Edition.

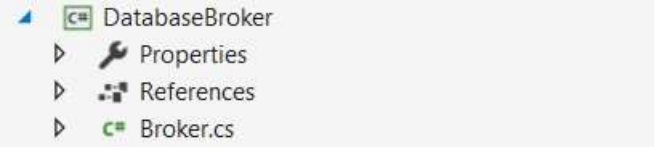
Organizacija projekta je prikazana na slici:



Projektom ApplicationLogic implementirana je poslovna logika koja je sa jedne strane povezana sa ClientHandler klasom koja opslužuje klijente koji se preko komunikacije povezuju na server, a sa druge strane preko opšte sistemske opracije poziva sistemske operacije identifikovane za svaki slučaj korišćenja.

Projektom Common implementirane su zajedničke klase koje koriste klijentska i serverska strana. To su klase Request, Response, CommunicationHelper i enum Operation.

Projektom DatabaseBroker implementirana je klasa za pristup bazi podataka u smislu otvaranja i zatvaranja konekcije sa bazom podataka kao i za rad sa transakcijama.



Projektom Domain specificirana je opšta domenska klasa kao i domenske klase koje preslikavaju database objekte i implementiraju opštu domensku klasu.

Projektom KorisničkiInterfejs implementirana je klijentska strana koja identifijuje tri osnovne cjeline. Kao prva cjelina to su korisničke forme koje preko grafičkih objekata prikupljaju ili prezentuju podatke vezane za svaki slučaj korišćenja. Kao druga cjelina to su GUI kontroleri korisničkih formi. Za svaku formu implementiran je poseban GUI kontroler čija je uloga da grafičke objekte konvertuje u domenske objekte i obrnuto. GUI kontroler ima vezu sa sledećom cjelinom, a to je ServerCommunication koja je zadužena za uspostavljanje TCP komunikacije sa serverskom stranom.

Projektom Repository implementirane su funkcije za rad sa bazom podataka kao što su: Add, Update, Delete, SearchAll, SearchOne. Generički repozitorij implementira

specifikaciju generičkog intergejsa IRepository. U implementaciji ovih funkcija korišnjen je opšti domenski tip IDomainObject.

Projektom Server implementirana je serverska strana aplikacije koja prati sve prijavljenje korisnike na sistemu a takođe zadužena je i za opsluživanje svih zahtjeva od strane klijenata. U ovom radu korišnjena je sinhrona komunikacija izmedju korisničkih zahtjeva i serverskih odgovora.

Projektom ObserverPatern implementiran je observer patern.

Projektom SQLDependency implementiran je dependency sa tabelom LogBook. On služi da se svaki put kad se izvrši promjena u LogBook-u u bazi ista izvrši i na formi.

Projektom SistemOperation implementirana je opšta sistemska operacija kao i sve sistemske operacije identifikovane u svim slučajevima korišćenja koje implementiraju svaka na svoj način abstract metodu Execute() opšte sistemske operacije.


SystemOperation
Properties
References
Exceptions
SO
AzurirajKartonSO.cs
AzurirajLetSO.cs
DbUpdateException.cs
DodajAvionSO.cs
DodajDioNaAvionSO.cs
NadjiDioAvionSO.cs
NadjiDioMagacinSO.cs
NadjiKartonSO.cs
NadjiNeservisiraneSO.cs
NadjiServisiraneSO.cs
NeservisiraniDioSO.cs
PrebaciDioIzMagacinaUServisSO.cs
PrebaciDioIzServisaUMagacinSO.cs
PrebaciDioSaAvionaUMagacinSO.cs
PronadjiLetSO.cs
RezultatInspekcijeSO.cs
ServisiraniDio.cs
UcitajAvionSO.cs
UcitajListuAerodromaSO.cs
UcitajListuAvionaSO.cs
UcitajListuLetovaSO.cs
VerifikacijaKorisnikaSO.cs
VratiIstorijuDijelaSO.cs
VratiPreostaleCikluseSO.cs
VratiPreostaleDaneSO.cs
VratiPreostaleSateSO.cs
VratiResurseSO.cs
ZapamtiKartonSO.cs
ZapamtiLetSO.cs
ZapamtiRegistracijuSO.cs
SystemOperationBase.cs

4.5 FAZA TESTIRANJA

U fazi testiranja, testiran je svaki od implementiranih slučajeva korišćenja. Prilikom testiranja svakog slučaja korišćenja, pored unijetih pravilnih podataka, unošeni su i nepravilni podaci da bi se utvrdio rezultat izvršenja. Nakon faze testiranja, softver je spreman za korišćenje od strane krajnjeg korisnika.

5 ZAKLJUČAK

Za razvoj softverskog sistema AIRMAC, korišćena je pojednostavljena Larmanova metoda za razvoj softvera. Trenutno razvijeno softversko rešenje jeste primenljivo, ali takođe ostavlja puno prostora, da se korišćenjem savremenih tehnologija, poboljšaju i otklone potencijalni nedostaci, kao i prostora da se nadograde nove funkcionalnosti koje bi zadovoljile potrebe korisnika, pruživši mu veću upotrebnu vrijednost i doživljaj.

6 LITERATURA

Knjiga:

[1] Vlajić, S. (2020). Projektovanje softvera skripta – radni materijal ver 1.3. Beograd
[2] Vlajić, S. (2014). Softverski paterni. Beograd
[2] Coronel, C., Morris, S., Rob, P. (2010). Database Systems: Design, Implementation, and Management,
[3] Price, M. (2022). C# 11 and .NET 7 – Modern Cross-Platform Development
[4] Cleary, S. (2019). Concurrency in C# Cookbook, Second Edition
[5] Skeet, J. (2019). C# in Depth Fourth Edition
[6] Aschenbrenner, K. (2008). Pro SQL Server 2008 Service Broker

Internet sajtovi:

[7] AircraftDB (https://opensky-network.org/aircraft-database)
[8] Stack Overflow (https://stackoverflow.com/)
[9] SQL Server Central (https://www.sqlservercentral.com/)
[10] W3Schools (https://www.w3schools.com/sql/)
[11] CodeProject (https://www.codeproject.com/Articles/3238/Applying-Observer-Pattern-in-NET-Remoting)
[12] DZone (https://dzone.com/articles/receive-notifications-with-new-values-when-table-r)
[13] FoxLearn (https://foxlearn.com/windows-forms/monitoring-data-change-using-sqldependency-in-csharp-377.html#google_vignette)
[14] MSSQLTips (https://www.mssqltips.com/sqlservertip/1836/sql-server-service-broker-example-on-how-to-configure-send-and-receive-messages/)