Универзитет у Београду

Факултет oрганизационих наукa

Лабораторија за софтверско инжењерство

Завршни рад

Тема: Развој софтвера за едукативни центар применом .NET технологија.

|  |  |
| --- | --- |
| Ментор:  др Саша Лазаревић | Студент:  Ђорђе Петровић 284/12 |
|  |  |

Београд, јул 2019.

Sadržaj

[1. Uvod 10](#_Toc26814476)

[2. Opis Tehnologija 11](#_Toc26814477)

[2.1 C# programski jezik 11](#_Toc26814478)

[2.1.1 Istorija C# programskog jezika 11](#_Toc26814479)

[2.1.2 .NET platforma 13](#_Toc26814480)

[Od izvornog koda C# do izvršavanja mašine 13](#_Toc26814481)

[2.1.3 Osnovni koncepti objektno orijentisanog programiranja 14](#_Toc26814482)

[Principi OOP-a 14](#_Toc26814483)

[Objektno orijentisano programiranje 15](#_Toc26814484)

[Kritika OOP-a 15](#_Toc26814485)

[2.1.4 Prednosti I mane C# programskog jezika 16](#_Toc26814486)

[Prednosti 16](#_Toc26814487)

[Nedostaci 16](#_Toc26814488)

[2.2 Konkurentno i mrežno programiranje 16](#_Toc26814489)

[2.2.1 Konkurentno programiranje 17](#_Toc26814490)

[2.2.2 Mrežno programiranje 19](#_Toc26814491)

[2.3 Softverski paterni 21](#_Toc26814492)

[2.3.1 Definicija paterna 21](#_Toc26814493)

[2.3.2 Podela paterna 22](#_Toc26814494)

[2.3.3 Bridge patern 22](#_Toc26814495)

[2.3.4 Singleton patern 23](#_Toc26814496)

[2.3.5 Template method 23](#_Toc26814497)

[2.4 Rad sa bazom podataka 24](#_Toc26814498)

[2.4.1 Rad sa bazom podataka 25](#_Toc26814499)

[3. Studijski Primer 27](#_Toc26814500)

[3.1 Prikupljanje zahteva od korisnika 28](#_Toc26814501)

[3.1.2 Verbalni opis 28](#_Toc26814502)

[3.1.2 Slučajevi korišćenja 30](#_Toc26814503)

[SK1: Slučaj korišćenja – Ubacivanje novog člana 31](#_Toc26814504)

[SK 2: Slučaj korišćenja – Ubacivanje novog kursa 32](#_Toc26814505)

[SK 3: Slučaj korišćenja – Izmena kursa 33](#_Toc26814506)

[SK 4: Slučaj korišćenja – Pregled kurseva 33](#_Toc26814507)

[SK 5: Slučaj korišćenja – Brisanje kursa 34](#_Toc26814508)

[SK 6: Slučaj korišćenja – Unos prisustva 35](#_Toc26814509)

[3.2 Analiza 37](#_Toc26814510)

[3.2.1 Ponašanje softverskog sistema - Dijagram sekvenci slučaja poseta 37](#_Toc26814511)

[DS 1: Dijagram sekvenci slučaja korišćenja – Ubacivanje novog člana 37](#_Toc26814512)

[DSK 2: Dijagram sekvenci slučajeva korišćenja – Ubacivanje novog kursa 38](#_Toc26814513)

[DS3: Dijagram sekvenici slučaja korišćenja – Izmena kursa 40](#_Toc26814514)

[DS 4: Dijagram sekvenci slučaja korišćenja – Pregled kurseva 42](#_Toc26814515)

[DS5: Dijagram sekvenci slučaja korišćenja – Brisanje kursa 44](#_Toc26814516)

[DS 6: Dijagram sekvenci slučaja korišćenja – Unos posete 46](#_Toc26814517)

[3.2.2 Ponasanje softerskog sistema – Definisanje ugovora (Contracts) o sistemskim operacijama 48](#_Toc26814518)

[3.2.3 Struktura softverskog sistema 50](#_Toc26814519)

[3.2 Projektovanje 54](#_Toc26814522)

[3.3.1 Arhitektura softverskog sistema 54](#_Toc26814523)

[3.3.2 Projektovanje korisničkog interfejsa 55](#_Toc26814524)

[3.3.2.1 Projektovanje kontrolera korisničkog interfejsa 55](#_Toc26814525)

[3.3.3 Projektovanje ekranskih formi 56](#_Toc26814526)

[SK1: Slučaj korišćenja – Ubacivanje novog člana 58](#_Toc26814527)

[SK 2: Slučaj korišćenja – Ubacivanje novog kursa 59](#_Toc26814528)

[SK 3: Slučaj korišćenja – Izmena kursa 61](#_Toc26814529)

[SK 4: Slučaj korišćenja – Pregled kurseva 65](#_Toc26814530)

[SK 6: Slučaj korišćenja – Unos prisustva 67](#_Toc26814531)

[SK 14: Slučaj korišćenja – Prikaz izveštaja 69](#_Toc26814533)

[3.3.3 Projektovanje aplikacione logike – Kontroler 71](#_Toc26814534)

[3.4. Implementacija 78](#_Toc26814535)

[3.4.1 Sloj baze podataka 82](#_Toc26814536)

[3.4.2 Srednji sloj 88](#_Toc26814541)

[3.4.3 Prezentacioni sloj 92](#_Toc26814542)

[SK1: Slučaj korišćenja – Ubacivanje novog člana 94](#_Toc26814543)

[SK 2: Slučaj korišćenja – Ubacivanje novog kursa 98](#_Toc26814544)

[SK 3: Slučaj korišćenja – Izmena kursa 102](#_Toc26814545)

[SK 4: Slučaj korišćenja – Pregled kurseva 108](#_Toc26814546)

[SK 6: Slučaj korišćenja – Unos prisustva 111](#_Toc26814547)

[SK 14: Slučaj korišćenja – Prikaz izveštaja 118](#_Toc26814550)

[4. Testiranje 121](#_Toc26814551)

[5. Zaključak 121](#_Toc26814552)

[6. Literatura 122](#_Toc26814553)

**Prikaz slika:**

[Slika 1 Nedrziva struktura 20](#_Toc26814663)

[Slika 2 Odrziva struktura 20](#_Toc26814664)

[Slika 3 Prikaz tronivovske arhitekture 25](#_Toc26814665)

[Slika 4 Faze razvoja softverskog sistema 26](#_Toc26814666)

[Slika 5 Slucajevi koriscenja 27](#_Toc26814667)

[Slika 6 DS1 – Osnovni scenario 36](#_Toc26814668)

[Slika 7 DS1 – Alternativni scenario 36](#_Toc26814669)

[Slika 8 DS2 – alternativni scenario 38](#_Toc26814670)

[Slika 9 DS3 – osnovni scenario 39](#_Toc26814671)

[Slika 10 DS3 – alternativni scenario 39](#_Toc26814672)

[Slika 11 DS3 – alternativni scenario 2 40](#_Toc26814673)

[Slika 12 DS4 – osnovni scenario 41](#_Toc26814674)

[Slika 13 DS4 – alternativni scenario 41](#_Toc26814675)

[Slika 14 DS5 – osnovni scenario 42](#_Toc26814676)

[Slika 15 DS5 – alternativni scenario 43](#_Toc26814677)

[Slika 16 DS5 – alternativni scenario 2 44](#_Toc26814678)

[Slika 17 DS6 – osnovni scenario 45](#_Toc26814679)

[Slika 18 DS6- alternativni scenario 46](#_Toc26814680)

[Slika 19 Konceptualni (Domenski) model 48](#_Toc26814681)

[Slika 20 klasa IOOpstiDomenskiObjekat 1 49](#_Toc26814682)

[Slika 21 Dijagram klasa I domenski objekat 52](#_Toc26814683)

[Slika 22 Troslojna arhitektura – detaljan prikaz 53](#_Toc26814684)

[Slika 23 Prikaz veze korisnika sa sistemom 54](#_Toc26814685)

[Slika 24 Pokretanje servera 54](#_Toc26814686)

[Slika 25 Login korisnika 55](#_Toc26814687)

[Slika 26 Glavna forma 55](#_Toc26814688)

[Slika 27 Unos clana 56](#_Toc26814689)

[Slika 28 Obavestenje o uspešnosti operacije 57](#_Toc26814690)

[Slika 29 Obaveštenje o neuspešnosti operacije 57](#_Toc26814691)

[Slika 30 Forma za unos kursa 58](#_Toc26814692)

[Slika 31 Obaveštenje o uspešnosti operacije 58](#_Toc26814693)

[Slika 32 Obaveštenje o neuspešnosti operacije 59](#_Toc26814694)

[Slika 33 Forma za rad sa kursevima 60](#_Toc26814695)

[Slika 34 Forma za rad sa kursevima 2 61](#_Toc26814696)

[Slika 35 Detalji kursa forma 61](#_Toc26814697)

[Slika 36 Obaveštenje o uspešnosti operacije 62](#_Toc26814698)

[Slika 37 Obaveštenje o neuspešnosti operacije 1 62](#_Toc26814699)

[Slika 38 Obaveštenje o neuspešnosti operacije 2 63](#_Toc26814700)

[Slika 39 Forma za rad sa kursevima 3 64](#_Toc26814701)

[Slika 40 Forma za rad sa kursevima 4 64](#_Toc26814702)

[Slika 41 Obaveštenje o neuspešnosti operacije 65](#_Toc26814703)

[Slika 42 Forma za rad sa prisustvima SK 6 66](#_Toc26814704)

[Slika 43 Obaveštenje o uspešnosti operacije SK6 66](#_Toc26814705)

[Slika 44 Obaveštenje o neuspešnosti operacije SK6 67](#_Toc26814706)

[Slika 45 Prikaz izveštaja 68](#_Toc26814707)

[Slika 46 Obaveštenje o neuspešnosti operacije SK 14 69](#_Toc26814708)

[Slika 47 Sistemske Operacije 70](#_Toc26814709)

[Slika 48 Dijagram klasa 71](#_Toc26814710)

[Slika 49 Dijagram sekvenci - UnosClana 72](#_Toc26814711)

[Slika 50 Dijagram sekvenci - PrikaziClanove 72](#_Toc26814712)

[Slika 51 Dijagram sekvenci - ZapamtiPosete 73](#_Toc26814713)

[Slika 52 Dijagram sekvenci - IzmeniPosetu 73](#_Toc26814714)

[Slika 53 Dijagram sekvenci - ObrisPosetu 74](#_Toc26814715)

[Slika 54 Dijagram sekvenci - VratiPosete 74](#_Toc26814716)

[Slika 55 Konacna arhitektura sistema 75](#_Toc26814717)

[Slika 56 Scripta baze 1 80](#_Toc26814718)

[Slika 57 Clan skripta 81](#_Toc26814719)

[Slika 58 Kurs skripa 82](#_Toc26814720)

[Slika 59 Prisustvo skripta 83](#_Toc26814721)

[Slika 60 Ucionica skripta 83](#_Toc26814722)

[Slika 61 Skripta osoba 85](#_Toc26814723)

[Slika 62 Veza između DBBrokera i OpštegDomenskogObjekta 87](#_Toc26814724)

[Slika 63 Pokretanje servera 90](#_Toc26814725)

[Slika 64 Login korisnika 91](#_Toc26814726)

[Slika 65 Glavna forma 91](#_Toc26814727)

[Slika 66 Unos clana 92](#_Toc26814728)

[Slika 67 Obavestenje o uspešnosti operacije 93](#_Toc26814729)

[Slika 68 Obaveštenje o neuspešnosti operacije 93](#_Toc26814730)

[Slika 69 Forma za unos kursa 96](#_Toc26814731)

[Slika 70 Obaveštenje o uspešnosti operacije 97](#_Toc26814732)

[Slika 71 Obaveštenje o neuspešnosti operacije 97](#_Toc26814733)

[Slika 72 Forma za rad sa kursevima 101](#_Toc26814734)

[Slika 73 Forma za rad sa kursevima 2 102](#_Toc26814735)

[Slika 74 Detalji kursa forma 102](#_Toc26814736)

[Slika 75 Obaveštenje o uspešnosti operacije 103](#_Toc26814737)

[Slika 76 Obaveštenje o neuspešnosti operacije 1 103](#_Toc26814738)

[Slika 77 Obaveštenje o neuspešnosti operacije 2 103](#_Toc26814739)

[Slika 78 Forma za rad sa kursevima 3 106](#_Toc26814740)

[Slika 79 Forma za rad sa kursevima 4 107](#_Toc26814741)

[Slika 80 Obaveštenje o neuspešnosti operacije 107](#_Toc26814742)

[Slika 81 Forma za rad sa prisustvima SK 6 110](#_Toc26814743)

[Slika 82 Obaveštenje o uspešnosti operacije SK6 110](#_Toc26814744)

[Slika 83 Obaveštenje o neuspešnosti operacije SK6 111](#_Toc26814745)

[Slika 84 Prikaz izveštaja 117](#_Toc26814746)

[Slika 85 Obaveštenje o neuspešnosti operacije SK 14 118](#_Toc26814747)

**Prikaz** **tabela**:

[Tabela 1 Strukturna i vrednosna ograničenja za tabelu Clan 50](#_Toc24898813)

[Tabela 2 Strukturna i vrednosna ograničenja za tabelu kurs 51](#_Toc24898814)

[Tabela 3 Strukturna i vrednosna ograničenja za tabelu TipKursa 51](#_Toc24898815)

[Tabela 4 Strukturna i vrednosna ograničenja za tabelu Prisustvo 51](#_Toc24898816)

[Tabela 5 Strukturna i vrednosna ograničenja za tabelu Ucionica 52](#_Toc24898817)

[Tabela 6 Strukturna i vrednosna ograničenja za tabelu Korisnik 52](#_Toc24898818)

[Tabela 7 Strukturna i vrednosna ograničen ja za tabelu osoba 52](#_Toc24898819)

[Tabela 8 Clan 82](#_Toc24898820)

[Tabela 9 Kurs 83](#_Toc24898821)

[Tabela 10 Prisustvo 84](#_Toc24898822)

[Tabela 11 Ucionica 85](#_Toc24898823)

[Tabela 12 TipKursa 86](#_Toc24898824)

[Tabela 13 Osoba 86](#_Toc24898825)

[Tabela 14 Korisnik 87](#_Toc24898826)

# Uvod

C# je višenamenski jezik koji je Microsoft implementiran 2000. odine u okviru .NET inicijative koji je kasnije odobren kao standard od strane Emca i SO-a. Dizajnirao ga je Anders Heljsberg i tima developera koji je predvodio Mads Torgersen. [3]

C# je jezik čija je ideja da bude višenamenski, jednostavan i moderan. Programeri koji su ga napisali su želeli da bude pogodan za razvijanje softvera koji bi bio pogodan za pisanje aplikacija za *host* embeded sisteme. Iako je bila namena da aplikacije koje budu napisane u C# jeziku budu ekonomičnije u pogledu zauzeća resursa i cene, ipak ne može da se uporedi sa veličinom I perfomansama C programskog jezika.

U drugom poglavlju ovog rada detaljnije se govori o programskom jeziku C#, konceptima koji karakterišu objektno-orijentisano programiranje kao I prednostima I nedostacima C# kao programskog jezika. Objašnjeni su I koncepti konkurentnog programiranja, mrežnog programiranja, kao I paterni koji su korišćeni u studijskom primeru. Na kraju je pojašnjena veza sa bazom podataka I kreiranje grafičkog interfejsa.

U trećem poglavlju prikazani su svi koncepti opisani u poglavlju dva kroz primer administracije šahovskog kluba. Aplikacija je kreirana korišćenjem Larmanove metode za razvoj softvera.

# Opis Tehnologija

U ovom poglavlju biće opisan C# programski jezik, osnovni koncepti istog kao i prednosti I nedostaci C#-a. Takođe će se opisati konkurentno I mrežno programiranje, tačnije rad sa nitima I soketima, kao i paterna kao vrlo važnog koncepta u programiranju. Na kraju će biti reči o skladištima podataka I njihovoj vezi sa programskim jezikom kao i korisnički grafički interfejs koji omogućava interakciju korisnika sa sistemom.

## 2.1 C# programski jezik

C# je elegantan i tipski siguran objektno orijentisani jezik koji omogućava programerima da izgrade razne sigurne i robusne aplikacije koje rade na platform .NET framework-a. C3 se koristi za kreiranje Windows desktop aplikacija, XSML Web servisa i klijent-server aplikacija. Visual C# pruža sadrži alate kao što su garbage kolektor, biblioteke za rad sa soketima, nitima i mnoge druge alate koji olakšavaju razvoj aplikacija zasnovanih na jeziku C# i .NET Framework-u.

## 2.1.1 Istorija C# programskog jezika

Tokom razvoja .NET Framework-a, biblioteke klasa su prvobitno napisane korišćenjem sistema kompajlera upravljanog koda pod nazivom Simple Managed C (SMC). U januaru 1999, Anders Hejlsberg formirao je tim za izgradnju novog jezika zvanim Cool, koje je predstavljalo "C-like Object Oriented Language". Microsoft je razmišljao o tome da ime "Cool" zadrži kao konačni naziv jezika, ali je odlučio da to ne učini zbog žigova. Do vremena kada je .NET projekat bio javno objavljen na konferenciji za profesionalne programere u julu 2000. godine, jezik je preimenovan u C#, a biblioteke klasa i ASP.NET runtime su prevedene na C#.

Hejlsberg je C# glavni dizajner i vodeći arhitekta u Microsoftu, a prethodno je bio uključen u dizajn Turbo Pascala, Embarcadero Delphi (ranije CodeGear Delphi, Inprise Delphi i Borland Delphi) i Visual J++. U intervjuima i tehničkim dokumentima on je izjavio da su mane u većini glavnih programskih jezika (npr. C ++, Java, Delphi i Smalltalk) pokrenule osnove Common Language Runtime (CLR), što je, pak, dovelo do dizajna samog C # jezika.

James Gosling, koji je 1994. godine kreirao Java programski jezik, i Bill Joi, suosnivač Sun Microsistems, pokretača Jave, nazvao je C# "imitacijom" Jave. Gosling je dalje rekao da je C # vrsta Jave sa pouzdanošću, produktivnošću i sigurnošću. Klaus Kreft i Angelika Langer (autori knjige sa C ++ streamovima) su na blogu naveli da su Java i C # Gotovo identični programski jezici, dosadno ponavljanje koje nema inovaciju. Teško da bilo ko može da tvrdi da su Java ili C # revolucionarni programski jezici koji su promenili način na koji pišemo programe, C # pozajmio mnogo od Java - i obrnuto. U julu 2000, Hejlsberg je rekao da C # nije Java klon i da je mnogo bliži C++ jeziku.

Od izdavanja C # 2.0 u novembru 2005, C # i Java su evoluirali na sve divergentnijim putanjama, postajući dva veoma različita jezika. Jedna od prvih velikih promena došla je sa dodatkom generičkih izraza na oba jezika, sa znatno drugačijim implementacijama. C # koristi reifikaciju da bi obezbedio "prvoklasne" generičke objekte koji se mogu koristiti kao bilo koja druga klasa, sa generisanjem koda u vremenu učitavanja klase. Pored toga, C # je dodao nekoliko glavnih karakteristika za prilagođavanje funkcionalnog programiranja, što je kulminiralo sa LINK ekstenzijama koje su objavljene sa C # 3.0 i njegovim podržavajućim okvirom za lambda izraze, metode proširenja i anonimne tipove. Ove funkcije omogućavaju programerima C#-a da koriste funkcionalne tehnike programiranja, kao što su zatvarači, kada je to korisno za njihovu primenu. LINK ekstenzije i funkcionalni uvoz pomažu programerima da smanje količinu predloženih kodova koji su uključeni u uobičajene zadatke, kao što su upiti u bazu podataka, parsiranje xml datoteke ili pretraživanje kroz strukturu podataka, prebacujući naglasak na stvarnu programsku logiku kako bi poboljšali čitljivost i održivost.

C# je imao maskotu zvanu Andi (nazvan po Anders Hejlsberg). Penzionisan je 29. januara 2004.

C# je prvobitno podnesen pododboru ISO JTC 1 / SC 22 na pregled, prema ISO / IEC 23270: 2003 je povučen i zatim je odobren u skladu sa ISO / IEC 23270: 2006.

## 2.1.2 .NET platforma

C# programi mogu da se izvršavaju na .NET Framework-u. Integralnakomponenta Windows-a koja uključuje virtuelni izvršni sistem koji se zove *runtime* (Common Language Runtime) zajedničkog jezika i objedinjeni skup biblioteka klasa. CLR je komercijalna implementacija od strane Microsoftove zajedničke jezične infrastrukture (Common Language Infrastructure), međunarodnog standarda koji je osnova za stvaranje izvršnog i razvojnog okruženja u kojima jezici i biblioteke rade bez problema.[2]

Izvorni kod napisan u C# kompiliran je u *intermediate language* (IL) koji je u skladu sa CLI specifikacijom. IL kod i resursi, kao što su bitmape i nizovi, skladište se na disku u izvršnoj datoteci, obično sa ekstenzijom „.exe“ ili „.dll“. Sklop sadrži manifest koji pruža informacije o tipovima sklopova, verziji, kulturi i sigurnosnim zahtevima.

