Факултет инжењерских наука

Универзитета у Крагујевцу

Назив студијског програма: Машинско инжењерство

Ниво студија: Мастер академске студије

Модул: Информатика у инжењерству

Предмет: Пројектовање информационих система и база података

Број индекса: 384/2015

Спасоје И. Радојевић

ИЗРАДА ПРОЈЕКТНЕ ДОКУМЕНТАЦИЈЕ ИНФОРМАЦИОНОГ СИСТЕМА ФИРМЕ ФОРИ ТЕКСТИЛ

Мастер рад

Комисија за преглед и одбрану:

1. **Проф. др Милан Ерић - ментор**
2. _____
3. _____

Датум одбране:_____

Оцена:_____

У оквиру мастер рада са темом "*Израда пројектне документације информационог система фирме* **"*Фори текстил*"" кандидат треба да:**

На основу информационог система (апликативног софтвера и физичких база података) у делу набавке, производње и отпреме фирме "*Фори текстил*", за који не постоји пројектна документација, развије адекватну пројектну документацију. Рад треба да обухвати уводна разматрања везана за информационе технологије и развој информационих система, методолошке приступе и софтверске алате као подршке развоја, опис реалног система, развијене моделе и опис развијеног софтвера. На крају дати закључна разматрања и списак коришћене литературе.

Препоручена литература:

1. Полишчук Ј.: **Пројектовање информационих система**, Електротехнички факултет, Подгорица, 2007.
2. Rainer K., Turban E.: **Introduction to Information Systems**, John Wiley & Sons, Inc., 2009.
3. Вељовић А.: **Пројектовање информационих система**, Компјутер библиотека, 2003.
4. Лазаревић Б., ……: **Базе података**, ФОН Београд, Београд 2003.
5. Jon Reid, Jacob Hammer Pedersen, Ranga Raghunathan, Scott Allen, Syed Fahad Gilani:
6. **C# базе података**, CET Computer Equipment and Trade, Београд, 2003.
7. Michael Lee, Gentry Bieker: ***Mastering SQL Server 2008***, Wiley Publishing, Inc., Indianapolis, Indiana, 2009.
8. J.Sharp, **Microsoft Visual C# 2010**, Microsoft Press, 2010.
9. Tim Anderson, **C# in easy steps**, Published by Computer Step, 2014.

Крагујевац, јун 2017.

Ментор:

Проф. др Милан Д. Ерић

Изјава о самосталној изради рада

Изјављујем да сам овај мастер рад урадио самостално, служећи се стеченим знањем и искуством као и наведеном литературом.

384/2015 Спасоје Радојевић

_____

(потпис)

Резиме

У овом раду дат је поступак пројектовања информационих система. Пресликавање реалног система у адекватан логички систем применом одређених правила. Дата је детаљна анализа реалног система која обухвата идентификацију информација које фигуришу у оквиру система, њихов међусобни однос и утицај, као и веза између тих информација и спољног дела система. У раду је такође описан рад апликације коју смо одрадили у циљу симулирања већ постојећег система у предузећу "Фори Текстил".

Кључне речи: Информациони системи, реалан систем, логички систем, апликација, Фори Текстил.

Abstract

In this work is given process of designing information systems. Copying the real system into an adequate logical system using certain rules. A detailed analysis of the real system is included in the work, which includes the identification of the information that they represent within the system, their interaction and influence, as well as the connection between this information and the external part of the system. In this work also was described the work of the application that we have done in order to simulate the already existing system in “Fori Textile” company.

Keywords: Information system, Real system, logical system, application, Fori Textile

Садржај

1. Увод

Поступак пројектовања информационих система подразумева пресликавање посматраног реалног система у адекватан логички систем, применом одређених правила. Процесу пресликавања претходи детаљна анализа реалног система, која обухвата идентификацију информација које фигуришу у оквиру система, њихов међусобни однос и утицај, као и веза између тих информација и спољног дела система. Резултат добијен у току анализе потребно је приказати у одговарајућем облику, применом одговарајућих метода и техника. Оваква врста приказа назива се моделирање. Поступак моделирања информационог система, обрађен у овом тексту, има за основ примену техника и метода стандарда IDEF0, са освртом на стандард IDEF1x. Заправо, IDEF (Intregrated DEFinition) представља фамилију метода пројектовања за структурни приступ моделирању одлука, акција и активности организација и система. IDEF је настао из веома добро заснованог графичког језика тзв. the Structured Analysis and Desing Technique (SADT). Полазећи од SADT, током 60-тих и 70-тих година, авијација SAD развила је хијерархијски метод за моделирање којим се могу анализирати активности и сагледати функционалне перспективе неког система. [1] Почетком 90-тих година, IDEF Users Group ствара стандарде IDEF0 за функционално моделирање посматраног система и IDEF1x (eXtend) за семантичко-информационо моделирање података, а у сарадњи са National Institutes for Standards and Tehnology (NIST). Ови стандарди су прихваћени и од стране International Organization of Standards (ISO).