Kada se program C# izvrši, sklop se učitava u CLR, što može da preduzme različite akcije na osnovu informacija u manifestu. Zatim, ako su zadovoljeni sigurnosni zahtevi, CLR izvodi samo na vreme (*just-in-time*) kompilaciju za pretvaranje IL koda u izvorne strojne upute. CLR takođe pruža i druge usluge koje se odnose na automatsko prikupljanje smeća, rukovanje izuzecima i upravljanje resursima. Kod koji se izvršava od strane CLR-a ponekad se naziva "upravljački kod", za razliku od "neupravljivog koda" koji je kompajliran u mašinski jezik određenog sistema. Na sledećem dijagramu prikazani su odnosi vremena kompajliranja i vremena izvršavanja datoteka izvornog koda C#, biblioteka, skupova i CLR klasa .NET Framework-a.

### Od izvornog koda C# do izvršavanja mašine

Jezička interoperabilnost je ključna karakteristika .NET Framework-a. Zbog toga što je IL kod koji je kreirao C# kompajler u skladu sa Common Tipe Specification (CTS), IL kod generisan od C# može da interaktuje sa kodom koji je generisan iz .NET verzija Visual Basic-a, Visual C++-a ili bilo kog od više od 20 drugih CTS-compliant jezicima. Jedan skup može sadržati više modula napisanih na različitim .NET jezicima, a tipovi se mogu međusobno pozivati kao da su napisani na istom jeziku.

Pored usluga za vreme izvođenja, .NET Framework takođe uključuje obimnu biblioteku od preko 4000 klasa organizovanih u prostorima imena koji pružaju široku lepezu korisnih funkcija za sve, od unošenja datoteka i izlaza do manipulacije stringovima do XSML parsiranja, do Windows Forms kontrola. Tipična C# aplikacija ekstenzivno koristi biblioteku klasa .NET Framework-a da bi obradila uobičajene "plumbing" poslovi kao što su „commit“ i „rollback“ transakcija, obrada grešaka pomoću try-catch blokova i korišćenje connect() metode.

## 2.1.3 Osnovni koncepti objektno orijentisanog programiranja

Objektno orijentisano programiranje (OOP) je model programskog jezika u kojem su programi organizovani oko podataka ili objekata, a ne funkcija i logika.[2] Objekat se može definisati kao polje podataka koje ima jedinstvene atribute i ponašanje. Primeri objekta mogu biti u rasponu od fizičkih entiteta, kao što je ljudsko biće koje je opisano svojstvima poput imena i adrese, sve do malih programa, kao što su widget-i. Ovo se suprotstavlja istorijskom pristupu programiranju gde je naglasak stavljen na to kako je logika napisana, a ne kako definisati podatke unutar logike.

Prvi korak u OOP-u je da identifikuje sve objekte koje programer želi da manipuliše i kako se oni odnose jedni prema drugima, što je često poznato kao modeliranje podataka. Jednom kada je objekat poznat, on je generalizovan kao klasa objekata koja definiše vrstu podataka koje sadrži i bilo koje logičke sekvence koje mogu da manipulišu njime. Svaka odvojena logička sekvenca je poznata kao metoda, a objekti mogu da komuniciraju sa dobro definisanim interfejsima koji se nazivaju poruke.

Jednostavno rečeno, OOP se fokusira na objekte kojima programeri žele da manipulišu, a ne na logiku potrebnu za manipulaciju njima. Ovaj pristup programiranju je veoma pogodan za programe koji su veliki, kompleksni i aktivno se ažuriraju ili održavaju. Zbog organizacije objektno orijentisanog programa, ova metoda je takođe pogodna za kolaborativni razvoj gde se projekti mogu podeliti u grupe. Dodatne prednosti OOP-a uključuju ponovnu upotrebu, skalabilnost i efikasnost kodova.

### Principi OOP-a

Objektno orijentisano programiranje se zasniva na sledećim principima:

**Enkapsulacija** - Implementacija i stanje svakog objekta se privatno drže unutar definisane granice ili klase. Drugi objekti nemaju pristup atributima i implementaciji metoda, ali mogu samo pozivati potpis javnih funkcija ili metoda. Ova karakteristika skrivanja podataka obezbeđuje veću bezbednost programa i izbegava neželjenu korupciju podataka.

**Apstrakcija** - Objekti otkrivaju samo interne mehanizme koji su relevantni za upotrebu drugih objekata, skrivajući bilo koji nepotrebni kod implementacije. Ovaj koncept pomaže programerima da lakše izmene i dopune tokom vremena.

**Nasleđivanje** - Odnosi i potklasa između objekata mogu biti dodeljeni, dozvoljavajući programerima da ponovo koriste zajedničku logiku, a da pritom zadrže jedinstvenu hijerarhiju. Ovo svojstvo OOP-a dovodi do detaljnije analize podataka, smanjuje vreme razvoja i osigurava viši nivo tačnosti.

**Polimorfizam** - Objektima je dozvoljeno da preuzmu više od jednog oblika u zavisnosti od kontekstaTo znači da ako bi napravili abstraktnu klasu Osoba, klase kao što su Učitelj, Pilot, Građevinski radnik su direktno ili indirektno potklase Osoba. One nasleđuju atribute i ponašanje klase Osoba i sadrže svoje atribute i određeno ponašanje.Program se na ovaj način izvršava brže jer nema dupliciranja koda.[3]

### Objektno orijentisano programiranje

Iako je Simula priznat kao prvi objektno-orijentisani programski jezik, najpopularniji OOP jezici su:

* Java
* JavaScript
* Python
* C++
* Visual Basic .NET
* Rubi
* Scala
* PHP

OOPSLA je godišnja konferencija za objektno-orijentisane sistemske sisteme, jezike i aplikacije.

### Kritika OOP-a

Ideju o objektno orijentisanom programiranju programeri su kritikovali iz više razloga. Najveća briga je da OOP prenaglašava komponentu podataka razvoja softvera i ne fokusira se dovoljno na računanje ili algoritme. Pored toga, OOP kod može biti komplikovaniji za pisanje i potrebno je više vremena za kompajliranje. Alternativne metode za OOP obuhvataju funkcionalno programiranje, strukturirano programiranje i imperativno programiranje, ali najnapredniji programski jezici daju programerima mogućnost da ih kombinuju.[4]

## 2.1.4 Prednosti I mane C# programskog jezika

Kao i svaki programski jezik, C# ima svoje prednosti i mane koje ćemo opisati u ovom poglavlju.

### Prednosti

1. C # je objektno orijentisan, ali je C++ mešavina objektno orijentiranog i proceduralno orijentisanog.

2. C # je više bezbedan za tip

3. Ne postoje problemi kao što je curenje memorije jer ima garbage kolektor, što je problem za C ++ programere.

4. Koncept Asemblija dobro rešava problem kontrole nad verzijama.

5.Ease-to-development, bogata biblioteka klasa čini mnoge funkcije lakim za implementaciju.

6. Dobra podrška za distribuirane sistema.

### 

### Nedostaci

1. Ne može da radi na niskim nivoima kao što je povezivanje direktno sa hardverom preko upravljačkih programa i firmvera.
2. Glavni nedostatak ovog jezika bilo bi vreme potrebno za savladavanje i učenje .NET-a kao jezičnog koncepta. Jedna stvar koja otežava učenje su nove funkcionalnosti koje se razvijaju svake godine.

# 2.2 Konkurentno i mrežno programiranje

Uovom poglavlju biće opisane neke od glavnih karakteristika mrežnog i konkurentnog

programiranja. Kroz ovo poglavlje bi trebalo ili će se pomenuti koncepti niti, soketa, kao

i njihova primena u klijent-server aplikacijama.

## 2.2.1 Konkurentno programiranje

Konkurentno programiranje resave problem paralelnog izvrsavanja operacija. Neke od prednosti

konkurentnosti su minimizacija vremena čekanja na odgovor, maksimizacija protoka,

modelovanje, paralelizam pri izvršavanju, iskorišćavanje multiprocesorskog

okruženja, izolovanje aktivnosti u okviru niti. Naravno, postoje i mane, a to su

kompleksnost, sigurnosni problemi, preopterećenje, visoko korišćenje

resursa, mogućnosti potpunog zastoja. Najčešća primena konkurentnog

programiranja je kada više korisnika mora imati pristup bazi podataka, nekom

resursu, veb stranicama...

Za realizaciju konkurentog programiranja koriste se 3 načina. To su:

**Višenitni sistemi** – višenitni softverski sistemi omogućavaju korišćenje više

aplikacija najednom računaru i karakteristični su za jednoprocesorske

računare.

**Distribuirani sistemi** – obezbeđuju rad sa udaljenim podacima. To su sistemi

koji se sastojeod većeg broja računara koji su povezani u lokalnu ili globalnu

računarsku mrežu.

**Paralelni sistemi**– predstavljaju računare koji omogućavaju podelu podataka

ili zadataka između više svojih komponenti, kompleksnost, sigurnosni problemi, preopterećenje, visoko korišćenje resursa, mogućnosti potpunog zastoja. Najčešća primena konkurentnog programiranja je kada više korisnika mora imati pristup bazi podataka, nekom resursu, veb stranicama...

#### 2.2.1.1 Niti

Niti rešavaju problem multitaskinga. To su aktivnosti koje se istovremeno izvršavaju

u okviru jednog procesa. Niti nisu program, već se one izvršavaju u okviru programa.

Multitasking je metoda pomoću koje više zadataka (eng. *tasks*), odnosno procesa,

dele zajedničke resurse kao što je CPU (eng. *Central Processing Unit*). U slučaju

računara sa jednim CPU, u nekom trenutku vremena samo jedan zadatak može da se

izvršava, što praktično znači da CPU aktivno izvršava samoinstrukcije tog zadatka.

Multitasking omogućava da se u nekom periodu vremena više procesa izvršava, tako

što pravi raspored (eng. *scheduling*) koji zadatak će se izvršavati u kom trenutku

vremena a koji zadaci će čekati dok na njih ne dođe red. [5]

Glavni problemi kod korišćenja niti su potpuno zaključavanje (eng. *deadlock*) i

gladovanje (eng. *starvation*).

**Deadlock** je multitasking problem koji se dešava kada dva procesa zauzmu resurse i

međusobno se čekaju da oslobode te resurse.

**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. [5]

Pri svakom pokretanju Java programa automatski se pokreće glavna programska

nit. Ona služi da obradi sve potrebe i zahteve programa tokom njegovog izvršenja.

#### 2.2.1.2 Stanja niti

O tome šta niti trenutno rade govore nam stanja niti. Nit može biti u 4 stanja:

* Novo (eng. *new*)
* Izvršno (eng. *runnable*)
* Blokirano (eng. *blocked*)
* Svršeno (eng. dead)

Nit se može definisati na sledeći način:

Thread nit = new Thread(this);

Prilikom kreiranja niti ona se nalazi u stanju **novo**. Nit se nalazi u tom stanju sve dok se ne pozove metoda *start()*.

Pozivom metode *start()*, nit prelazi u stanje **izvršno**. Operativni sistem obezbeđuje procesorsko vreme za izvršavanje niti. Svaka nit nekog programa dobija procesorsko vreme. Redosled dobijanja procesorskog vremena nam nikada nije poznat. Vreme koje dobija svaka nit je približno jednako.

ThreadPool.QueueUserWorkItem(ShowThreadInformation);

var th1 = new Thread(ShowThreadInformation);

th1.Start();

var th2 = new Thread(ShowThreadInformation);

th2.IsBackground = true;

th2.Start();

Thread.Sleep(500);

ShowThreadInformation(null);

Nit može preći u **blokirano** stanje u jednom od ovih slučajeva:

* Pozove se metoda ***sleep***
* Pozivom metode ***suspend***
* Kada je nit već blokirana nekom drugom akcijom
* Kada je neki resurs zaključan od strane nekog ulazno izlaznog uređaja

**Svršeno** stanje je stanje niti kada se nit završila.

#### 2.2.1.3 Prekidanje niti

Nit se može prekinuti pozivanjem metode interrupt() ili jednostavnim završenjem

operacije za koju je ona bila zadužena(završenje metode). Kada želimo da prekinemo nit logički jednostavno postavimo atribut kraj na true vrednost.

Kada računar ima jedan procesor, izvršava se jedna nit u svakom trenutku ali tako da

izgleda konkurentno. Izvršavanje više niti na jednom procesoru po nekom redosledu naziva se raspoređivanje (engleski: scheduling).

Kadaje više niti spremno za izvršavanje sistem bira onu nit sa najvišim prioritetom za izvršavanje. Samo kada je nit stopirana ili nit nije u izvršnom stanju počinje da se izvršava nit manjeg prioriteta. [5]

#### 2.2.1.4 Sinhronizacija niti

Često se dešava da niti koriste iste resurse nad kojima upisuju neke podatke ili čitaju neke podatke. Da ne bi došlo do oštećenja datog resursa, moraju se date niti sinhronizovati. Sinhronizacija omogućava da jedna nit u jednom trenutku pristupi оbjektu.

Sinhronizacija obezbeđuje uzajamnoisključivanje niti nad zajedničkim objektom. [5]

Postoje metode unutar klase Object koje služe za rad sa sinhronizovanim nitima i to

su :

**wait**() – blokira tekuću nit i oslobađa zauzeti objekat sve dok se ne pozove

metodanotify() ili notifyAll() za objekat kome pripada metoda wait().

**notify**() – poziva nit koja je prva pozvala metodu wait() da nastavi sa radom.

**notifyAll**() – restartuje sve niti koje su pozvale metodu wait().

### 

### 2.2.2 Mrežno programiranje

U ovom poglavlju biće objašnjeno kako se koriste soketi i serverski soketi, kao i neki osnovni koncepti mrežnog programiranja kao i komunikacija preko TCP/IP protokola.

Mrežno programiranje omogućava da se dva ili više programa povežu i da

razmenjuju poruke. Programi mogu biti na istom ili različitim računarima. Samo

povezivanje se vrši preko soketa. Pravi se uređena dvojka i povezuju se dva

programa tako što jedan implementira serverski soket dok drugi gađa IP adresu

servera i bira port na kojem se server aplikacija nalazi. Kada uspostave konekciju dva soketa mogu komunicirati međusobno.

Klijent/server računarstvo povezuje dva ili više programa gde jedan program pruža

usluge ostalima. Program koji pruža usluge naziva se server, a ostali programi su

klijenti. U ovoj vrsti računarstva se koristi distribuirana obrada, gde se zadaci obrade

raspoređuju između dva ili više programa.

Jedan od tipova ovakve obrade je i obrada u mreži ravnopravnih računara (eng.

*peer2peer*). Pri ovakvom tipu obrade svaki računar može da pristupi svim fajlovima

drugih računara.

#### 2.2.2.1 TCP/IP protokoli

TCP / IP, ili Transmission Control Protocol / Internet Protocol, skup je komunikacijskih protokola koji se koriste za povezivanje mrežnih uređaja na internetu. TCP / IP se može koristiti i kao komunikacioni protokol u privatnoj mreži (intranet ili ekstranet).

Kompletan paket internet protokola - skup pravila i procedura - obično se naziva TCP / IP, iako su drugi uključeni u paket.

TCP / IP specificira kako se podaci razmenjuju preko interneta tako što se obezbeđuju komunikacije od kraja do kraja koje identifikuju kako bi trebalo razbiti u pakete, adresirati, preneti, preusmeriti i primiti na odredište. TCP / IP zahteva malo centralnog upravljanja i dizajniran je da učini mreže pouzdanim, uz mogućnost automatskog oporavka od kvara bilo kog uređaja na mreži.

Dva glavna protokola u skupu internet protokola služe specifičnim funkcijama. TCP definiše kako aplikacije mogu da kreiraju kanale komunikacije preko mreže. On, takođe, upravlja načinom na koji je poruka sastavljena u manje pakete pre nego što se zatim prenose preko interneta i ponovo sastavljaju u pravom redosledu na adresi odredišta.

IP definiše kako adresirati i usmeriti svaki paket kako bi se uverio da je dostigao pravo odredište. Svaki računar mrežnog prolaza na mreži proverava ovu IP adresu da bi odredio gde da prosledi poruku.

TCP / IP funkcionalnost je podeljena u četiri sloja, od kojih svaki sadrži specifične protokole.

* Aplikacioni sloj obezbeđuje aplikacije sa standardizovanom razmenom podataka. Njegovi protokoli uključuju protokol za prenos hiperteksta (HTTP), protokol za prenos datoteka (FTP), protokol za poštu 3 (POP3), protokol SMTP (Simple Mail Transfer Protocol) i protokol za jednostavno upravljanje mrežom (SNMP).
* Transportni sloj je odgovoran za održavanje komunikacija od kraja do kraja preko mreže. TCP upravlja komunikacijama između domaćina i obezbeđuje kontrolu protoka, multipleksiranje i pouzdanost. Transportni protokoli uključuju TCP i User Datagram Protocol (UDP), koji se ponekad koriste umesto TCP-a za posebne svrhe.
* Mrežni sloj, koji se naziva i internet sloj, bavi se paketima i povezuje nezavisne mreže za prenos paketa preko granica mreže. Protokoli mrežnog sloja su IP i Internet Control Message Protocol (ICMP), koji se koristi za prijavljivanje grešaka.
* Fizički sloj se sastoji od protokola koji rade samo na linku - mrežnoj komponenti koja povezuje čvorove ili hostove u mreži. Protokoli u ovom sloju uključuju Ethernet za lokalne mreže (LAN) i protokol za adresiranje (ARP).

#### 2.2.2.2 Soketi

Socket klasa obezbeđuje bogat skup metoda i svojstava za mrežne komunikacije. Socket klasa omogućava da izvršite i sinhroni i asinhroni prenos podataka koristeći bilo koji od komunikacionih protokola navedenih u nabrajanju ProtocolType.

Klasa Socket sledi šablon za imenovanje .NET Framework-a za asinhrone metode. Na primer, sinhroni prijemni metod odgovara asinhronim metodama BeginReceive i EndReceive.

Ako se koristi protokol povezan sa vezom, kao što je TCP, server može da sluša veze koristeći metod slušanja. Metoda Accept obrađuje sve dolazne zahteve za vezu i vraća Socket koji se može koristiti za komuniciranje podataka s udaljenim hostom. Poziva se Bind metoda pre pozivanja metode Listen ako treba da navede lokalna IP adresa i broj porta. Koristi se broj porta nula da bi servis provajder dodelio slobodan port. Povezivanje sa hostom za slušanje mora da se obavi preko povezivanje metode Connect. Da biste preneli podatke, pozovite način slanja ili primanja. Ako se izvršava više asinhronih operacija na soketu, one ne moraju nužno da se završe redosledom kojim su pokrenute.

# 2.3 Softverski paterni

U ovom poglavlju će se govoriti o softverskim paternima, njihovoj ulozi i primerima u

radu, kao što je *singleton patern*

### 2.3.1 Definicija paterna

Jedna od definicija kaže da su paterni(uzori) zapravo „transformacija strukture

problema u strukturu rešenja. Definicija koju je dao Kristofer Aleksander je:

„Svaki patern opisuje problem koji se stalno ponavlja u našem okruženju i zatim

opisuje suštinu rešenja problema tako što se to rešenje može upotrebiti milion puta, a

da se dva puta neponovi na isti način “. Jedna od definicija kaže da su paterni(uzori) zapravo „transformacija structure problema (neodržive strukture) u strukturu rešenja (održivu strukturu).[6]



Slika Nedrziva struktura



Slika Odrziva struktura

**Klijent** je element strukture paterna koji koristi funkcionalnosti konkretnog servera

kako bi mogaoda obavi sopstvenu funkcionalnost.

**Konkretni server** je element strukture paterna koji klijentu daje konkretnu

funkcionalnost.

### 2.3.2 Podela paterna

Paterni se mogu podeliti na:

* Kreacioni paterni
* Strukturni paterni
* Paterni ponašanja

### 2.3.3 Bridge patern

Bridge patern pripada strukturnim paternima. Ideja ovog paterna je da odvoji

apstrakciju od same implementacije tako da se mogu menjati nezavisno. Na slici 10

je prikazana struktura paterna Bridge.

Učesnici su:

1.**Abstraction**– definiše interfejs apstrakcije i čuva reference na objekat tipa

Implementor

2.**RefinedAbstraction**– proširuje interfejs koji je definisan klasom Abstraction

3.**Implementor**– definiše interfejs za implementacione klase ConcreteImplementorA,

ConcreteImplementorB. Obično Implementor interfejs obezbeĎuje samo primitivne

operacije, a klasa Abstractiondefiniše operacije visokog nivoa koje su zasnovane na

navedenim primitivnim operacijama.

4.**ConcreteImplementor**– implementira interfejs klase Implementor [7]

Na studijskom primeru Bridge patern se primenjuje nad klasom DBBroker i

interfejsom OpstiDomenskiObjekat čija je strukturaprikazana na slici 11.

### 2.3.4 Singleton patern

Singleton patern je jedan od najjednostavnijih i najkorišćenijih paterna trenutno.

Jednostavne je strukture. Instancira se samo jednom i obezbeđuje globalni pristup do

te instance, zabranjujući instanciranje iste klase na nekom drugom mestu.

Učesnici su:

1.**Singleton** – definiše Instance()operacijukoja omogućava klijentima pristup do

njenog jedinstvenog pojavljivanja.

Jedan od primera implementiranih u radu je klasa za komunikaciju sa serverom:

public class Komunikacija{

private static Komunikacija instanca;

private Komunikacija() {

}

public static Komunikacija getInstanca() {

if(instanca == null)

instanca = new Komunikacija();

return instanca;

}

}

### 2.3.5 Template method

Template method je patern ponašanja. Odlikuje ga struktura takva da u nadklasi

postoji metoda koja je generička a da delove te metode koji su specifični zapravo

implementiraju potklase koje nasleđuju tu natklasu. Time je ovaj patern usko vezan

za pojam nasledjivanja.

Metoda TemplateMethod() je generička i ona je nepromenjiv deo algoritma, ali sve

promenjive delove zapravo implementira ConcreteClass.

U studijskom primeru u klasi OpstaSistemskaOperacija može se videti primena ovog

paterna.

public ServerskiTransferObjekat

IzvrsiTransakciju(OpstiDomenskiObjekat odo) {

ServerskiTransferObjekat transferObjekat = new

ServerskiTransferObjekat();

try {

DBBroker.getInstance().ucitajDrajver();

DBBroker.getInstance().otvoriKonekciju();

proveriPreduslov(odo);

transferObjekat = izvrsiOperaciju(odo);

DBBroker.getInstance().commit();

} catch (Exception ex) {

DBBroker.getInstance().rollback();

transferObjekat.setUspesno(false);

transferObjekat.setPoruka(ex.getMessage());

}finally {

try {

DBBroker.getInstance().zatvoriKonekciju();

} catch (SQLException ex) {

transferObjekat.setUspesno(false);

transferObjekat.setPoruka(ex.getMessage());

}

}

return transferObjekat;

}

public abstract ServerskiTransferObjekat

izvrsiOperaciju(OpstiDomenskiObjekat odo) throws Exception;

public abstract void proveriPreduslov(OpstiDomenskiObjekat odo)

throws Exception;

}

## 2.4 Rad sa bazom podataka

**SQL**je upitni jezik za rad sa relacionim bazama podataka. U relacionoj bazi domenski objekti su predstavljeni u vidu tabela, a čuvanje podataka je zasnovano na relacionom modelu. Uovakvom modelu podaci su organizovani u skup relacija između kojih se definišu određene veze.

**Relacija** se definiše kao skup n-torki sa istim atributima, definisanih nad istim domenima iz kojihmogu da uzimaju vrednosti. U relacionim bazama podataka svaka relacija mora da ima definisanprimarni ključ, koji predstavlja atribut pomoću kojeg se jedinstveno identifikuje svaka n-torka. [7]

### 2.4.1 Rad sa bazom podataka

SQL (Structured Query Language) je programski jezik namenjen za upravljanje podacima u relacionim sistemima za upravljanje bazama podataka. Obuhvata unos podataka, upite, ažuriranje i brisanje, šeme kreiranja i menjanja, kao i podatke za kontrolu pristupa.

SQL je najviše korišćen programski jezik za baze podataka. SQL je postao standard Američkog nacionalnog instituta za standarde (ANSI) u 1986. god., kao i Međunarodne organizacije za standarde (ISO) u 1987. god. Od tada standard je poboljšan nekoliko puta sa dodatim funkcijama.

Postoje tri kategorije naredbi u SQL-u:

1. naredbe za definisanje podataka,

2. naredbe za rukovanje podacima i

3. naredbe za upravljačke funkcije

Nas najviše interesuju naredbe za rukovanje podacima jer smo baš te naredbe

najviše koristili kroz rad.

Ove naredbe se dalje dele na:

naredbe za održavanje baze podataka:

* INSERT
* UPDATE
* DELETE

naradbe za izveštavanje

* SELECT

Primer jednog SELECT upita:

Select \* from Korisnik where Sifra = ‘222’

Primer jednog INSERT upita :

Insert into Clan Values (‘Marko’,’Markovic’)

Primer UPDATE upita :

update Clan set Ime = ‘Nikola’ where Ime = ‘Marko’

Primer DELETE upita

Delete from Osoba where Sifra = 232

# Studijski Primer

U ovom poglavlju biće prikazan studijski primer razvoja softvera za administraciju

Edukativnog centra primenom Larmanove metode. Biće opisana Larmanova metoda

kao i njeni koraci.

Larmanova metoda se zasniva na slučajevima korišćenja i jeziku za modelovanje

(eng. *Unified Modeling Language* - **UML**).

Larmanova metoda se sastoji od pet faza:

1) Prikupljanje zahteva od korisnika,

2) Analiza,

3) Projektovanje,

4) Implementacija,

5) Testiranje [8]

U fazi prikupljanja zahteva od korisnika definišu se svojstva i uslovi koje sistem treba

da zadovolji. U fazi analize opisuje se logička struktura softverskog sistema, njegova poslovna logika. Struktura se predstavlja preko konceptualnog i relacionog modela, a

ponašanje preko sistemskih operacija i dijagrama sekvenci.

U fazi projektovanja opisuje se arhitektura sistema, tačnije tronivojska arhitektura

koja se sastoji od korisničkog interfejsa, aplikacione logike i skladišta podataka. Na

slici je prikazana tronivovska arhitektura.



Slika Prikaz tronivovske arhitekture

Na slici 4 su prikazane su faze razvoja softverskog sistema.



Slika Faze razvoja softverskog sistema

## Prikupljanje zahteva od korisnika

**Zahtevi** (eng. *requirements*) predstavljaju svojstva i uslove koje sistem mora da

zadovolji. Zahtevi se kod Larmanove metode opisuju pomoću UML Modela slučajeva

korišćenja. Takav model se sastoji od skupa SK (slučajeva korišćenja), aktora (spoljnih korisnika sistema) i veza izmedju SK i aktora.

**Slučaj korišćenja** opisuje skup scenarija, tj. skup željenih korišćenja sistema od

strane aktora. Slučaj korišćenja (SK) se sastoji od jednog glavnog i jednog ili više

alternativnih scenarija.

Specifikacija korisničkih zahteva podrazumeva opis potreba ili želja koje softverski

proizvod treba da omogući.

### 3.1.2 Verbalni opis

Korisnički zahtev – verbalni opis: Potrebno je napraviti softver koji će pratiti rad jednog edukativnog centra, odnosno prati rad sa članovima (ubacivanje novog, promena postojećeg, brisanje članova), evidentirati njihove posete kao i pregled istih, a takođe se prate i kursevi koji centar pruža i koji članovi mogu da pohađaju. Takođe je omogućeno i prijavljivanje članova na određene kurseve.

Ovaj sistem je namenjen radniku u edukativnom centru i on treba da mu obezbedi efikasno evidentiranje i pretragu zahteva za određenim kursevima, kao i samih članova i poseta.



Slika Slucajevi koriscenja

### 3.1.2 Slučajevi korišćenja

U datom primeru identifikovani su sledeći slučajevi korišćenja:

* SK1: Slučaj korišćenja – Ubacivanje novog člana

Ovaj slučaj opisuje proces unosa novog člana u sistem.

* SK2: Slučaj korišćenja – Ubacivanje novog kursa

Ovaj slučaj opisuje proces unosa novog kursa u sistem.

* SK 3: Slučaj korišćenja – Izmena kursa

Ovaj slučaj opisuje izmenu kursa koji se već nalazi u sistema

* SK 4: Slučaj korišćenja – Pregled kurseva

Ovaj slučaj opisuje pregled svih kurseva koji se nalaze u sistemu. Ova pretraga može da prikaže kurseve koji zadovoljavaju određene parametre koji su definisani od strane korisnika.

* SK 5: Slučaj korišćenja – Brisanje kursa

Ovaj slučaj opisuje brisanje postojećeg kursa iz sistema.

* SK 6: Slučaj korišćenja – Unos prisustva

Ovaj slučaj opisuje unos evidencije prisustva člana nekog kursa koji pohađa.

**SK1: Slučaj korišćenja – Ubacivanje novog člana**

**Naziv SK**:

Ubacivanje novog *člana*

**Akteri SK:**

**Radnik**

**Učesnici SK:**

**Radnik** i **program**(**sistem**)

Preduslov: **Sistem** je uključen. Otvorena je forma za unos *članova*.

**Osnovni scenario SK:**

1. **Radnik** unosi podatke o novom *članu*. (APUSO)
2. **Radnik** poziva sistem da zapamti novog *člana.* (APSO)
3. **Sistem** pamti novog *člana*. (SO)
4. **Sistem** pokazuje radniku poruku: „**Sistem** je uspešno zapamtio *člana* “. (IA)

**Alternativni scenario**:

4.1 Ukoliko **sistem** ne može da zapamti novog *člana*, prikazuje se poruka: „**Sistem** ne može da zapamti *člana* “. (IA)

**SK 2: Slučaj korišćenja – Ubacivanje novog kursa**

**Naziv SK:**

Ubacivanje novog *kursa*

**Akteri SK:**

**Radnik**

**Učesnici SK:**

**Radnik** i **sistem**

**Preduslov**: **Sistem** je uključen. Otvorena je forma za unos *kursa*. Učitani su svi tipovi *kurseva*.

**Osnovni scenario SK:**

1. **Radnik** unosi podatke o novom *kursu*. (APUSO)
2. **Radnik** poziva **sistem** da zapamti novi *kurs*. (APSO)
3. **Sistem** pamti novi *kurs*. (SO)
4. **Sistem** prikazuje **radniku** poruku: „*Kurs* je uspešno zapamćen“. (IA)

**Alternativni scenario:**

* 1. Ukoliko **sistem** ne može da zapamti novi *kurs*, prikazuje se poruka: „**Sistem** ne može da zapamti *kurs*“. (IA)

**SK 3: Slučaj korišćenja – Izmena kursa**

**Naziv SK:**

Izmena postojećeg *kursa*

**Akteri SK:**

**Radnik**

**Učesnici SK:**

**Radnik** i **sistem**

**Preduslov**: **Sistem** je uključen i **radnik** je ulogovan pod svojom šifrom. Otvorena je forma za izmenu *kursa*. Učitani su svi tipovi *kurseva*.

**Osnovni scenario SK:**

1. **Radnik** bira tip *kursa* iz spiska *kurseva*. (APUSO)
2. **Radnik** poziva **sistem** da pronadje kurseve za izabrani tip *kursa*. (APSO)
3. **Sistem** pronalazi *kurseve*. (SO)
4. **Sistem** prikazuje listu *kurseva*. (IA)
5. **Radnik** bira *kurs* iz spiska *kurseva*. (APUSO)
6. **Sistem** prikazuje podatke o izabranom *kursu*. (IA)
7. **Radnik** menja podatke *kursa*. (APUSO)
8. **Radnik** poziva sistem da ažurira *kurs*. (APSO)
9. **Sistem** ažurira *kurs*. (SO)
10. **Sistem** prikazuje poruku da je *kurs* ažuriran. (IA)

**Alternativni scenario:**

* 1. Ukoliko **sistem** ne može da pronađe *kurs*, prikazuje se poruka: „**Sistem** ne može da pronađe *kurs* “. Prekida se izvršenje scenarija. (IA)

8.1 Ukoliko **sistem** ne može da izmeni *kurs*, prikazuje se poruka: „**Sistem** ne može da izmeni *kurs*“. (IA)

**SK 4: Slučaj korišćenja – Pregled kurseva**

**Naziv SK:**

Pregled *kurseva*

**Akteri SK:**

**Radnik**

**Učesnici SK:**

**Radnik** i **sistem**

**Preduslov**: **Sistem** je uključen. Otvorena je forma za pregled *kurseva*. Učitani su svi tipovi *kurseva*.

**Osnovni scenario SK:**

1. **Radnik** bira tip *kursa* iz spiska svih tipova *kurseva*. (APUSO)
2. **Radnik** poziva **sistem** da pronadje *kurseve* za izabrani tip *kursa*. (APSO)
3. **Sistem** pronalazi *kurseve*. (SO)
4. **Sistem** prikazuje listu *kurseva* i poruku: „**Sistem** je uspešno pronašao *kurseve* “. (IA)

**Alternativni scenario:**

* 1. Ukoliko **sistem** ne može da pronađe *kurseve* za određeni tip, prikazuje se poruka: „**Sistem** ne može da pronađe *kurseve* za dati tip “. (IA)

**SK 5: Slučaj korišćenja – Brisanje kursa**

**Naziv SK:**

Brisanje postojećeg *kursa*

**Akteri SK:**

**Radnik**

**Učesnici SK:**

**Radnik** i **sistem**

**Preduslov**: **Sistem** je uključen i **radnik** je ulogovan pod svojom šifrom. Otvorena je forma za brisanje *kurseva*. Učitani su svi tipovi *kurseva*.

**Osnovni scenario SK:**

1. **Radnik** bira tip *kursa* iz spiska *kurseva*. (APUSO)
2. **Radnik** poziva **sistem** da pronadje kurseve za izabrani tip *kursa*. (APSO)
3. **Sistem** pronalazi *kurseve*. (SO)
4. **Sistem** prikazuje listu *kurseva*. (IA)
5. **Radnik** bira *kurs* iz spiska *kurseva*. (APUSO)
6. **Radnik** poziva sistem da izbriše izabrani *kurs*. (APSO)
7. **Sistem** briše *kurs*. (SO)
8. **Sistem** prikazuje poruku: „**Sistem** je uspešno obrisao *kurs*“. (IA)

**Alternativni scenario:**

4.1 Ukoliko **sistem** ne može da pronađe *kurs*, prikazuje se poruka: „**Sistem** ne može da pronađe *kurs*“. Prekida se izvršenje scenarija. (IA)

8.1 Ukoliko **sistem** ne može da izbriše *kurs*, prikazuje se poruka: „**Sistem** ne može da izbriše izabrani *kurs*“. (IA)

**SK 6: Slučaj korišćenja – Unos prisustva**

**Naziv SK:**

Unos novog *prisustva*

**Akteri SK:**

**Radnik**

**Učesnici SK:**

**Radnik** i **sistem**

**Preduslov**: **Sistem** je uključen. Otvorena je forma za ubacivanje nove *posete*. Učitani su svi članovi, tipovi kursa i kursevi.

**Osnovni scenario SK:**

1. **Radnik** unosi podatke o novom *prisustva* člana. (APUSO)
2. **Radnik** poziva **sistem** da zapamti novo *prisustvu*. (APSO)
3. **Sistem** pamti novo *prisustvo*. (SO)
4. **Sistem** prikazuje radniku poruku: „**Sistem** je uspešno zapamtio *prisustvo*“. (IA)

**Alternativni scenario:**

* 1. Ukoliko **sistem** ne može da zapamti novo *prisustvo*, prikazuje se poruka: „**Sistem** ne može da zapamti novo *prisustvo* “. (IA)

## 3.2 Analiza

Nakon faze prikupljanja zahteva, prelazi se na sledeću fazu, fazu analize. Ova faza opisuje logičku strukturu i ponašanje softverskog sistema, tj. poslovnu logiku softverskog sistema.

* Ponašanje opisujemo pomoću:
* sistemskih dijagrama sekvenci
* ugovora o sistemskim operacijama
* Strukturu opisujemo pomoću:
* konceptualnog modela
* relacionog modela

### 3.2.1 Ponašanje softverskog sistema - Dijagram sekvenci slučaja poseta

Ponašanje sistema se može opisati preko UML-ovih **sekvencnih dijagrama** [Larman]**,** odnosno preko **dijagrama saradnje**[6].

**Sistemski dijagram sekvenci** prikazuje, za izdvojeni scenario SK, događaje u određenom redosledu, koji uspostavljaju interakciju izmenu aktora i softverskog sistema.

**Događaj** koji **napravi** aktor je pobuda za poziv sistemske operacije. Preciznije rečeno, događaj koji napravi aktor prihvata primalac događaja koji nakon toga poziva sistemsku operaciju. To znači da aktor ne poziva sistemsku operaciju neposredno već to čini preko posrednika (primaoca događaja). Poziv sistemske operacije ukazuje na interakciju izmenu aktora i sistema. Za događaj koji predstavlja pobudu za poziv SO se često kaže da je to **sistemski događaj.**

**DS 1: Dijagram sekvenci slučaja korišćenja – Ubacivanje novog člana**

**Osnovni scenario SK:**

1. **Radnik** poziva sistem da zapamti novog *člana*(APSO)
2. **Sistem** pokazuje radniku poruku : „**Sistem** je uspešno zapamtio *člana*“. (IA)



Slika DS1 – Osnovni scenario

**Alternativni scenario**:

* 1. Ukoliko **sistem** ne može da zapamti novog *člana*, prikazuje se poruka: „**Sistem** ne može da zapamti *člana*“.(IA)



Slika DS1 – Alternativni scenario

Sa navedenih sekvencnih dijagrama uočavaju se sistemske operacije koje treba projektovati:

1. *Signal* **ZapamtiNovogClana** (*Clan)*

**DSK 2: Dijagram sekvenci slučajeva korišćenja – Ubacivanje novog kursa**

**Osnovni scenario SK:**

1. **Forma** poziva **sistem** da učita sve *prostore*. (APSO)
2. **Sistem** učitava sve *prostore* na **formi**. (IA)
3. **Forma** poziva sistem da prikaže *tipove kursa*. (APSO)
4. **Sistem** učitava sve *tipove* na **formi**. (IA)
5. **Radnik** poziva **sistem** da zapamti novi *kurs*. (APSO)
6. **Sistem** prikazuje **radniku** poruku: „*Kurs* je uspešno zapamćen “. (IA)



**Alternativni scenario:**

* 1. Ukoliko **sistem** ne može da zapamti novi *kurs*, prikazuje se poruka: „**Sistem** ne može da zapamti *kurs*“.(IA)



Slika DS2 – alternativni scenario

Sa navedenih sekvencnih dijagrama uočavaju se sistemske operacije koje treba projektovati:

1. *Signal* **PrikaziProstore**(*List<Prostor>*)
2. *Signal* **PrikaziTipove**(*List<TipKursa>*)
3. *Signal* **ZapamtiKurs**(*Kurs*)
4. *Signal* **VratiSveProstore**(*List*<*Prostor*>)

**DS3: Dijagram sekvenici slučaja korišćenja – Izmena kursa**

**Osnovni scenario SK:**

1. **Forma** poziva **sistem** da pronadje *tipove kursa*. (APSO)
2. **Sistem** prikazuje *tipove kursa.* (IA)
3. **Radnik** poziva **sistem** da pronadje *kurs* za izabrani tip kursa. (APSO)
4. **Sistem** prikazuje željeni *kurs*. (IA)
5. **Radnik** poziva sistem da ažurira *kurs*. (APSO)
6. **Sistem** prikazuje poruku da je *kurs* ažuriran. (IA)



Slika DS3 – osnovni scenario

**Alternativni scenario:**

* 1. Ukoliko **sistem** ne može da pronađe *kurs*, prikazuje se poruka: „**Sistem** ne može da pronađe *kurs*“. Prekida se izvršenje scenarija.(IA)



Slika DS3 – alternativni scenario

* 1. Ukoliko **sistem** ne može da ažurira *kurs*, prikazuje se poruka: „**Sistem** ne može da ažurira *kurs*“.(IA)



Slika DS3 – alternativni scenario 2

Sa navedenih sekvencnih dijagrama uočavaju se sistemske operacije koje treba projektovati:

1. *Signal* **PrikaziTipove**(*List<TipKursa>*)
2. *Signal* **PronadjiKurs** (*Kurs)*
3. *Signal* **AzurirajKurs**(*Kurs*)

**DS 4: Dijagram sekvenci slučaja korišćenja – Pregled kurseva**

**Osnovni scenario SK:**

1. **Forma** poziva **sistem** da pronadje *tipove kursa*. (APSO)
2. **Sistem** prikazuje *tipove kursa.* (IA)
3. **Radnik** poziva **sistem** da pronadje *kurseve* za određeni uslov. (APSO)
4. **Sistem** prikazuje listu *kurseva* i poruku: „**Sistem** je uspešno pronašao *kurseve*“. (IA)



Slika DS4 – osnovni scenario

**Alternativni scenario:**

* 1. Ukoliko **sistem** ne može da pronađe *kurseve* za određeni uslov, prikazuje se poruka: „**Sistem** ne može da pronađe *kurseve* za određeni uslov“. (IA)



Slika DS4 – alternativni scenario

Sa navedenih sekvencnih dijagrama uočavaju se sistemske operacije koje treba projektovati:

1. *Signal* **PrikaziTipove**(*List<TipKursa>*)
2. *Signal* **PronadjiKurs** (*Kurs, List<Kurs>)*

**DS5: Dijagram sekvenci slučaja korišćenja – Brisanje kursa**

**Osnovni scenario SK:**

1. **Forma** poziva **sistem** da pronadje *tipove kursa*. (APSO)
2. **Sistem** prikazuje *tipove kursa.* (IA)
3. **Radnik** poziva **sistem** da pronadje kurseve za izabrani tip *kursa*. (APSO)
4. **Sistem** prikazuje listu *kurseva*. (IA)
5. **Radnik** poziva sistem da izbriše izabrani *kurs*. (APSO)
6. **Sistem** prikazuje poruku : „**Sistem** je uspešno obrisao *kurs*“. (IA)



Slika DS5 – osnovni scenario

**Alternativni scenario:**

4.1 Ukoliko **sistem** ne može da pronađe *kurseve*, prikazuje se poruka: „**Sistem** ne može da pronađe *kurseve*“. Prekida se izvršenje scenarija. (IA)



Slika DS5 – alternativni scenario

* 1. Ukoliko **sistem** ne može da izbriše *kurs*, prikazuje se poruka : „**Sistem** ne može da izbriše izabrani *kurs*“. (IA)



Slika DS5 – alternativni scenario 2

Sa navedenih sekvencnih dijagrama uočavaju se sistemske operacije koje treba projektovati:

1. *Signal* **PrikaziTipove**(*List<TipKursa>*)
2. *Signal* **PronadjiKurs** (*Kurs,List<Kurs>)*
3. *Signal* **ObrisiKurs**(*Kurs*)

**DS 6: Dijagram sekvenci slučaja korišćenja – Unos posete**

**Osnovni scenario SK:**

1. **Forma** poziva **sistem** da učita sve *članove*. (APSO)
2. **Sistem** učitava sve *članove* na **formi**. (IA)
3. **Forma** poziva **sistem** da učita sve *tipove kurseva* na **formi**. (APSO)
4. **Sistem** učitava sve *tipove kurseva* na **formi**. (IA)
5. **Forma** poziva **sistem** da učita sve *kurseve* na **formi**. (APSO)
6. **Sistem** učitava sve *kurseve* na **formi**. (IA)
7. **Radnik** poziva **sistem** da zapamti nova *prisustva*. (APSO)
8. **Sistem** prikazuje radniku poruku: „**Sistem** je uspešno zapamtio *prisustva* “. (IA)



Slika DS6 – osnovni scenario

**Alternativni scenario:**

* 1. Ukoliko **sistem** ne može da zapamti novu *posetu*, prikazuje se poruka: „**Sistem** ne može da zapamti novo *prisustvo*“. (IA)



Slika DS6- alternativni scenario

Sa navedenih sekvencnih dijagrama uočavaju se sistemske operacije koje treba projektovati:

1. *Signal* **PrikaziClanove**(*List<Clan>*)
2. *Signal* **PrikaziTipove**(*List<TipKursa>*)
3. *Signal* **PronadjiKurs** (*Kurs,List<Kurs>)*
4. *Signal* **ZapamtiPrisustva**(*List*<*Prisustvo>)*

### 3.2.2 Ponasanje softerskog sistema – Definisanje ugovora (Contracts) o sistemskim operacijama

**UGOVOR UG1: UnosClana**

Operacija: ZapamtiNovogClana(Clan): signal

Veza sa SK: SK1

Preduslovi: Vrednosna i strukturna ograničenja nad objektom član su ispunjena.