2. IDEF стандарди

Техника IDEF моделирања базирана је на комбинацији графике и текста који су представљени на организован и систематичан начин да би се повећала разумљивост и прихваћена је као основа за спровођење инжењеринга и реинжењеринга пословних процеса. Састоји се од хијерархијског низа дијаграма који постепено приказују све више детаља о пословним функцијама и њиховој повезаности са осталим деловима система. Током 60-их и 70-их година прошлог века развијена је SADT (Structured Analysis and Desing Technique) техника моделирања. Авијација SAD-а је прихватила SADT технику, наставила даље да је развија као део ICAM (Integrated Computer Aided Manufacturing) програма, и дала јој назив IDEF, Intregrated DEFinition, технике моделирања. Први назив методе био је ICAM DEFinition, да би се касније назив променио у Integration DEFinition - IDEF. Циљ ICAM програма био је побољшање производне продуктивности применом компјутерских технологија. Графички приказ пословних процеса у некој организацији ефикасан је алат за увид у функционисање пословних процеса и природу елемената процеса и веза између њих. Већина постојећих алата за развој информационих система подразумева IDEF моделирање.

Ова метода моделирања пословних процеса има дефинисану формалну технологију давања назива процесима, дијаграмима и повратним везама на дијаграмима који су лаки за читање. Основни задатак моделирања пословних процеса је да се премосте разлике између инжењерског, пројектантског и корисничког приступа опису неког пословног процеса. У пракси се показало да је техника моделирања пословних процеса једна од најефектнијих техника за једнозначну комуникацију између корисника функција и пројектанта информационих система. Крајем прошлог века створени су стандарди IDEF0 за моделирање процеса и стандард IDEF1x (eXtend) као техника за информационо моделирање, семантичко моделирање података. IEEE (Institute of Electrical and Electronics) институт је покровитељ ових стандарда који су прихваћени од стране међународне организације за стандарде - ISO (International Organization of Standards). [1]
Тренутно IDEF стандарде развија, одржава и унапређује америчка фирма Knowledge Based Systems Inc. Као стратешки партнер америчке владе и војске.
IDEF методологија се користи за следеће активности: анализу система, пројектовање, унапређење и интеграцију система, односно за документовање, разумевање, пројектовање, анализу, планирање и интеграцију функција.

Прва генерација обухвата следеће методе:

- IDEF0 Function Modeling: моделирање функција/ процеса / активности,
- IDEF1 Information Modeling: моделирање информација,
- IDEF2 Simulation Model Design: симулацијско моделирање.

Друга генерација обухвата следећу методу:

- IDEF1X Data Modeling: моделирање података.

Трећа генерација обухвата следеће методе:

- IDEF3 Process Description Capture: опис процеса, дијаграм тока података
- IDEF4 Object-Oriented Design: објектно оријентисани дизајн,
- IDEF4/C++ object-oriented design: C++ објектно оријентисани дизајн,
- IDEF5 Ontology Description Capture: опис организацијске онотологије, моделирање ентитета и њихових међусобних односа

Делимично развијене IDEF методе:

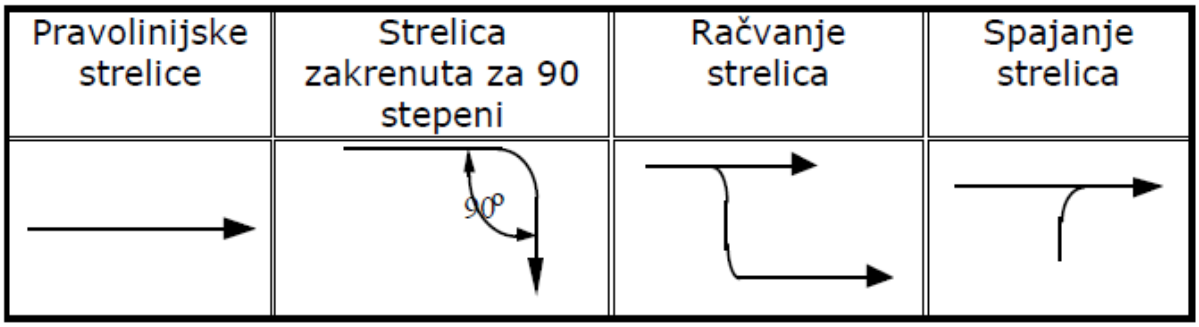
- IDEF6 Design Rationale Capture: опис логике дизајна,
- IDEF7 Information System Auditing: инспекција информационог система,
- IDEF8 User Interface Modeling: дизајн интеракције човек - систем,
- IDEF9 Business Constraint Discovery: откривање пословних ограничења,
- IDEF10 Information Artifact Modeling: моделирање информацијских медија
- IDEF11 Implementation Architecture Modeling: моделирање имплементације,

- IDEF12 Organization Modeling: моделирање организације,
- IDEF13 Three Schema Mapping Design: моделирање података са три шеме, интерном, концептуалном и екстерном.
- IDEF14 Network Design: дизајн мреже

Три су основна разлога за настанак IDEF0 стандарда: Потреба за документовањем комплексних пословних процеса, метода омогућава брзе организационе промене (модел процеса документује све важне активности и омогућава увид у критичне активности које треба извести са одговарајућим ресурсима), те потреба за прототипским приступом функционалном моделирању, где се на на брз и једноставан начин проверавају алтернативне идеје. Ово је веома важна карактеристика ове методе, јер брзи развој информационих технологија неминовно доводи до реинжењеринга пословних процеса. Метод за моделирање функција IDEF0 моделира понашање система, активности система и одлучивање у систему. Може се употребљавати за моделирање широког спектра аутоматизованих и неаутоматизованих система. Нула (0) у ознаци IDEF0 стандарда означава највиши ниво анализе. IDEF0 техника се састоји од хијерархијског низа дијаграма који постепено приказују све више детаља о процесима и њиховој вези са осталим деловима система. IDEF0 моделирање омогућава анализу особина одређеног пословног процеса ради његовог максималног унапређења, извршење функционалне декомпозиције на свим нивоима, стварање документације, паралелно са реинжењерингом пословних процеса, бољу комуникацију између пројектног тима, корисника и менаџера, управљање великим и сложеним пројектима и обезбеђење потребних елемената за моделирање информација (IDEF1x методологија). Графички језик IDEF0 описује методу функционалне декомпозиције преко скупа дијаграма. Сваки приказује ограничену количину детаља дефинисаних одговарајућом синтаксом и семантиком, при чему детаљност дијаграма зависи од његовог положаја у хијерархији модела. Поред графичког приказа, потребно је дати и прецизне дефиниције предмета који се појављују у моделу, као и попратни текст, који је критичан према моделу који има улогу средства комуникације. Овакав приступ наметнуо је потребу за апстракцијом, којом се изводи контролисано искључивање детаља, односно извлаче заједничке карактеристике у описивању неког система. Тако је на вишим нивоима апстракције систем описан јасније, а на нижим детаљније. Поступак моделирања реалног система зависи од способности, знања и искуства пројектног тима, јер се не могу дати строга формална правила моделирања која би водила до јединственог модела сложеног реалног система, без обзира на то ко врши моделирање. Синтаксу графичког језика IDEF0 чине правугаоници и стрелице. Правоугаоници омогућују опис активности (функције, процеса, трансформације) (слика 1) тј. онога што се дешава у дизајнираној функцији.