Postuslovi: Snimljen je novi član.

**UGOVOR UG2: PrikaziClanove**

Operacija: PrikaziClanove(List<Clan>):ListaClanova

Veza sa SK: SK6,SK7,SK8,SK9

Preduslovi:

Postuslovi:

**UGOVOR UG3: ZapamtiPrisustvo**

OperacijaZapamtiPrisustvo(Prisustvo): signal

Veza sa SK: SK6

Preduslovi: Vrednosna i strukturna ograničenja nad objektom prisustvo su ispunjena.

Postuslovi: Sacuvane su posete

**UGOVOR UG4: IzmeniPrisustvo**

Operacija: IzmeniPrisustvo (Prisustvo): signal

Veza sa SK: SK8

Preduslovi: Vrednosna i strukturna ograničenja nad objektom prisustvo su ispunjena.

Postuslovi:Izmenjena su prisustva.

**UGOVOR UG5: ObrisiPrisustvo**

Operacija: ObrisiPosetu(Prisustvo): signal

Veza sa SK: SK9

Preduslovi: Strukturna ograničenja nad objektom prisustvo su ispunjena.

Postuslovi:Obrisana su prisustva.

**UGOVOR UG6: VratiPrisustva**

Operacija: VratiPrisustva (Poseta,List<Prisustvo>): signal

Veza sa SK: SK7,SK8,SK9

Preduslovi:

Postuslovi:

**UGOVOR UG7: AzurirajKurs**

Operacija: AzurirajKurs (Kurs): signal

Veza sa SK: SK3

Preduslovi: Vrednosna i strukturna ograničenja nad objektom kurs su ispunjena.

Postuslovi: Kurs je izmenjen.

**UGOVOR UG8: ObrisiKurs**

Operacija: ObrisiKurs (Kurs): signal

Veza sa SK: SK5

Preduslovi: Strukturna ograničenja nad objektom kurs su ispunjena.

Postuslovi: Obrisani su podaci o kursu.

**UGOVOR UG9: ZapamtiKurs**

Operacija: ZapamtiKurs (Kurs) : signal

Veza sa SK: SK2

Preduslovi: Vrednosna i strukturna ograničenja nad objektom kurs su ispunjena.

Postuslovi: Sacuvan je kurs

### 3.2.3 Struktura softverskog sistema

U ovom poglavlju je detaljno prikazana struktura sistema.

#### 3.2.3.1 Struktura softverskog sistema – konceptualni (domenski) model

Sledeća slika prikazuje konceptualni model sistema:



Slika Konceptualni (Domenski) model



Slika klasa IOOpstiDomenskiObjekat 1

#### 3.2.3.2 Struktura softverskog sistema – Relacioni model

Na osnovu konceptualnog modela koji je predstavljen dijagramom klasa dobijen je

sledeći relacioni model:

Clan(sifraClana ,ime, prezime, email, mobilni)

Kurs(idKursa, naziv, trajanje, cena, *idTipa, sifraProstora*)

Ucionica(sifraProstora,naziv)

TipKursa(idTipa, nazivTipa)

Prisustvo(*sifraClana*,*idKursa*,brojPosete, datum)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Tabela Clan** | | **Prosto vrednosno ograničenje** | | **Složeno vrednosno ograničenje** | | **Strukturno ograničenje** |
| **Atributi** | **Ime** | **Tip atributa** | **Vrednost atributa** | **Međuzav. atributa jedne tabele** | **Međuzav. atributa više tabela** | INSERT/  UPDATE  CASCADE  Prisustvo  DELETE  RESTRICTED Prosustvo |
| sifraClana | Integer | Not null  and >0 |  |  |
| email | String | Not null |  |  |

Tabela Strukturna i vrednosna ograničenja za tabelu Clan

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Tabela Kurs** | | **Prosto vrednosno ograničenje** | | **Složeno vrednosno ograničenje** | | **Strukturno ograničenje** |
| **Atributi** | **Ime** | **Tip atributa** | **Vrednost atributa** | **Međuzav. atributa jedne tabele** | **Međuzav. atributa više tabela** | INSERT  RESTRICTED  TipKursa  UPDATE RESTRICTED  TipKursa  UPDATE CASCADES Kurs  DELETE  RESTRICTED  Prisustvo |
| idKursa | Integer | Not null  and>0 |  |  |
| Cena | Double | Not null |  |  |
| Naziv | String | Not null |  |  |
| Trajanje | Integer | Not null |  |  |
| idTipa | Integer | Not null  and >0 |  |  |

Tabela Strukturna i vrednosna ograničenja za tabelu kurs

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Tabela TipKursa** | | **Prosto vrednosno ograničenje** | | **Složeno vrednosno ograničenje** | | **Strukturno ograničenje** |
| **Atributi** | **Ime** | **Tip atributa** | **Vrednost atributa** | **Međuzav. atributa jedne tabele** | **Međuzav. atributa više tabela** | INSERT/  UPDATE  CASCADE  Kurs  DELETE  RESTRICTED Kurs |
| idTipa | Integer | Not null  and >0 |  |  |
| NazivTipa | String | Not null |  |  |

Tabela Strukturna i vrednosna ograničenja za tabelu TipKursa

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Tabela Prisustvo** | | **Prosto vrednosno ograničenje** | | **Složeno vrednosno ograničenje** | | **Strukturno ograničenje** |
| **Atributi** | **Ime** | **Tip atributa** | **Vrednost atributa** | **Međuzav. atributa jedne tabele** | **Međuzav. atributa više tabela** | INSERT  RESTRICTED Clan, Kurs  UPDATE RESTRICTED Clan, Kurs  DELETE/ |
| idKursa | Integer | Not null  and >0 |  |  |
| sifraClana | Integer | Not nul  and >0 |  |  |
| datum | String | Not null |  |  |
| brojPosete | Integer | Not null |  |  |

Tabela Strukturna i vrednosna ograničenja za tabelu Prisustvo

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Tabela Ucionica** | | **Prosto vrednosno ograničenje** | | **Složeno vrednosno ograničenje** | | **Strukturno ograničenje** |
| **Atributi** | **Ime** | **Tip atributa** | **Vrednost atributa** | **Međuzav. atributa jedne tabele** | **Međuzav. atributa više tabela** | INSERT/  UPDATE  CASCADE  Kurs  DELETE  RESTRICTED Kurs |
| SifraProstora | Integer | Not null |  |  |
| Naziv | String |  |  |  |

Tabela Strukturna i vrednosna ograničenja za tabelu Ucionica

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Tabela Korisnik** | | **Prosto vrednosno ograničenje** | | **Složeno vrednosno ograničenje** | | **Strukturno ograničenje** |
| **Atributi** | **Ime** | **Tip atributa** | **Vrednost atributa** | **Međuzav. atributa jedne tabele** | **Međuzav. atributa više tabela** | INSERT/  UPDATE  CASCADE  Poseta  DELETE  RESTRICTED Poseta |
| sifra | Integer | Not null  and >0 |  |  |
| Username | String | Not null |  |  |
|  | Password | String | Not null |  |  |  |

Tabela Strukturna i vrednosna ograničenja za tabelu Korisnik

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Tabela Osoba** | | **Prosto vrednosno ograničenje** | | **Složeno vrednosno ograničenje** | | **Strukturno ograničenje** |
| **Atributi** | **Ime** | **Tip atributa** | **Vrednost atributa** | **Međuzav. atributa jedne tabele** | **Međuzav. atributa više tabela** | INSERT  /  CASCADES  Korisnik, Clan  RESTRICTED  CLAN |
| Sifra | Integer | Not null  and>0 |  |  |
| Ime | String | Not null |  |  |
| Prezime | String | Not null |  |  |
| Mobilni | String | Not null  and >0 |  |  |

Tabela Strukturna i vrednosna ograničen ja za tabelu osoba

Kao rezultat analize scenarija pravljenja konceptualnog modela dobija se logička

struktura i ponašanje sistema. Struktura i ponašanje sistema je prikazano na slici



Slika Dijagram klasa I domenski objekat

## Projektovanje

Faza projektovanja opisuje fizičku strukturu i ponašanje softverskog sistema

(arhitekturu softverskog sistema).

Ona obuhvata projektovanje:

* Aplikacione logike
* Kontrolera
* Poslovne logike
* Logičke strukture
* Ponašanja softverskog sistema
* Brokera baze podataka
* Skladišta podataka

### 3.3.1 Arhitektura softverskog sistema

Na primeru ovog rada se koristi troslojna arhitektura, i obuhvata pomenute nivoe:

1) Korisnički interfejs koji predstavlja ulazno-izlaznu reprezentaciju softverskog

sistema

2) Aplikaciona logika koja opisuje strukturu i ponašanje softverskog sistema

3) Skladište podataka koje čuva stanje atributa softverskog sistema



Slika Troslojna arhitektura – detaljan prikaz

### 3.3.2 Projektovanje korisničkog interfejsa

Korisnički interfejs predstavlja realizaciju ulaza i/ili izlaza softverskog sistema. Korisnički interfejs sastoji se od ekranske forme i kontrolera korisničkog interfejsa. Ekranska forma prihvata podatke i događaje, poziva kontrolera i prikazuje rezultate, a Kontroler korisničkog interfejsa prihvata podatke, konvertuje ih, proverava da li su u zadovoljavajućoj formi, šalje zahtev za izvršenje SO do softverskog sistema, prihvata izlaz i konvertuje objekat u podatke grafičkih elemenata.

### 3.3.2.1 Projektovanje kontrolera korisničkog interfejsa

Kontroler korisničkog interfejsa ima ulogu da prosledjuje zahteve ka serverskom

programu. Na ovom primeru je implementiran kao singlton patern i može se pozvati

jedna instanca sa svih formi sistema.



Slika Prikaz veze korisnika sa sistemom

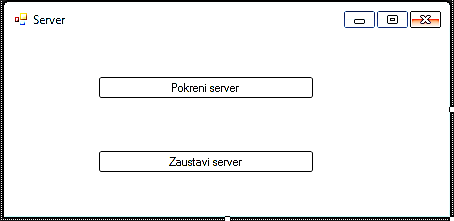
### 3.3.3 Projektovanje ekranskih formi

Korisnički interfejs definisan je preko skupa ekranskih formi. Scenarija slučajeva korisćenja ekranskih formi je direktno povezan sa scenarijima slučajeva korišćenja. Postoje dva aspekta projektovanja ekranske forme:

1. Projektovanje scenarija slučajeva korišćenja koji se izvode preko ekranske forme

2. Projektovanje metoda ekranske forme

Glavna ekranska forma serverskog dela aplikacije izgleda ovako:



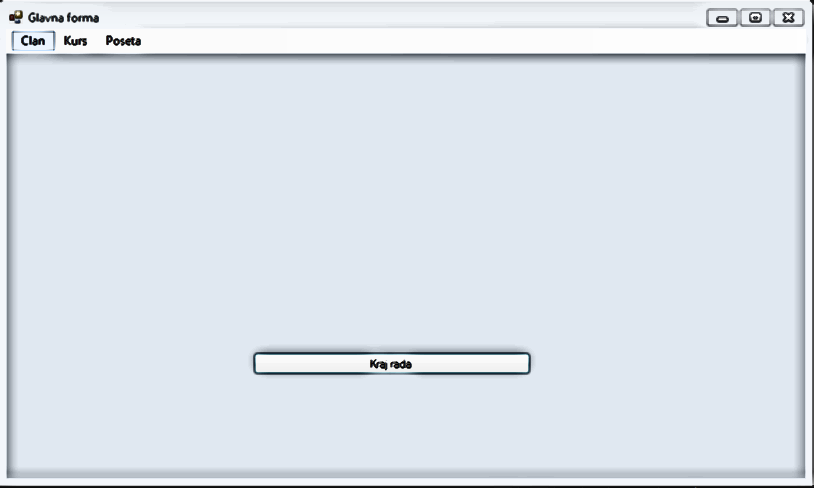
Slika Pokretanje servera

Glavna ekranska forma klijentskog dela aplikacije izgleda ovako:



Slika Login korisnika

Glavna ekranska forma klijentskog dela aplikacije kada se administrator uloguje izgleda ovako:



Slika Glavna forma

**SK1: Slučaj korišćenja – Ubacivanje novog člana**

**Naziv SK**:

Ubacivanje novog *člana*

**Akteri SK:**

**Radnik**

**Učesnici SK:**

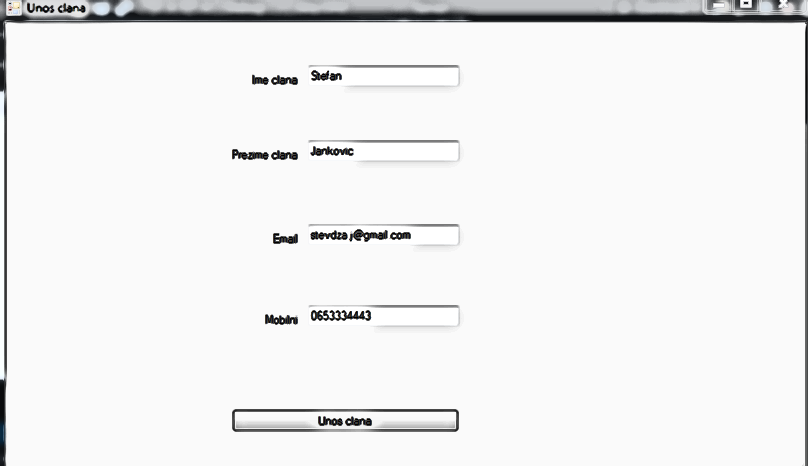
**Radnik** i **program**(**sistem**)

Preduslov: **Sistem** je uključen. Otvorena je forma za unos *članova*.

**Osnovni scenario SK:**

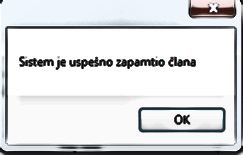
1. **Radnik** unosi podatke o novom *članu*. (APUSO)
2. **Radnik** poziva sistem da zapamti novog *člana* (APSO)

Opis akcije: Administrator klikom na dugme Unos clana poziva sistemsku operaciju UnesiClana(Clan) koja pamti novog clana.



Slika Unos clana

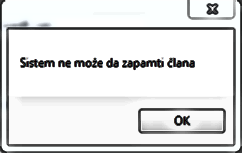
1. **Sistem** pamti novog *člana*. (SO)
2. **Sistem** pokazuje radniku poruku: „**Sistem** je uspešno zapamtio *člana* “. (IA)



Slika Obavestenje o uspešnosti operacije

**Alternativni scenario**:

* 1. Ukoliko **sistem** ne može da zapamti novog *člana*, prikazuje se poruka: „**Sistem** ne može da zapamti *člana* “. (IA)



Slika Obaveštenje o neuspešnosti operacije

**SK 2: Slučaj korišćenja – Ubacivanje novog kursa**

**Naziv SK:**

Ubacivanje novog *kursa*

**Akteri SK:**

**Radnik**

**Učesnici SK:**

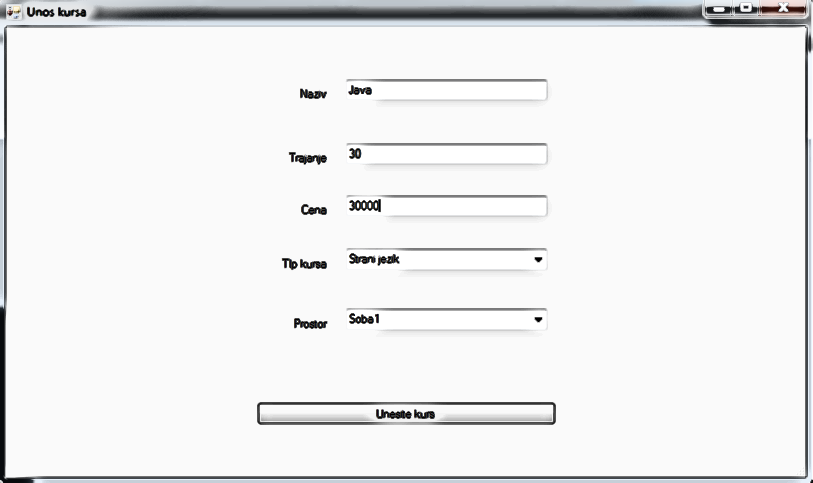
**Radnik** i **sistem**

**Preduslov**: **Sistem** je uključen. Otvorena je forma za unos *kursa*. Učitani su svi tipovi *kurseva*.

**Osnovni scenario SK:**

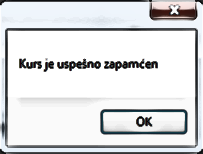
1. **Radnik** unosi podatke o novom *kursu*. (APUSO)
2. **Radnik** poziva **sistem** da zapamti novi *kurs*. (APSO)

Opis akcije: Randik pritiskom na dugme “Unesite kurs” poziva operaciju ZapamtiKurs(kurs) koja pamti novi kurs.



Slika Forma za unos kursa

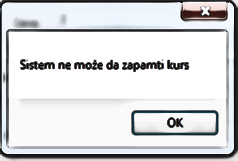
1. **Sistem** pamti novi *kurs*. (SO)
2. **Sistem** prikazuje **radniku** poruku: „*Kurs* je uspešno zapamćen“. (IA)



Slika Obaveštenje o uspešnosti operacije

**Alternativni scenario:**

* 1. Ukoliko **sistem** ne može da zapamti novi *kurs*, prikazuje se poruka: „**Sistem** ne može da zapamti *kurs* “. (IA)



Slika Obaveštenje o neuspešnosti operacije

**SK 3: Slučaj korišćenja – Izmena kursa**

**Naziv SK:**

Izmena postojećeg *kursa*

**Akteri SK:**

**Radnik**

**Učesnici SK:**

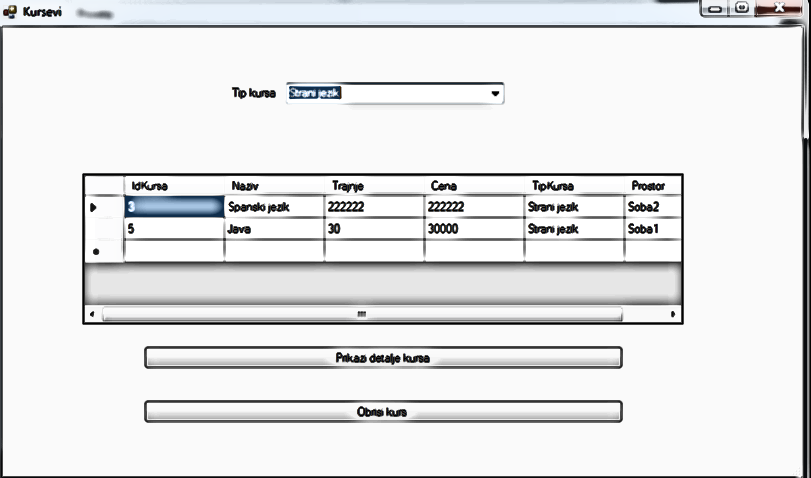
**Radnik** i **sistem**

**Preduslov**: **Sistem** je uključen i **radnik** je ulogovan pod svojom šifrom. Otvorena je forma za izmenu *kursa*. Učitani su svi tipovi *kurseva*.

**Osnovni scenario SK:**

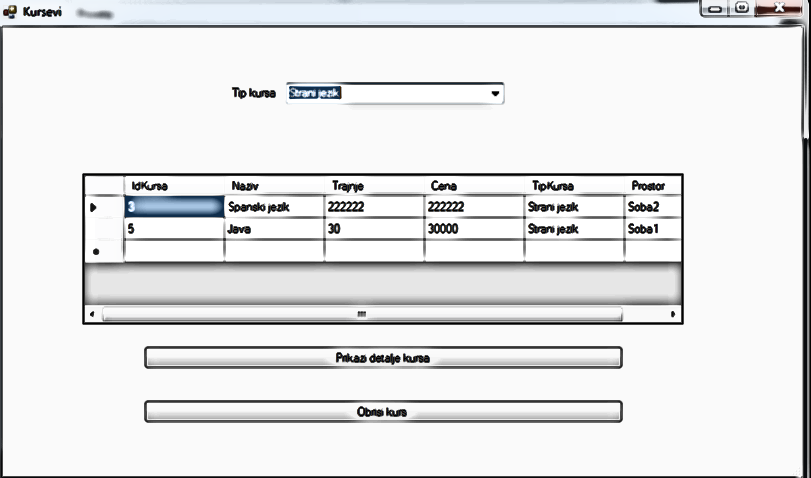
1. **Radnik** bira tip *kursa* iz spiska *kurseva*. (APUSO)
2. **Radnik** poziva **sistem** da pronadje kurseve za izabrani tip *kursa*. (APSO)

Opis akcije: Administrator unosom kriterijuma pretrage automatski poziva sistemsku operaciju PronađiKurs(kriterijumPretrage, List<Kurs>) koja pronalazi kurseve



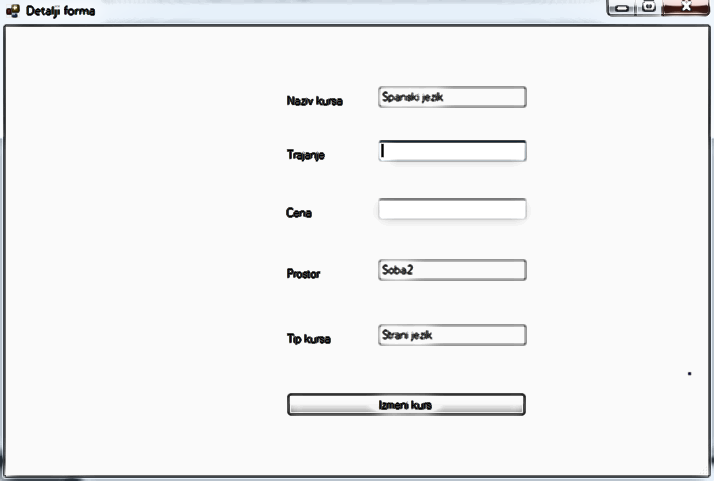
Slika Forma za rad sa kursevima

1. **Sistem** pronalazi *kurseve*. (SO)
2. **Sistem** prikazuje listu *kurseva*. (IA)
3. **Radnik** bira *kurs* iz spiska *kurseva*. (APUSO)



Slika Forma za rad sa kursevima 2

1. **Sistem** prikazuje podatke o izabranom *kursu*. (IA)
2. **Radnik** menja podatke *kursa*. (APUSO)

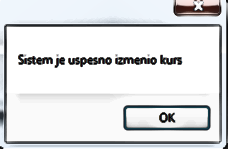


Slika Detalji kursa forma

1. **Radnik** poziva sistem da ažurira *kurs*. (APSO)

Opis akcije: Radnik klikom na dugme Izmeni kurs poziva sistemsku operaciju IzmeniKurs(Kurs) koja azurira informacije o kursu.

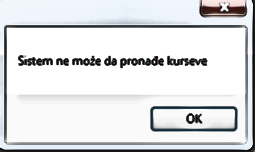
1. **Sistem** ažurira *kurs*. (SO)
2. **Sistem** prikazuje poruku da je *kurs* ažuriran. (IA)



Slika Obaveštenje o uspešnosti operacije

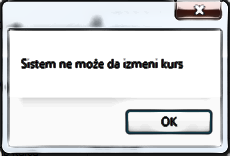
**Alternativni scenario:**

2.1 Ukoliko **sistem** ne može da pronađe *kurseve*, prikazuje se poruka: „**Sistem** ne može da pronađe *kurseve* “. Prekida se izvršenje scenarija. (IA)



Slika Obaveštenje o neuspešnosti operacije 1

8.1 Ukoliko **sistem** ne može da izmeni *kurs*, prikazuje se poruka: „**Sistem** ne može da izmeni *kurs*“. (IA)



Slika Obaveštenje o neuspešnosti operacije 2

**SK 4: Slučaj korišćenja – Pregled kurseva**

**Naziv SK:**

Pregled *kurseva*

**Akteri SK:**

**Radnik**

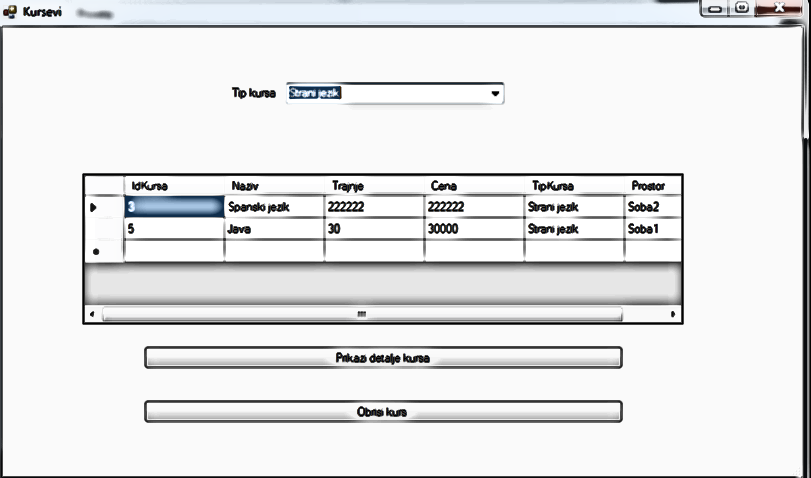
**Učesnici SK:**

**Radnik** i **sistem**

**Preduslov**: **Sistem** je uključen. Otvorena je forma za pregled *kurseva*. Učitani su svi tipovi *kurseva*.

**Osnovni scenario SK:**

1. **Radnik** bira tip *kursa* iz spiska svih tipova *kurseva*.(APUSO)

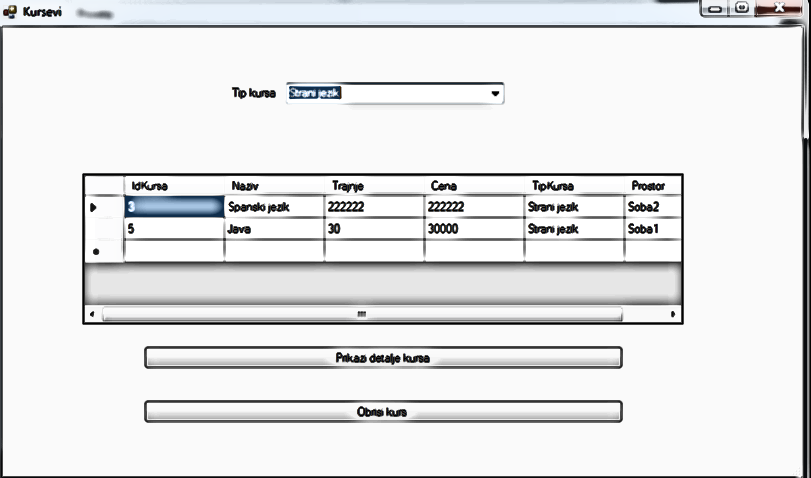


Slika Forma za rad sa kursevima 3

1. **Radnik** poziva **sistem** da pronadje *kurseve* za izabrani tip *kursa*. (APSO)

Opis akcije: Radnik unosom kriterijuma pretrage automatski poziva sistemsku operaciju PronadjiKurs(kriterijumPretrage, List<Kurs>) koja pronalazi kurseve.

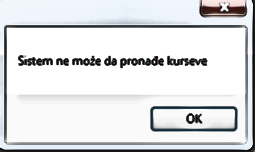
1. **Sistem** pronalazi *kurseve*. (SO)
2. **Sistem** prikazuje listu *kurseva* i poruku: „**Sistem** je uspešno pronašao *kurseve* “. (IA)



Slika Forma za rad sa kursevima 4

**Alternativni scenario:**

* 1. Ukoliko **sistem** ne može da pronađe *kurseve* za određeni tip, prikazuje se poruka: „**Sistem** ne može da pronađe *kurseve* “. (IA)



Slika Obaveštenje o neuspešnosti operacije

**SK 6: Slučaj korišćenja – Unos prisustva**

**Naziv SK:**

Unos novog *prisustva*

**Akteri SK:**

**Radnik**

**Učesnici SK:**

**Radnik** i **sistem**

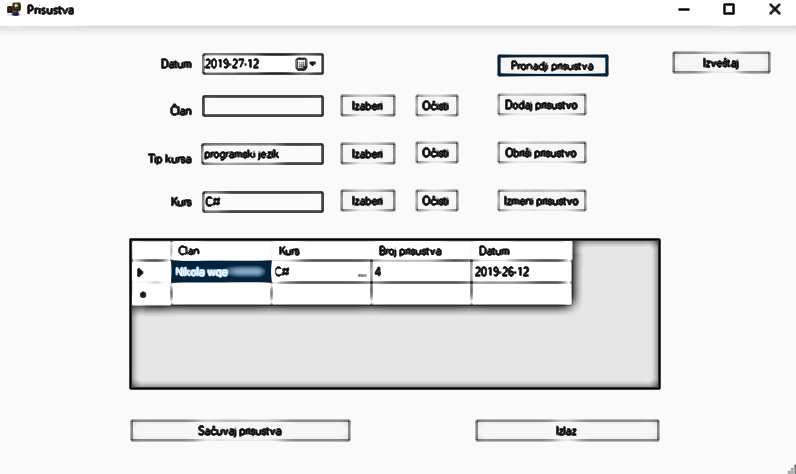
**Preduslov**: **Sistem** je uključen. Otvorena je forma za ubacivanje novog *prisustva*. Učitani su svi članovi, tipovi kursa, kursevi i njihova *prisustva*.

.

**Osnovni scenario SK:**

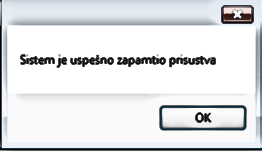
1. **Radnik** unosi podatke o novom prisustvu člana. (APUSO)
2. **Radnik** poziva **sistem** da zapamti novo *prisustvo*. (APSO)

Opis akcije: Administrator klikom na dugme Sacuvaj posete poziva sistemsku operaciju ZapamtiPrisustvo(Prisustvo) koja pamti nova prisustva.



Slika Forma za rad sa prisustvima SK 6

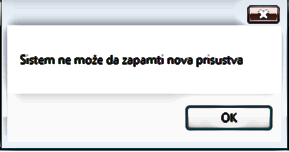
1. **Sistem** pamti nova *prisustva*. (SO)
2. **Sistem** prikazuje radniku poruku: „**Sistem** je uspešno zapamtio *prisustva*“. (IA)



Slika Obaveštenje o uspešnosti operacije SK6

**Alternativni scenario:**

* 1. Ukoliko **sistem** ne može da zapamti nove *posete*, prikazuje se poruka: „**Sistem** ne može da zapamti nova *prisustva* “. (IA)

****

Slika Obaveštenje o neuspešnosti operacije SK6

**SK 14: Slučaj korišćenja – Prikaz izveštaja**

**Naziv SK:**

Prikaz izveštaja

**Akteri SK:**

**Radnik**

**Učesnici SK:**

**Radnik** i **sistem**

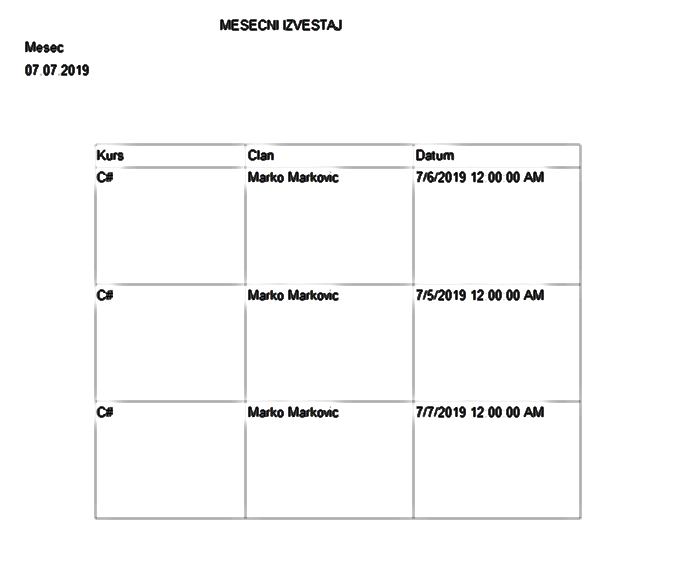
**Preduslov**: **Sistem** je uključen i **radnik** je ulogovan pod svojom šifrom. Otvorena je forma za rad sa *posetama*. Učitani su svi članovi i kursevi.

**Osnovni scenario SK:**

1. **Radnik** poziva sistem da izvrši prikaz izveštaja o kursevima za dati mesec. (APSO)

Opis akcije: Radnik unosom kriterijuma pretrage automatski poziva sistemsku operaciju PrikaziIzvestaj(kriterijumPretrage, List<Kurs>) koja pronalazi kurseve.

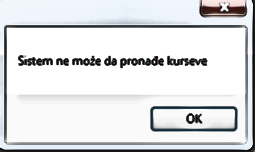
1. **Sistem** otvara prozor za izvetaj. (SO)
2. **Sistem** prikazuje listu *kurseva*. (IA)

****

Slika Prikaz izveštaja

**Alternativni scenario:**

* 1. Ukoliko **sistem** ne može da pronađe *listu*, prikazuje se poruka: „**Sistem** ne može da pronađe *listu kurseva* “. Prekida se izvršenje scenarija. (IA)



Slika Obaveštenje o neuspešnosti operacije SK 14

### 3.3.3 Projektovanje aplikacione logike – Kontroler

Aplikacioni serveri su odgovorni da obezbede servise koji će da omoguće realizaciju aplikacione logike softverskog sistema. Projektovani aplikacioni server sadrži:

* deo za komunikaciju sa klijentima,
* kontroler aplikacione logike,
* deo za komunikaciju sa skladištem podataka (Broker baze podataka),
* deo koji sadrži poslovnu logiku.

Kontroler aplikacione logike prihvata zahtev za izvršenje sistemske operacije od niti klijenta i dalje ga preusmerava do klasa koje su odgovorne za izvršenje sistemskih operacija. Nakon izvršenja sistemske operacije kontroler aplikacione logike prihvata rezultat i prosleđuje ga pozivaocu (niti klijenta). Kako je u fazama prikupljanja zahteva i analize data specifikacija strukture i ponašanja softverskog sistema, odnosno specifikacija poslovne logike softverskog sistema, sledeća slika daje opis sistema nakon faze projektovanja komunikacije sa klijentima i kontrolera aplikacione logike.



Slika Sistemske Operacije

#### 3.3.3.1 Poslovna logika – Struktura softverskog sistema

Domenske klase prave se na osnovu konceptualnog modela. Struktura modela je identicna strukturi klasa u samoj aplikaciji

#### 

Slika Dijagram klasa

#### 3.3.3.2 Poslovna logika – Ponašanje softverskog sistema

Za svaku sistemsku operaciju pravimo konceptualna rešenja koja su povezana sa

problemom. Za svaki od ugovora projektuje se jedno ovakvo konceptualno rešenje.

**UGOVOR UG1: UnosClana**

Operacija: UnosClana(clan): signal

Veza sa SK: SK1

Preduslovi:

Postuslovi: Snimljen je novi član.



Slika Dijagram sekvenci - UnosClana

**UGOVOR UG2: PrikaziClanove**

Operacija: PrikaziClanove(List<Clan>):ListaClanova

Veza sa SK: SK6,SK7,SK8,SK9

Preduslovi:

Postuslovi:



Slika Dijagram sekvenci - PrikaziClanove

**UGOVOR UG3: ZapamtiPosete**

OperacijaZapamtiPosetu (Poseta): signal

Veza sa SK: SK6

Preduslovi: Vrednosna i strukturna ograničenja nad objektom poseta su ispunjena.

Postuslovi: Sacuvane su posete



Slika Dijagram sekvenci - ZapamtiPosete

**UGOVOR UG4: IzmeniPosetu**

Operacija: IzmeniPosetu (Poseta): signal

Veza sa SK: SK8

Preduslovi: Vrednosna i strukturna ograničenja nad objektom član su ispunjena.

Postuslovi:Izmenjene su posete.



Slika Dijagram sekvenci - IzmeniPosetu

**UGOVOR UG5: ObrisiPosetu**

Operacija: ObrisiPosetu(Poseta): signal

Veza sa SK: SK9

Preduslovi: Strukturna ograničenja nad objektom član su ispunjena.

Postuslovi:Obrisane su posete.



Slika Dijagram sekvenci - ObrisPosetu

**UGOVOR UG6: VratiPosete**

Operacija: VratiPosete (Poseta,List<Poseta>): signal

Veza sa SK: SK7,SK8,SK9

Preduslovi:

Postuslovi:



Slika Dijagram sekvenci - VratiPosete



Slika Konacna arhitektura sistema

## Implementacija

Ovaj sistem je razvijen za potrebe edukativnog centra koji će biti instaliran na jednom računaru. Definisano je da će dva obučena korisnika koristiti aplikaciju, pri tom što postoji mogućnost povećanja tog broja. Njegova uloga je da beleži korisnike I njihove posete raznih kurseva koje centar ima na raspolaganju. Radi se o windows aplikaciji.

Izrada aplikacije je počela u maju, završetak implementacije je planiran do kraja septembra. Iznos budžeta nije definisan, tako da se novac izdvaja po potrebi. Softver se nabavljao sa raznih oficijalnih sajtova. Znanje se prikupljalo takođe sa raznih sajtova, knjiga. Na projektu je jedan programer, koji je ujedno i autor dokumentacije. Ukoliko je potrebno unapređivanje sofrvera biće potrebno da se ulože dodatna sredstva. Sam sistem nema ograničenje na broj korisnika koji može da pokriva mesečno, iako se to može promeniti po potrebi.

Trajanje same analize je bilo mesec dana. Tu su se definisale klase koje će se koristiti, njihova povezanost i relacije u samom modelu, slučajevi korišćenja, dijagram sekvenci koji predstavlja arhitekturu naše baze.

Što se hardvera na kom se sistem implementira tiče, nije potrebna preterano zahtevna konfiguracija. Bilo bi poželjno imati SSD zbog brzine izvršavanja zadataka. Tu se može uštedeti na grafičkoj kartici koja nije mnogo bitna za funkcionisanje našeg softvera.

Obuka zaposlenih je planirana nakon završetka izrade aplikacije. Njeno trajanje bi bila jedna radna nedelja. Biće obučeni kako da upravljaju aplikacijom, neće učestvovati u izmenama I popravkama softvera. Programeri koji budu radili na održavanju će biti obučavani po potrebi. Obe strane će obučavati solution architect sa saradnicima ukoliko bude bilo potrebe za njima. Oni će imati odgovarajuće predznanje koje se tiče rad na odgovarajućim softverskim alatima, tak oda neće biti potrebe za velikom količinom resursa za njihovu obuku. Obučavaćemo ih tako što ćemo prolaziti kroz aplikaciju I detaljno objašnjavati svaku formu. Biće potrebni: Projektor, računar i prostorija sa određenim brojem stolica.

O sigurnosti nije bila obraćana velika pažnja. Postoje šifre za login korisnika koji pristupaju i rade na sistemu. Po potrebi se može proširiti.

Ideja samog sistema ja da olakša centru evidenciju klijenata. Zaposleni bi mnogo manje trošili vremena pri obavljanju operacija kao što su unos, izmena, brisanje, prikaz …

Što se samog testiranja tiče, postoji mogućnost instalacije alata koji bi to izvršavali automatski u određenom vremenskom period koje bi nadgledala osoba obučena za to.

Bazu podataka će održavati programeri na mesečnom nivou. Tu će se voditi računa o stvarima kao što su dupli I nepotrebni podaci. Na zahtev klijenta baza će biti proširivana ili menjana, za sad to nije planirano. Moguće je da se na osnovu ovog modela aplikacija prebaci na web. U tom slučaju je potrebno da se zakupi server na kome bi radila aplikacija. Za održavanje web aplikacije bi bio zadužen web programer za kog bi morale da se odvoje određena novčana sredstva.

Da bi za aplikaciju rekli da je uspešno implementirana, moraju da se ostvare određeni kriterijumi:

* Ne sme doći do prekida rada aplikacije pri njenom pokretanju
* Ne sme doći do prekida rada aplikacije pri izvršavanju funkcija te aplikacije
* Moraju biti zadovoljene sve validacije koje su definisane u sistemu

Što se samih validacija tiče, stvari koje moraju biti zadovoljene su:

* Datumi moraju biti u odgovarajućem formatu
* Cena mora biti realan broj
* Trajanja su izraženi u danima koji su predstavljeni celim brojevima
* Mejlovi moraju biti u odgovarajućem formatu
* Brojevi ne smeju sadržati slova

Softverski sistem, rezultat ovog rada razvijen je u programskom jeziku C#, a izvorni kod je priložen na pratećem CD-u. Sistem je projektovan kao klijent-server. Kao razvojno okruženje korišćen je Visual Studio 2017. Kao sistem za upravljanjem bazom podataka korišćen je MS SQL Server Managment Studio, a za izradu UML dijagrama Microsoft Visio 2007. Klijent je implementiran kao desktop aplikacije, a server kao konzolna aplikacija. Na osnovu arhitekture softverskog sistema dobijene su sledeće softverske klase:

Čitav sistem je realizovan u projektima:

* + Broker
  + Domen
  + Forme
  + KontrolerPL
  + Server

Broker:

BrokerBaze.cs

Domen:

Clan.cs

IOOpstiDomenskiObjekat.cs

Korisnik.cs

Kurs.cs

Osoba.cs

Prisustvo.cs

TipKursa.cs

Transfer.cs

Ucionica.cs

Forme:

DetaljiKursaForma.cs

Form1.cs

FormaReport.cs

FormaZaIzmenuPosete.cs

FormaZaRadSaKursevima.cs

IzmenaClana.cs

IzvestajForma.cs

Komunikacija.cs

KontrolerKI.cs

KurseviZaPoseteForma.cs

Login.cs

ManipulacijaSaClanovima.cs

PregledImpotovanihClanovaForma.cs

PrikazClanovaZaPosete.cs

Program.cs

ProstorForma.cs

RadSaPosetamaForma.cs

TipKursaZaPoseteForma.cs

TipoviKurseva.cs

TipoviKursevaZaRad.cs

UnosClanaForma.cs

UnosNovogKursa.cs

KontrolerPL:

Kontroler.cs

Repozitorijum.cs

Server:

ObradaKlijenata.cs

PokretanjeServer.cs

Program.cs

Server.cs

SistemskeOperacije:

AzurirajClana.cs

AzurirajKursa.cs

IzmeniPosetu.cs

LoginKorisnika.cs

ObrisiClana.cs

ObrisiKurs.cs

ObrisiPosetu.cs

OpstaSistemskaOperacija.cs

PrikaziClanove.cs

PrikaziKurseve.cs

PrikaziTipove.cs

UnosClana.cs

VratiPosete.cs

VratiPoseteZaMesec.cs

VratiSveProstore.cs

ZapamtiKurs.cs

ZapamtiPosete.cs

Softver se sastoji iz tri sloja:

* Sloj base podataka
* Srednji sloj
* Prezentacioni sloj

### Sloj baze podataka

U ovom poglavlju će biti prikazana struktura naše baze, to jest tabele, skripte za njihovo kreiranje, kao i njihove međusobne relacije.



Slika Scripta baze 1

Na osnovu softverskih klasa strukture projektovane su tabele tj. skladišta podataka.

Skladište podataka: Član

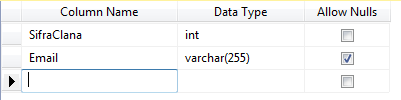
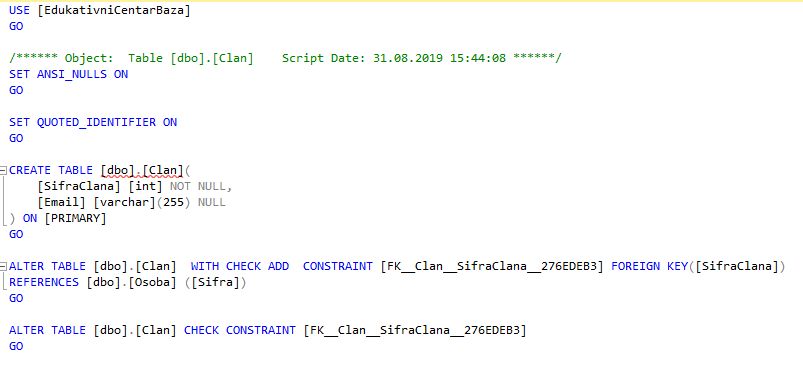


Tabela Clan



Slika Clan skripta

Skladište podataka: Kurs

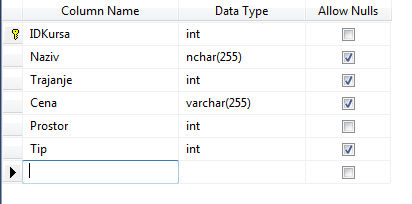
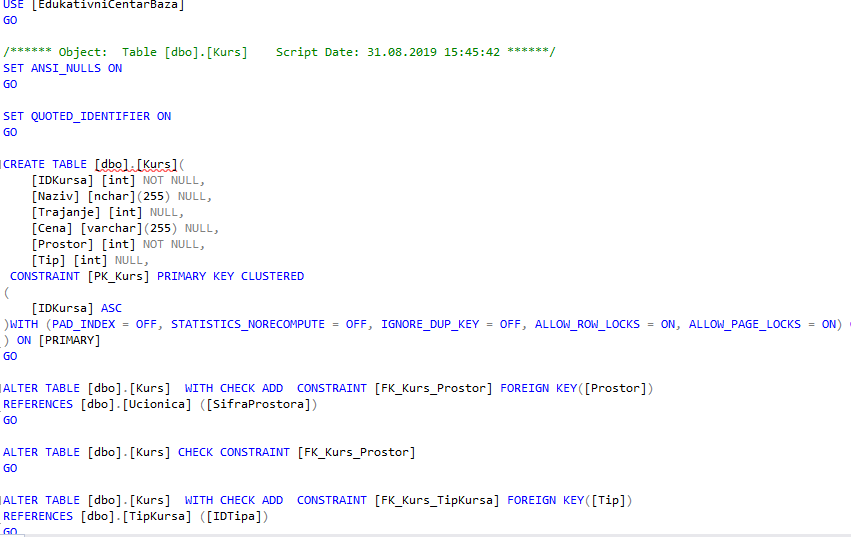


Tabela Kurs



Slika Kurs skripa

Skladište podataka: Prisustvo

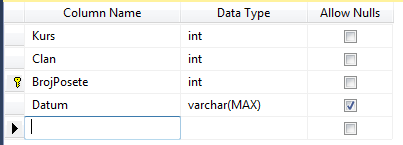
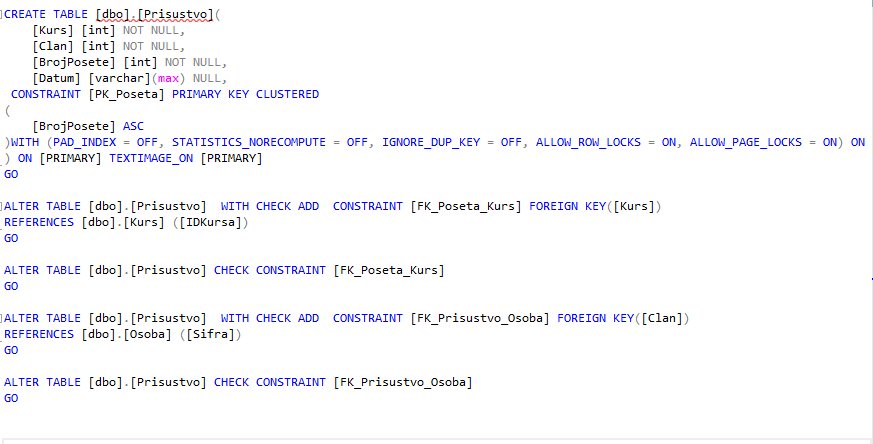


Tabela Prisustvo



Slika Prisustvo skripta

Skladište podataka: Ucionica

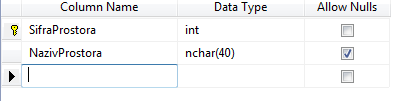
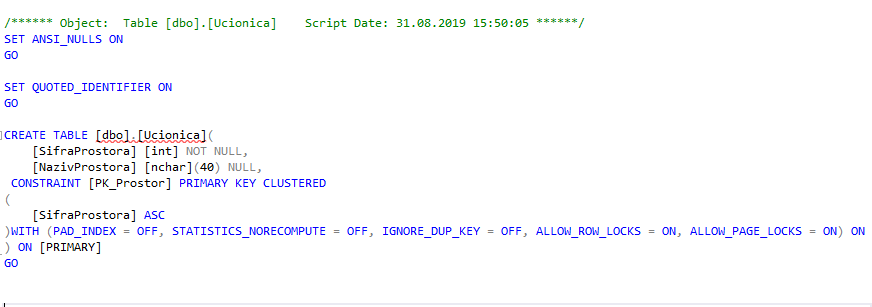


Tabela Ucionica



Slika Ucionica skripta

Skladište podataka: TipKursa

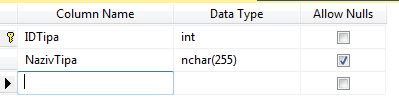
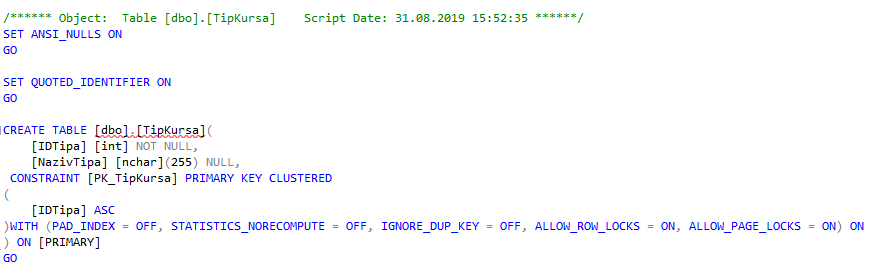


Tabela TipKursa



Tip kursa skripta

Skladište podataka: Osoba

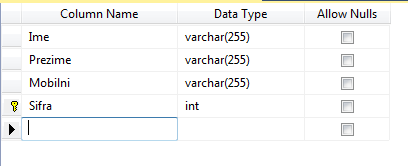
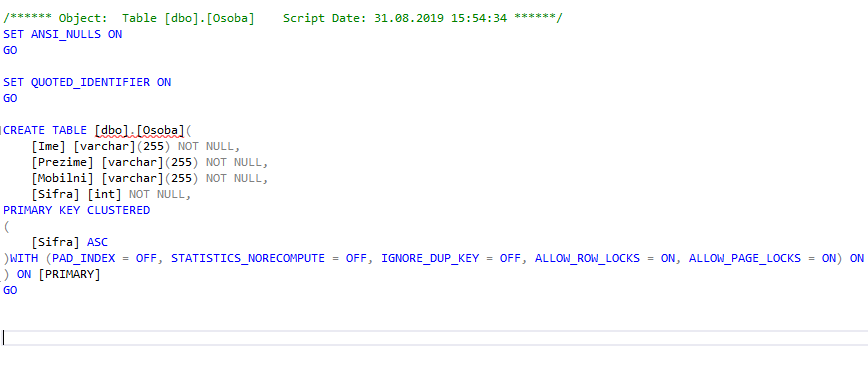
****

Tabela Osoba

****

Slika Skripta osoba

Skladište podataka: Korisnik

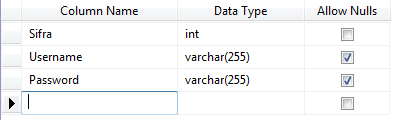
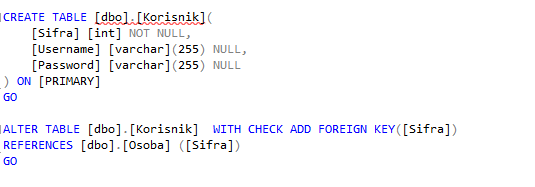
****

Tabela Korisnik

****

Korisnik skripta

### Srednji sloj

Klasa DBBroker predstavlja perzistentni okvir koji posreduje u svim operacijama nad bazom podataka i realizuje sledeće metode:

public int Ubaci(OpstiDomenskiObjekat objekat)

public bool Azuriraj(OpstiDomenskiObjekat objekat)

public bool Obrisi(OpstiDomenskiObjekat odo)

public List< IOOOpstiDomenskiObjekat > vratiSveZaUslovDva(OpstiDomenskiObjekat odo)

public List< IOOOpstiDomenskiObjekat > vratiSveZaUslov (OpstiDomenskiObjekat odo)

public List< IOOpstiDomenskiObjekat > vratiSveZaUslovJedan(OpstiDomenskiObjekat odo)

public int VratiSifru(OpstiDomenskiObjekat odo)

public void Rollback()

public void Commit()

public void OtvoriKonekciju()

public void ZatvoriKonekciju()

public void PokreniTransakciju()

public bool azuriraj2(OpstiDomenskiObjekat odo)

public int UbaciPomocna(OpstiDomenskiObjekat odo)

public List<IOOpstiDomenskiObjekat> VratiSvePomocna(IOOpstiDomenskiObjekat odo)

public bool Obrisi2(IOOpstiDomenskiObjekat odo)

Sve metode DBBroker klase su projektovane kao generičke, što znači da mogu da prihvate različite domenske objekte preko parametara. Na taj način postižemo da u DBBroker klasi nemamo implementaciju pojedinačnih metoda za svaku domensku klasu.

U procesu pravljenja DBBroker klase dobili smo metode interfejsa OpstiDomenskiObjekat. Svaka klasa iz domena implementira dati interfejs, i sve njegove metode. Na taj način je omogućeno da metode klase DBBroker budu generičke i da se DBBroker klasa povezuje sa interfejsom OpstiDomenskiObjekat.



Slika Veza između DBBrokera i OpštegDomenskogObjekta

Srednji sloj predstavlja klasa BrokerBaze sa sledecim metodama:

public bool Obrisi(IOOpstiDomenskiObjekat odo)

{

try

{

string upit = $" delete from {odo.vratiImeTabele()} where {odo.VratiUslovDva()} ";

OleDbCommand komanda3 = new OleDbCommand(upit, konekcija, transakcija);

komanda3.CommandType = CommandType.Text;

komanda3.ExecuteNonQuery();

return true;

}

catch (Exception e)

{

MessageBox.Show(e.Message);

return false;

}

}

public List<IOOpstiDomenskiObjekat> VratiSveZaUslovDva(IOOpstiDomenskiObjekat odo)

{

string upit = $"Select \* from {odo.vratiImeTabele2()} where {odo.VratiUslov3()}";

OleDbCommand komanda2 = new OleDbCommand(upit, konekcija, transakcija);

komanda2.CommandType = CommandType.Text;

OleDbDataReader citac1 = komanda2.ExecuteReader();

return odo.VratiListu(citac1);

}

public bool Azuriraj(IOOpstiDomenskiObjekat odo)

{

try

{

komanda.CommandText = $" update {odo.vratiImeTabele()} set {odo.vratiAzuriranje()} where {odo.VratiUslovDva()} ";

komanda.CommandType = CommandType.Text;

komanda.ExecuteNonQuery();

return true;

}

catch (Exception e)

{

MessageBox.Show(e.Message);

return false;

}

}

public static BrokerBaze DajInstancu()

{

if (instanca == null)

{

instanca = new BrokerBaze();

}

return instanca;

}

public int VratiSifru(IOOpstiDomenskiObjekat odo)

{

komanda.CommandText = "SELECT MAX(" + odo.vratiKljuc() + ") FROM " + odo.vratiImeTabeleZaKljuc();

komanda.CommandType = CommandType.Text;

try

{

int sifra = Convert.ToInt32(komanda.ExecuteScalar());

return sifra + 1;

}

catch (Exception )

{

return 1;

}

}

public int Ubaci(IOOpstiDomenskiObjekat odo)

{

try

{

komanda.CommandText = $"Insert into {odo.vratiImeTabele()} Values ({odo.vratiInsert()})";

komanda.CommandType = CommandType.Text;

int rezultat = komanda.ExecuteNonQuery();

return rezultat;

}

catch (Exception e)

{

MessageBox.Show(e.Message);

throw e;

}

}

public List<IOOpstiDomenskiObjekat> VratiSve(IOOpstiDomenskiObjekat odo)

{

komanda.CommandText = $"Select \* from {odo.vratiImeTabele()}";

komanda.CommandType = CommandType.Text;

OleDbDataReader citac = komanda.ExecuteReader();

return odo.VratiListu(citac);

}

BrokerBaze()

{

konekcija = new OleDbConnection(@"Provider = SQLOLEDB; Data Source = MC0XH0VC\EDUSQL; Initial Catalog = EdukativniCentarBaza; Integrated Security = SSPI");

komanda = konekcija.CreateCommand();

}

public void Commit()

{

transakcija.Commit();

}

public void Rollback()

{

transakcija.Rollback();

}

public void OtvoriKonekciju()

{

konekcija.Close();

konekcija.Open();

}

public void ZatvoriKonekciju()

{

konekcija.Close();

}

public void PokreniTransakciju()

{

transakcija = konekcija.BeginTransaction();

komanda.Transaction = transakcija;

}

### 3.4.3 Prezentacioni sloj

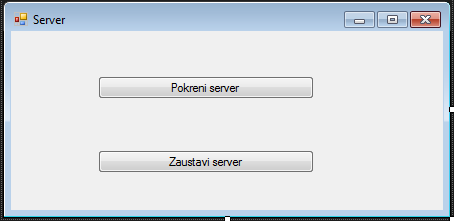
Korisnički interfejs definisan je preko skupa ekranskih formi. Scenarija slučajeva korisćenja ekranskih formi je direktno povezan sa scenarijima slučajeva korišćenja.

Postoje dva aspekta projektovanja ekranske forme:

1. Projektovanje scenarija slučajeva korišćenja koji se izvode preko ekranske forme

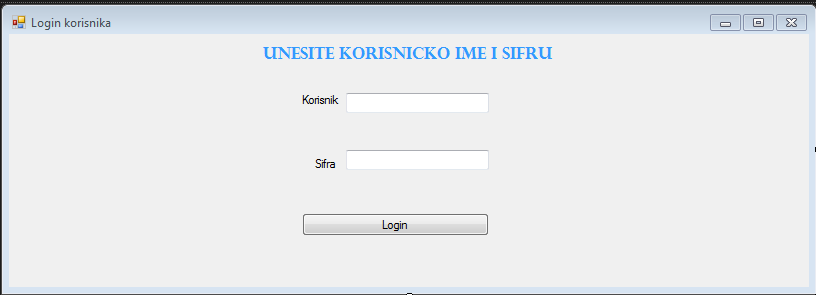
2. Projektovanje metoda ekranske forme

Glavna ekranska forma serverskog dela aplikacije izgleda ovako:



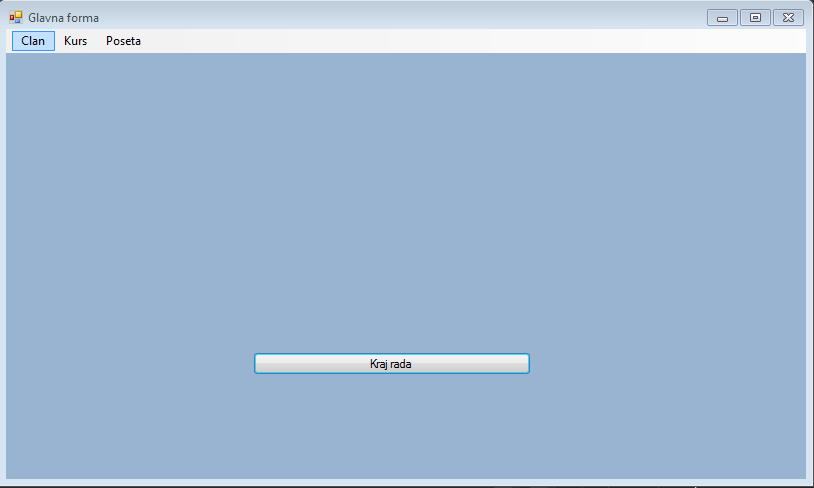
Slika Pokretanje servera

Glavna ekranska forma klijentskog dela aplikacije izgleda ovako:



Slika Login korisnika

Glavna ekranska forma klijentskog dela aplikacije kada se administrator uloguje izgleda ovako:



Slika Glavna forma

**SK1: Slučaj korišćenja – Ubacivanje novog člana**

**Naziv SK**:

Ubacivanje novog *člana*

**Akteri SK:**

**Radnik**

**Učesnici SK:**

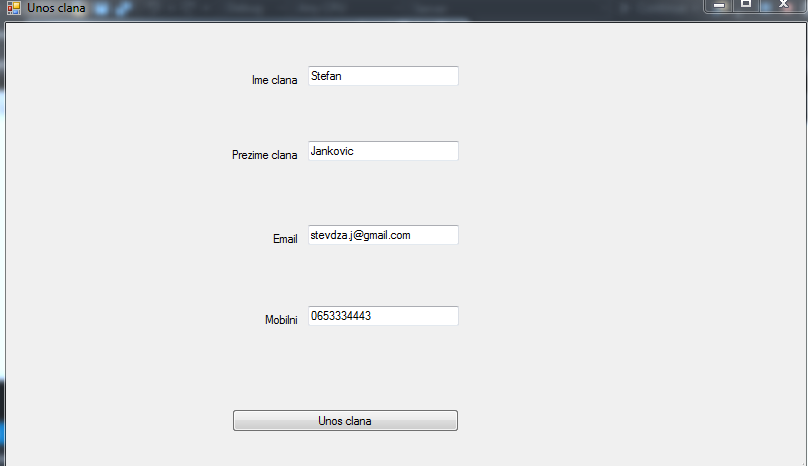
**Radnik** i **program**(**sistem**)

Preduslov: **Sistem** je uključen. Otvorena je forma za unos *članova*.

**Osnovni scenario SK:**

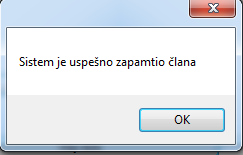
1. **Radnik** unosi podatke o novom *članu*. (APUSO)
2. **Radnik** poziva sistem da zapamti novog *člana* (APSO)

Opis akcije: Administrator klikom na dugme Unos clana poziva sistemsku operaciju UnesiClana(Clan) koja pamti novog clana.



Slika Unos clana

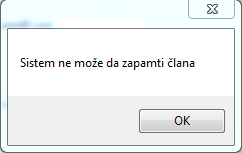
1. **Sistem** pamti novog *člana*. (SO)
2. **Sistem** pokazuje radniku poruku: „**Sistem** je uspešno zapamtio *člana* “. (IA)



Slika Obavestenje o uspešnosti operacije

**Alternativni scenario**:

* 1. Ukoliko **sistem** ne može da zapamti novog *člana*, prikazuje se poruka: „**Sistem** ne može da zapamti *člana* “. (IA)



Slika Obaveštenje o neuspešnosti operacije

public partial class UnosClanaForma : Form

{

Clan clan = new Clan();

public UnosClanaForma()

{

InitializeComponent();

}

private void UnosClanaForma\_Load\_1(object sender, EventArgs e)

{

}

private void btnUnos\_Click\_1(object sender, EventArgs e)

{

//Dodeljujemo vrednosti novom clanu

clan.Ime = txtIme.Text;

clan.Prezime = txtPrezime.Text;

clan.Mobilni = Convert.ToString(txtMobilni.Text);

clan.EMail = txtEmail.Text;

if(clan.Ime == null || clan.Ime == "")

{

MessageBox.Show("Morate uneti ime clana!");

return;

}

if(clan.Ime.Any(char.IsDigit))

{

MessageBox.Show("Ime ne sme sadrzati brojeve!");

return;

}

char[] nizKaraktera = clan.Ime.ToCharArray();

foreach(char c in nizKaraktera)

{

if(!Char.IsLetterOrDigit(c))

{

MessageBox.Show("Ime člana mora biti samo naziv!");

return;

}

}

if (clan.Ime.Any(char.IsDigit))

{

MessageBox.Show("Ime ne sme sadržati brojeve!");

return;

}

if (clan.Prezime == null || clan.Prezime == "")

{

MessageBox.Show("Morate uneti prezime člana!");

return;

}

if (clan.Prezime.Any(char.IsDigit))

{

MessageBox.Show("Prezime ne sme sadržati brojeve!");

return;

}

char[] nizKaraktera1 = clan.Prezime.ToCharArray();

foreach (char c in nizKaraktera1)

{

if (!Char.IsLetterOrDigit(c))

{

MessageBox.Show("Prezime mora biti samo naziv!");

return;

}

}

if (clan.Mobilni == null || clan.Mobilni == "" )

{

MessageBox.Show("Morate uneti mobilni člana! ");

return;

}

if(!clan.Mobilni.All(char.IsDigit))

{

MessageBox.Show("Mobilni sadrži samo brojeve!");

return;

}

if (clan.EMail == null || clan.EMail == "")

{

MessageBox.Show("Morate uneti email člana!");

return;

}

if(!clan.EMail.Contains('@') || !clan.EMail.EndsWith(".com"))

{

MessageBox.Show("Neispravna email adresa!");

return;

}

if (KontrolerKI.VratiInstancu().UbaciClana(clan))

{

MessageBox.Show("Sistem je uspešno zapamtio člana");

}

else

{

MessageBox.Show("Sistem ne može da zapamti člana");

}

}

private void button1\_Click(object sender, EventArgs e)

{

this.Close();

}

}

**SK 2: Slučaj korišćenja – Ubacivanje novog kursa**

**Naziv SK:**

Ubacivanje novog *kursa*

**Akteri SK:**

**Radnik**

**Učesnici SK:**

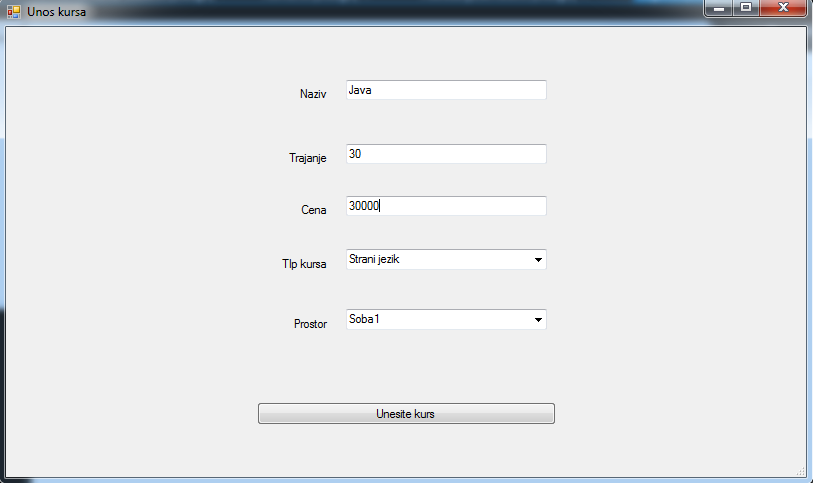
**Radnik** i **sistem**

**Preduslov**: **Sistem** je uključen. Otvorena je forma za unos *kursa*. Učitani su svi tipovi *kurseva*.

**Osnovni scenario SK:**

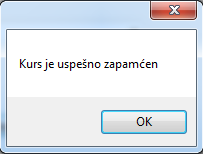
1. **Radnik** unosi podatke o novom *kursu*. (APUSO)
2. **Radnik** poziva **sistem** da zapamti novi *kurs*. (APSO)

Opis akcije: Randik pritiskom na dugme “Unesite kurs” poziva operaciju ZapamtiKurs(kurs) koja pamti novi kurs.



Slika Forma za unos kursa

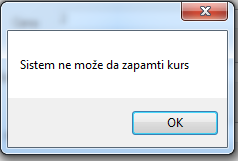
1. **Sistem** pamti novi *kurs*. (SO)
2. **Sistem** prikazuje **radniku** poruku: „*Kurs* je uspešno zapamćen“. (IA)



Slika Obaveštenje o uspešnosti operacije

**Alternativni scenario:**

* 1. Ukoliko **sistem** ne može da zapamti novi *kurs*, prikazuje se poruka: „**Sistem** ne može da zapamti *kurs* “. (IA)



Slika Obaveštenje o neuspešnosti operacije

public partial class UnosNovogKursa : Form

{

public Kurs kurs;

// public List<TipKursa> listaTipova;

private List<TipKursa> listaTipova;

public TipKursa tipKursa;

public Ucionica prostor;

public List<TipKursa> ListaTipova

{

get { return listaTipova; }

set { listaTipova = value; }

}

List<string> listaImenaTipova;

public UnosNovogKursa()

{

InitializeComponent();

}

private void UnosNovogKursa\_Load(object sender, EventArgs e)

{

//vracanje tipova kursa u combobox

listaImenaTipova = new List<string>();

}

private void button1\_Click\_1(object sender, EventArgs e)

{

kurs = new Kurs();

int errorCounter = 0;

if (txtCena.Text == null || txtCena.Text == "")

{

MessageBox.Show("Morate uneti cenu");

txtCena.Clear();

return;

}

if (txtNaziv.Text == null || txtNaziv.Text == "")

{

MessageBox.Show("Morate uneti naziv");

txtNaziv.Clear();

return;

}

if (txtNaziv.Text.Any(char.IsDigit))

{

MessageBox.Show("Naziv ne sme sadrzati brojeve!");

return;

}

if (txtTrajanje.Text == null || txtTrajanje.Text == "")

{

MessageBox.Show("Morate uneti trajanje");

txtTrajanje.Clear();

return;

}

if(kurs == null)

{

MessageBox.Show("Morate uneti kurs!");

return;

}

if ( prostor == null)

{

MessageBox.Show("Morate uneti učionicu!");

return;

}

errorCounter = Regex.Matches(txtCena.Text, @"[a-zA-Z]").Count;

if(errorCounter > 0)

{

MessageBox.Show("Cena ne sme da sadrzi slova!");

return;

}

int errorCounter1;

errorCounter1 = Regex.Matches(txtTrajanje.Text, @"[a-zA-Z]").Count;

if (errorCounter > 0)

{

MessageBox.Show("Trajanje ne sme da sadrzi slova!");

return;

}

char[] nizKaraktera = txtTrajanje.Text.ToCharArray();

foreach (char c in nizKaraktera)

{

if (!Char.IsLetterOrDigit(c))

{

MessageBox.Show("Trajanje mora biti ceo broj!");

return;

}

}

char[] nizKaraktera2 = txtCena.Text.ToCharArray();

foreach (char c in nizKaraktera2)

{

if (!Char.IsLetterOrDigit(c))

{

if (c == '.')

{

break;

}

else

{

MessageBox.Show("Cena mora biti broj!");

return;

}

}

}

if (txtTrajanje.Text.Count(x => Char.IsDigit(x) )>6)

{

MessageBox.Show("Nedozvoljeno trajanje jednog kursa!");

return;

}

kurs.Cena = txtCena.Text;

// kurs.Cena = Convert.ToDouble(txtCena.Text);

kurs.Naziv = txtNaziv.Text;

kurs.Prostor = prostor;

kurs.Trajnje = Convert.ToInt32(txtTrajanje.Text);

kurs.TipKursa = tipKursa;

KontrolerKI.VratiInstancu().ZapamtiKurs(kurs);

txtCena.Clear();

txtTrajanje.Clear();

txtNaziv.Clear();

txtProstor.Clear();

txtTipoviKursa.Clear();

}

private void txtTipoviKursa\_TextChanged(object sender, EventArgs e)

{

}

public void postaviVrednostZaTip(TipKursa tip)

{

tipKursa = tip;

txtTipoviKursa.Text = tipKursa.NazivTipa;

}

private void btnTipoviKursa\_Click(object sender, EventArgs e)

{

string filter = txtTipoviKursa.Text;

TipoviKurseva tipForma = new TipoviKurseva(filter);

tipForma.Show(this);

}

private void btnProstori\_Click(object sender, EventArgs e)

{

ProstorForma pf = new ProstorForma();

pf.Show(this);

}

private void button2\_Click(object sender, EventArgs e)

{

this.Close();

}

**SK 3: Slučaj korišćenja – Izmena kursa**

**Naziv SK:**

Izmena postojećeg *kursa*

**Akteri SK:**

**Radnik**

**Učesnici SK:**

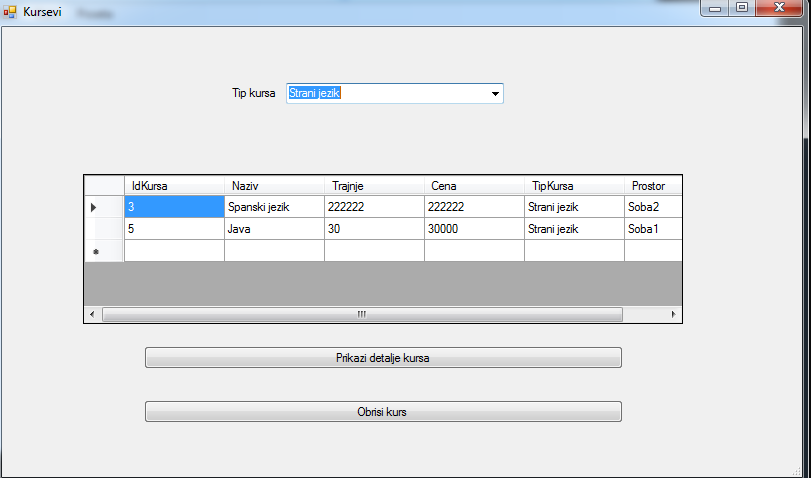
**Radnik** i **sistem**

**Preduslov**: **Sistem** je uključen i **radnik** je ulogovan pod svojom šifrom. Otvorena je forma za izmenu *kursa*. Učitani su svi tipovi *kurseva*.

**Osnovni scenario SK:**

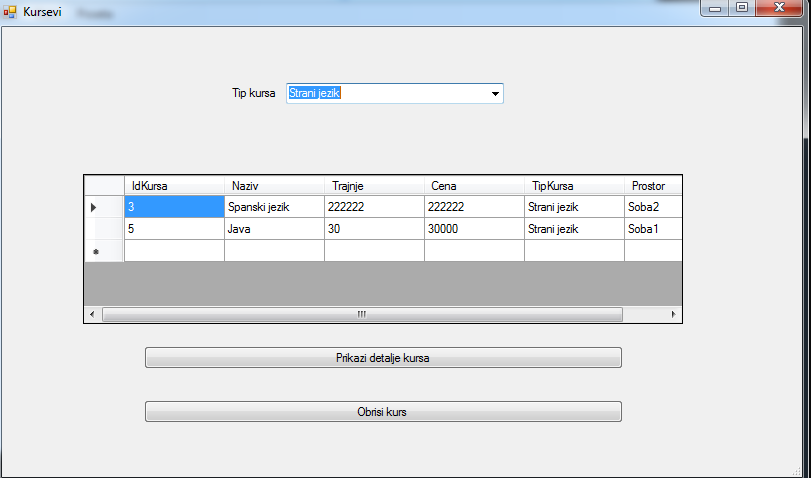
1. **Radnik** bira tip *kursa* iz spiska *kurseva*. (APUSO)
2. **Radnik** poziva **sistem** da pronadje kurseve za izabrani tip *kursa*. (APSO)

Opis akcije: Administrator unosom kriterijuma pretrage automatski poziva sistemsku operaciju PronađiKurs(kriterijumPretrage, List<Kurs>) koja pronalazi kurseve



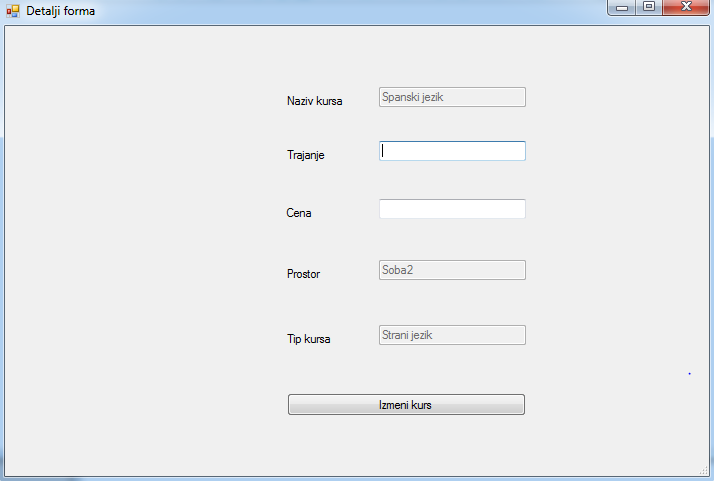
Slika Forma za rad sa kursevima

1. **Sistem** pronalazi *kurseve*. (SO)
2. **Sistem** prikazuje listu *kurseva*. (IA)
3. **Radnik** bira *kurs* iz spiska *kurseva*. (APUSO)



Slika Forma za rad sa kursevima 2

1. **Sistem** prikazuje podatke o izabranom *kursu*. (IA)
2. **Radnik** menja podatke *kursa*. (APUSO)

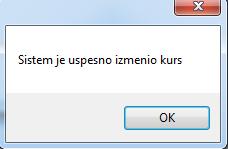


Slika Detalji kursa forma

1. **Radnik** poziva sistem da ažurira *kurs*. (APSO)

Opis akcije: Radnik klikom na dugme Izmeni kurs poziva sistemsku operaciju IzmeniKurs(Kurs) koja azurira informacije o kursu.

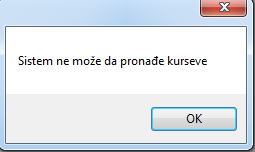
1. **Sistem** ažurira *kurs*. (SO)
2. **Sistem** prikazuje poruku da je *kurs* ažuriran. (IA)



Slika Obaveštenje o uspešnosti operacije

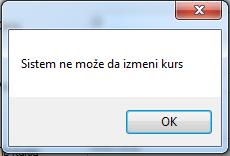
**Alternativni scenario:**

2.1 Ukoliko **sistem** ne može da pronađe *kurseve*, prikazuje se poruka: „**Sistem** ne može da pronađe *kurseve* “. Prekida se izvršenje scenarija. (IA)



Slika Obaveštenje o neuspešnosti operacije 1

8.1 Ukoliko **sistem** ne može da izmeni *kurs*, prikazuje se poruka: „**Sistem** ne može da izmeni *kurs*“. (IA)



Slika Obaveštenje o neuspešnosti operacije 2

public partial class DetaljiKursaForma : Form

{

Kurs kurs;

DataGridView dgvKursevi;

public DetaljiKursaForma(Kurs kurs, DataGridView dgvKursevi)

{

InitializeComponent();

this.kurs = kurs;

this.dgvKursevi = dgvKursevi;

}

private void DetaljiKursaForma\_Load(object sender, EventArgs e)

{

//ucitavamo podatke kursa i punimo polja

txtNaziv.Text = kurs.Naziv;

txtTipKursa.Text = kurs.TipKursa.NazivTipa;

txtProstor.Text = kurs.Prostor.Naziv;

}

private void button1\_Click\_1(object sender, EventArgs e)

{

// vrsimo potrebne validacije

if (txtCena.Text == "" || txtCena.Text == null)

{

MessageBox.Show("Morate uneti cenu");

return;

}

int errorCounter = Regex.Matches(txtCena.Text, @"[a-zA-Z]").Count;

if (errorCounter > 0)

{

MessageBox.Show("Cena ne sme da sadrzi slova!");

return;

}

int errorCounter1 = Regex.Matches(txtTrajanje.Text, @"[a-zA-Z]").Count;

if (errorCounter1 > 0)

{

MessageBox.Show("Trajanje ne sme da sadrzi slova!");

return;

}

char[] nizKaraktera = txtTrajanje.Text.ToCharArray();

foreach(char c in nizKaraktera)

{

if (!Char.IsLetterOrDigit(c))

{

MessageBox.Show("Trajanje mora biti ceo broj!");

return;

}

}

char[] nizKaraktera2 = txtCena.Text.ToCharArray();

foreach (char c in nizKaraktera2)

{

if (!Char.IsLetterOrDigit(c))

{

if (c == '.')

{

break;

}

else

{

MessageBox.Show("Cena mora biti broj!");

return;

}

}

}

kurs.Cena = txtCena.Text;

if (txtTrajanje.Text == "" || txtTrajanje.Text == null)

{

MessageBox.Show("Morate uneti cenu");

return;

}

kurs.Trajnje = Int32.Parse(txtTrajanje.Text);

//saljemo zahtev kontroleru za izmenu podataka kursa

KontrolerKI.VratiInstancu().AzurirajKurs(kurs, dgvKursevi);

this.Close();

}

private void button2\_Click(object sender, EventArgs e)

{

this.Close();

}

}

**SK 4: Slučaj korišćenja – Pregled kurseva**

**Naziv SK:**

Pregled *kurseva*

**Akteri SK:**

**Radnik**

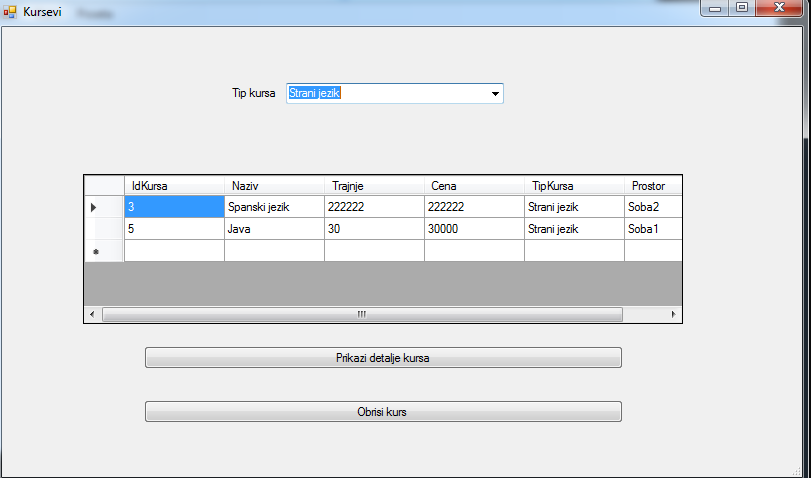
**Učesnici SK:**

**Radnik** i **sistem**

**Preduslov**: **Sistem** je uključen. Otvorena je forma za pregled *kurseva*. Učitani su svi tipovi *kurseva*.

**Osnovni scenario SK:**

1. **Radnik** bira tip *kursa* iz spiska svih tipova *kurseva*.(APUSO)

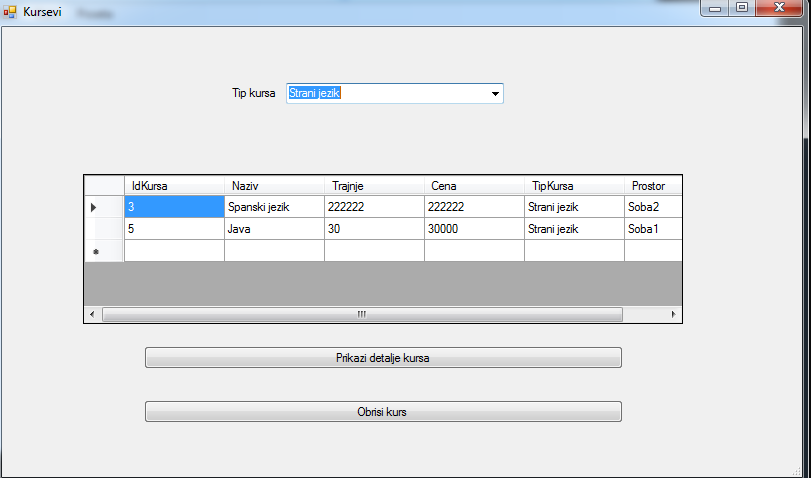


Slika Forma za rad sa kursevima 3

1. **Radnik** poziva **sistem** da pronadje *kurseve* za izabrani tip *kursa*. (APSO)

Opis akcije: Radnik unosom kriterijuma pretrage automatski poziva sistemsku operaciju PronadjiKurs(kriterijumPretrage, List<Kurs>) koja pronalazi kurseve.

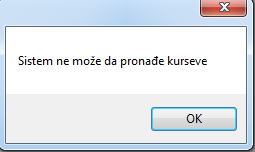
1. **Sistem** pronalazi *kurseve*. (SO)
2. **Sistem** prikazuje listu *kurseva* i poruku: „**Sistem** je uspešno pronašao *kurseve* “. (IA)



Slika Forma za rad sa kursevima 4

**Alternativni scenario:**

* 1. Ukoliko **sistem** ne može da pronađe *kurseve* za određeni tip, prikazuje se poruka: „**Sistem** ne može da pronađe *kurseve* “. (IA)



Slika Obaveštenje o neuspešnosti operacije

public partial class FormaZaRadSaKursevima : Form

{

Kurs kurs;

List<TipKursa> listaTipova;

public TipKursa tip;

public FormaZaRadSaKursevima()

{

InitializeComponent();

}

private void FormaZaRadSaKursevima\_Load(object sender, EventArgs e)

{

//vracamo tipove kursa

listaTipova = new List<TipKursa>();

}

private void btnPrikaziDetalje\_Click\_1(object sender, EventArgs e)

{

try

{

kurs = dgvKursevi.CurrentRow.DataBoundItem as Kurs;

DetaljiKursaForma dkf = new DetaljiKursaForma(kurs, dgvKursevi);

dkf.ShowDialog();

}

catch(Exception )

{

MessageBox.Show("Nije izabran kurs!");

return;

}

dgvKursevi.Columns[5].HeaderCell.Value = "Broj prisustva";

dgvKursevi.Columns[4].HeaderCell.Value = "Tip kursa";

}

private void cmbTipoviKursa\_SelectedIndexChanged\_1(object sender, EventArgs e)

{

}

private void btnObrisi\_Click\_1(object sender, EventArgs e)

{

try

{

kurs = dgvKursevi.CurrentRow.DataBoundItem as Kurs;

KontrolerKI.VratiInstancu().ObrisiKurs(kurs, dgvKursevi);

}

catch (Exception )

{

MessageBox.Show("Sistem ne može da izbriše izabrani kurs");

}

}

private void btnTipovi\_Click(object sender, EventArgs e)

{

TipoviKursevaZaRad tkzr = new TipoviKursevaZaRad();

tkzr.Show(this);

}

private void btnPretraga\_Click(object sender, EventArgs e)

{

if(tip == null)

{

MessageBox.Show("Morate izabrati tip!");

return;

}

try

{

KontrolerKI.VratiInstancu().PrikaziKurseve(dgvKursevi,tip);

}

catch

{

MessageBox.Show("Sistem ne može da pronađe kurseve");

}

}

private void button1\_Click(object sender, EventArgs e)

{

this.Close();

}

}

**SK 6: Slučaj korišćenja – Unos prisustva**

**Naziv SK:**

Unos novog *prisustva*

**Akteri SK:**

**Radnik**

**Učesnici SK:**

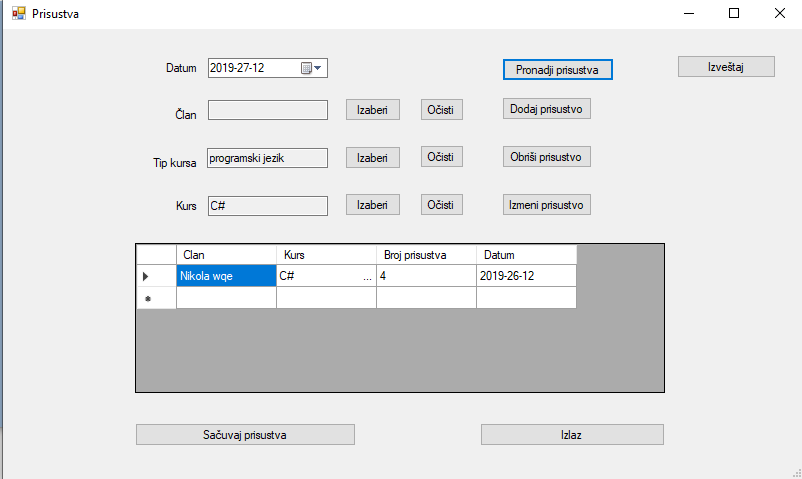
**Radnik** i **sistem**

**Preduslov**: **Sistem** je uključen. Otvorena je forma za ubacivanje novog *prisustva*. Učitani su svi članovi, tipovi kursa, kursevi i njihova *prisustva*.

**Osnovni scenario SK:**

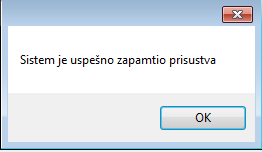
* 1. **Radnik** unosi podatke o novom prisustvu člana. (APUSO)
  2. **Radnik** poziva **sistem** da zapamti novo *prisustvo*. (APSO)

Opis akcije: Administrator klikom na dugme Sacuvaj posete poziva sistemsku operaciju ZapamtiPrisustvo(Prisustvo) koja pamti nova prisustva.



Slika Forma za rad sa prisustvima SK 6

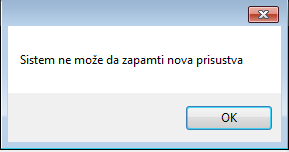
* 1. **Sistem** pamti nova *prisustva*. (SO)
  2. **Sistem** prikazuje radniku poruku: „**Sistem** je uspešno zapamtio *prisustva*“. (IA)



Slika Obaveštenje o uspešnosti operacije SK6

**Alternativni scenario:**

* 1. Ukoliko **sistem** ne može da zapamti nove *posete*, prikazuje se poruka: „**Sistem** ne može da zapamti nova *prisustva* “. (IA)

****

Slika Obaveštenje o neuspešnosti operacije SK6

public partial class RadSaPosetamaForma : Form

{

List<TipKursa> listaTipova;

public Clan clan;

public TipKursa tip;

public Kurs kurs;

string filename;

public RadSaPosetamaForma()

{

InitializeComponent();

}

private void RadSaPosetamaForma\_Load(object sender, EventArgs e)

{

listaTipova = new List<TipKursa>();

dgvPosete.DataSource = clan.ListaPoseta;

dgvPosete.Columns[3].DefaultCellStyle.Format = "yyyy-dd-MM";

dgvPosete.Columns[0].HeaderCell.Value = "Clan";

dgvPosete.Columns[1].HeaderCell.Value = "Kurs";

dgvPosete.Columns[2].HeaderCell.Value = "Broj posete";

}

private void btnDodaj\_Click\_1(object sender, EventArgs e)

{

Poseta poseta = new Poseta

{

Clan = clan,

Datum = dtpDatum.Value.Date,

Kurs = kurs,

BrojPrisustva = 0

};

if (clan == null)

{

MessageBox.Show("Morate izabrati clana!");

return;

}

if(kurs == null)

{

MessageBox.Show("Morate izabrati kurs!");

return;

}

clan.ListaPoseta.Add(poseta);

dgvPosete.DataSource = clan.ListaPoseta

;

dgvPosete.Columns[3].DefaultCellStyle.Format = "yyyy-dd-MM";

dgvPosete.Columns[0].HeaderCell.Value = "Clan";

dgvPosete.Columns[1].HeaderCell.Value = "Kurs";

dgvPosete.Columns[2].HeaderCell.Value = "Broj posete";

}

private void cmbTipKursa\_SelectedIndexChanged\_1(object sender, EventArgs e)

{

// KontrolerKI.VratiInstancu().PrikaziKurseveZaPosete(cmbTipKursa, cmbKurs);

}

private void btnObrisi\_Click\_1(object sender, EventArgs e)

{

KontrolerKI.VratiInstancu().ObrisiPosetu(dgvPosete);

}

private void button1\_Click(object sender, EventArgs e)

{

try

{

Poseta p = dgvPosete.CurrentRow.DataBoundItem as Poseta;

if (p == null)

{

MessageBox.Show("Morate izabrati posetu!");

return;

}

FormaZaIzmenuPosete izmena = new FormaZaIzmenuPosete(p);

izmena.ShowDialog();

dgvPosete.Refresh();

}

catch (Exception )

{

MessageBox.Show("Morate izabrati prisustvo!");

return;

}

}

private void cmbClan\_SelectedIndexChanged(object sender, EventArgs e)

{

}

private void btnSacuvaj\_Click(object sender, EventArgs e)

{

bool uspesno;

uspesno = KontrolerKI.VratiInstancu().ZapamtiPosete(clan);

if (uspesno == true)

{

PronadjiPosete();

}

}

private void btnExport\_Click(object sender, EventArgs e)

{

Microsoft.Office.Interop.Excel.Application app = new Microsoft.Office.Interop.Excel.Application();

Microsoft.Office.Interop.Excel.Workbook workbook = app.Workbooks.Add(Type.Missing);

Microsoft.Office.Interop.Excel.Worksheet worksheet = null;

worksheet = workbook.Sheets["Sheet1"];

worksheet = workbook.ActiveSheet;

worksheet.Name = "PoseteClanova";

for (int i = 1; i <dgvPosete.Columns.Count+1; i++)

{

worksheet.Cells[1,i] = dgvPosete.Columns[i - 1].HeaderText;

}

for(int i = 0; i< dgvPosete.Rows.Count-1; i++)

{

for (int j = 0; j < dgvPosete.Columns.Count; j++)

{

worksheet.Cells[i + 2, j + 1] = dgvPosete.Rows[i].Cells[j].Value.ToString();

}

}

var saveFileDialog = new SaveFileDialog();

saveFileDialog.FileName = "output";

saveFileDialog.DefaultExt = ".xlsx";

if (saveFileDialog.ShowDialog() == DialogResult.OK)

{

workbook.SaveAs(saveFileDialog.FileName,

Type.Missing,

Type.Missing,

Type.Missing,

Type.Missing,

Type.Missing,

Microsoft.Office.Interop.Excel.XlSaveAsAccessMode.xlExclusive,

Type.Missing,

Type.Missing,

Type.Missing,

Type.Missing,

Type.Missing);

MessageBox.Show("Uspešno eksportovane posete");

}

app.Quit();

}

private void btnPretragaClanova\_Click(object sender, EventArgs e)

{

string filter = txtClanovi.Text;

PrikazClanovaZaPosete pczp = new PrikazClanovaZaPosete(filter);

pczp.Show(this);

}

private void btnPretragaTipova\_Click(object sender, EventArgs e)

{

string filter = txtTipovi.Text;

TipKursaZaPoseteForma tkzpf = new TipKursaZaPoseteForma();

tkzpf.Show(this);

}

private void btnPretragaKurseva\_Click(object sender, EventArgs e)

{

if(tip == null)

{

MessageBox.Show("Morate izabrati tip!");

return;

}

string nazivTipa = txtTipovi.Text;

KurseviZaPoseteForma kzpf = new KurseviZaPoseteForma(nazivTipa);

kzpf.Show(this);

}

public void PronadjiPosete()

{

if (txtKursevi.Text == "")

{

MessageBox.Show("Morate izabrati kurs!");

return;

}

if (txtClanovi.Text == "")

{

MessageBox.Show("Morate izabrati clana");

return;

}

KontrolerKI.VratiInstancu().VratiPoseteZaClana(clan, dtpDatum, kurs, dgvPosete);

dgvPosete.Columns[3].DefaultCellStyle.Format = "yyyy-dd-MM";

dgvPosete.Columns[0].HeaderCell.Value = "Clan";

dgvPosete.Columns[1].HeaderCell.Value = "Kurs";

dgvPosete.Columns[2].HeaderCell.Value = "Broj prisustva";

}

private void btnVratiPosete\_Click(object sender, EventArgs e)

{

PronadjiPosete();

}

private void button2\_Click(object sender, EventArgs e)

{

this.Close();

}

private void btnImport\_Click(object sender, EventArgs e)

{

String name = "PoseteClanova";

var filePath = string.Empty;

OpenFileDialog openFileDialog = new OpenFileDialog();

openFileDialog.Filter = "Excel Files|\*.xls;\*.xlsx;\*.xlsm";

openFileDialog.FilterIndex = 2;

openFileDialog.RestoreDirectory = true;

if (openFileDialog.ShowDialog() == DialogResult.OK)

{

filePath = openFileDialog.FileName;

}

try

{

String constr = "Provider=Microsoft.ACE.OLEDB.12.0; Data Source=" + filePath

+

";Extended Properties='Excel 12.0 XML;HDR=YES;IMEX=1;';";

OleDbConnection con = new OleDbConnection(constr);

OleDbCommand oconn = new OleDbCommand("Select \* From [" + name + "$]", con);

con.Open();

OleDbDataAdapter sda = new OleDbDataAdapter(oconn);

DataTable data = new DataTable();

sda.Fill(data);

dgvPosete.DataSource = data;

int counter = dgvPosete.Rows.Count;

con.Close();

MessageBox.Show("Uspesan import fajla!");

}

catch(Exception)

{

MessageBox.Show("Neuspesan import fajla!");

return;

}

}

private void btnStampaj\_Click(object sender, EventArgs e)

{

DateTime mesec = dtpDatum.Value;

List<Poseta> listaPoseta = new List<Poseta>();

listaPoseta = KontrolerKI.VratiInstancu().VratiPoseteZaMesec(mesec);

if ( listaPoseta == null)

{

MessageBox.Show("Ne postoje prisustva za ovaj mesec!");

return;

}

IzvestajForma izvesta = new IzvestajForma(mesec, listaPoseta);

izvesta.ShowDialog();

}

}

**SK 14: Slučaj korišćenja – Prikaz izveštaja**

**Naziv SK:**

Prikaz izveštaja

**Akteri SK:**

**Radnik**

**Učesnici SK:**

**Radnik** i **sistem**

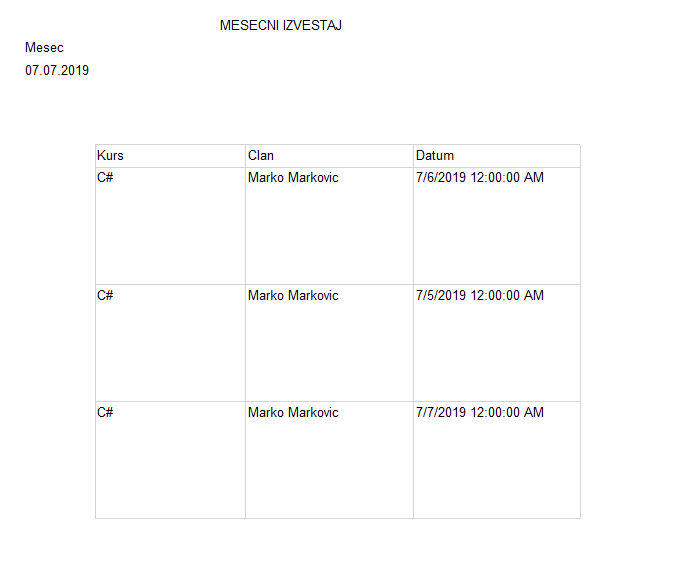
**Preduslov**: **Sistem** je uključen i **radnik** je ulogovan pod svojom šifrom. Otvorena je forma za rad sa *posetama*. Učitani su svi članovi i kursevi.

**Osnovni scenario SK:**

1. **Radnik** poziva sistem da izvrši prikaz izveštaja o kursevima za dati mesec. (APSO)

Opis akcije: Radnik unosom kriterijuma pretrage automatski poziva sistemsku operaciju PrikaziIzvestaj(kriterijumPretrage, List<Kurs>) koja pronalazi kurseve.

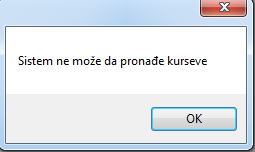
1. **Sistem** otvara prozor za izvetaj. (SO)
2. **Sistem** prikazuje listu *kurseva*. (IA)

****

Slika Prikaz izveštaja

**Alternativni scenario:**

* 1. Ukoliko **sistem** ne može da pronađe *listu*, prikazuje se poruka: „**Sistem** ne može da pronađe *listu kurseva* “. Prekida se izvršenje scenarija. (IA)



Slika Obaveštenje o neuspešnosti operacije SK 14

public partial class IzvestajForma : Form

{

List<Poseta> lista;

DateTime mesec;

public IzvestajForma(DateTime mesec, List<Poseta> lista)

{

InitializeComponent();

this.mesec = mesec;

this.lista = lista;

}

private void IzvestajForma\_Load(object sender, EventArgs e)

{

int broj;

PosetaBindingSource.DataSource = lista;

broj = lista.Count;

Microsoft.Reporting.WinForms.ReportParameter[] p = new Microsoft.Reporting.WinForms.ReportParameter[]

{

new Microsoft.Reporting.WinForms.ReportParameter("Mesec", mesec.ToShortDateString()),

new Microsoft.Reporting.WinForms.ReportParameter("Iznos", mesec.ToString()),

new Microsoft.Reporting.WinForms.ReportParameter("BrojKurseva", broj.ToString())

};

this.reportViewer1.LocalReport.SetParameters(p);

this.reportViewer1.RefreshReport();

}

}

# Testiranje

Svaki od implementiranih slučajeva korišćenja je testiran. Prilikom testiranja svakog slučaja korišćenja, pored unetih pravilnih podataka, unošeni su i nepravilni podaci da bi se utvrdilo kakav će biti rezultat izvršenja. Na osnovu izvršenih testiranja otklonjeni su uočeni nedostaci.

# Zaključak

U ovom radu opisani su osnovni koncepti C#-a kao i objektno-orijentisanog programiranja. Predstavljene su prednosti i mane C# kao programskog jezika. Pričano je i o konkurentnom i mrežnom programiranju, koncept niti i soketa je detaljno objašnjen.

Zatim je kroz faze Larmanove metode opisano projektovanje softverskog sistema za administraciju edukativnog centra. Vodi se računa o prisustvima, njihovim kursevima, kao i članovima koji te kurseve posećuju.

Jedan od zahtevnijih zadataka predstavljao je rešavanje problema čuvanja C# klasa u relacionu bazu podataka i prevazilaženje problema na koje sam nailazio. Moguća je potpuna administracija svih pomenutih objekata i dato rešenje se može koristiti u realnom sistemu.

Početak projektovanja softvera predstavljao je jedan od izazovnijih zadataka. Reč je od dve faze: prikupljanje zahteva i analiza. Bilo je potrebno osmisliti funkcionalnosti aplikacije koje bi trebalo da budu bliske realnim potrebama korisnika. Pored toga bitno je lako snalaženje prilikom korišćenja ove aplikacije.

Takođe je moguće unaprediti ovaj rad, u najvećoj meri u oblasti testiranja softvera. Kroz rad nisu implementirani automatizovani, i to je domen gde ova aplikacija treba da ide u budućnosti i da se razvija.

# Literatura

[1] dr Siniša Vlajić, Projektovanje softvera (skripta), Beograd, 2012.

[2] <https://docs.microsoft.com/en-us/dotnet/csharp/getting-started/introduction-to-the-csharp-language-and-the-net-framework>

[3] <https://en.wikipedia.org/wiki/C_Sharp_(programming_language)>

[4] <https://searchmicroservices.techtarget.com/definition/object-oriented-programming-OOP>

[5] Siniša Vlajić, Dušan Savić, Vojislav Stanojević, Ilija Antović, Miloš Milić,

„Projektovanje softvera– Napredne Java tehnologije “, godina izdavanja: 2008.

[6] dr Siniša Vlajić, „Softverski paterni “, godina izdavanja: 2014.

[7] Branislav Lazarević, Zoran Marjanović, Nenad Aničić, Slađan Babarogić „Baze

podataka “, godina izdavanja: 2012.

[8] dr Siniša Vlajić, „Projektovanje softvera – skripta “, godina izdavanja: 2015