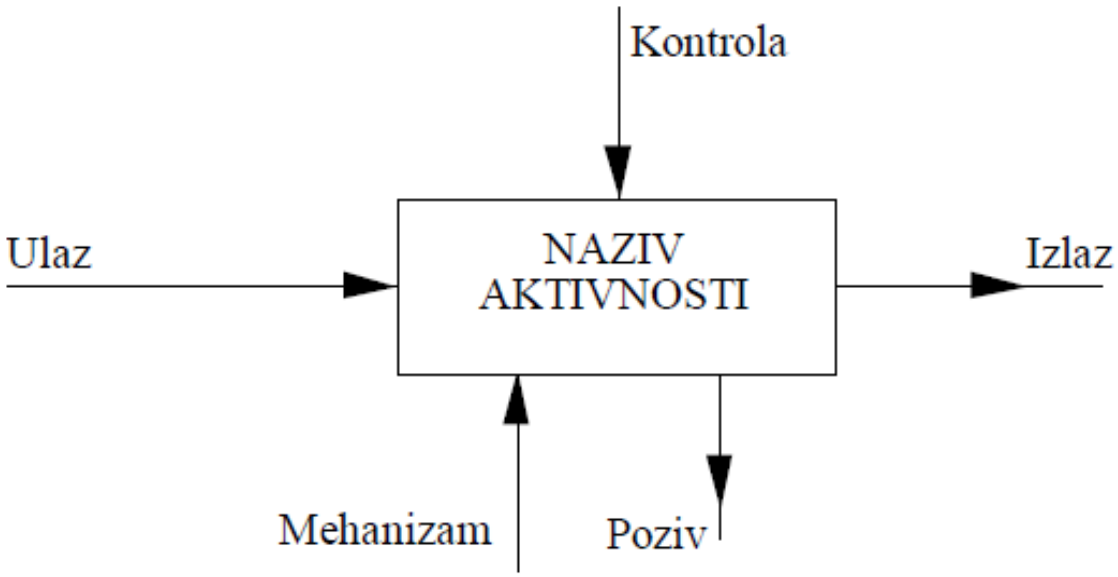
Слика 1 - Синтакса правоугаоника (Box) [1]

Сваки правоугаоник има назив и број у оквиру граница правоугаоника. За назив активности се користи активан глагол или глаголска фраза која описује функцију. Број се користи да би био препознат предмет описа правоугаоника у придруженом тексту. Стрелице (слика 2) не представљају ток или секвенцу као у традиционалном моделу дијаграма тока података, већ преносе податке или објекте везане за посматрану активност.



Слика 2 - Синтакса стрелице [1]

Однос између активности и стрелица је одређен помоћу стране правоугаоника (активности) на који је стрелица наслоњена (слика 3).



Слика 3 - Општи модел дијаграма контекста [1]

Улази се трансформишу у активности користећи механизме или ресурсе као што су особље и машине, да би произвеле излазе. Операције активности ће бити усмерене контролама као што су политике и процедуре. Позив је специфичан случај механизма и означава да позивајућа активност нема властити детаљнији дијаграм, већ даје детаљнији приказ изведен на неком другом моделу. [1]
Следи детаљнији опис токова података:

Улазна стрелица - представља материјал или информацију која се користи или трансформише са циљем дефинисања излаза. Дозвољава се могућност да одређене активности не морају имати улазне стрелице.

Контролне стрелице - стрелице које улазе у правоугаоник одозго дефинишу се као контроле (Control). Контролне стрелице регулишу, односно одговорне су за то како, када и да ли ће се активност извести, односно какви ће бити излази. Свака активност мора имати најмање једну контролну стрелицу.
Контроле су често у облику правила, политика, процедура или стандарда. Оне утичу на активност без могућности да буду трансформисане или употребљене.
Излазне стрелице - материјали или информације створене активношћу. Свака активност мора имати најмање једну излазну стрелицу. Активност која не ствара излаз, не треба ни моделирати.
Стрелице механизама - стрелице на доњој страни правоугаоника представљају механизме. Стрелице окренуте према горе идентификују значење које подржавају извршење активности. Стрелице механизама су они извори који изводе активности, а сами се не "троше". Механизми могу бити људи, машине и/или опрема тј. објекти која обезбеђују енергију потребну за извођење активности. Механизами могу бити изостављени из активности.
Стрелица позив - стрелице механизма које су окренуте на доле дефинишу се као стрелице позива (Call arrows). Стрелица позив је специфични случај стрелице механизма и она означава да позивајући правоугаоник нема свој властити детаљнији дијаграм, али је детаљнији приказ изведен на неком другом правоугаонику у истом или неком другом моделу. Више позивајућих правоугаоника могу позивати исти правоугаоник на неком другом или истом моделу. Именују се са бројем декомпозиционог дијаграма који садржи позвани правоугаоник заједно са бројем позивног правоугаоника.

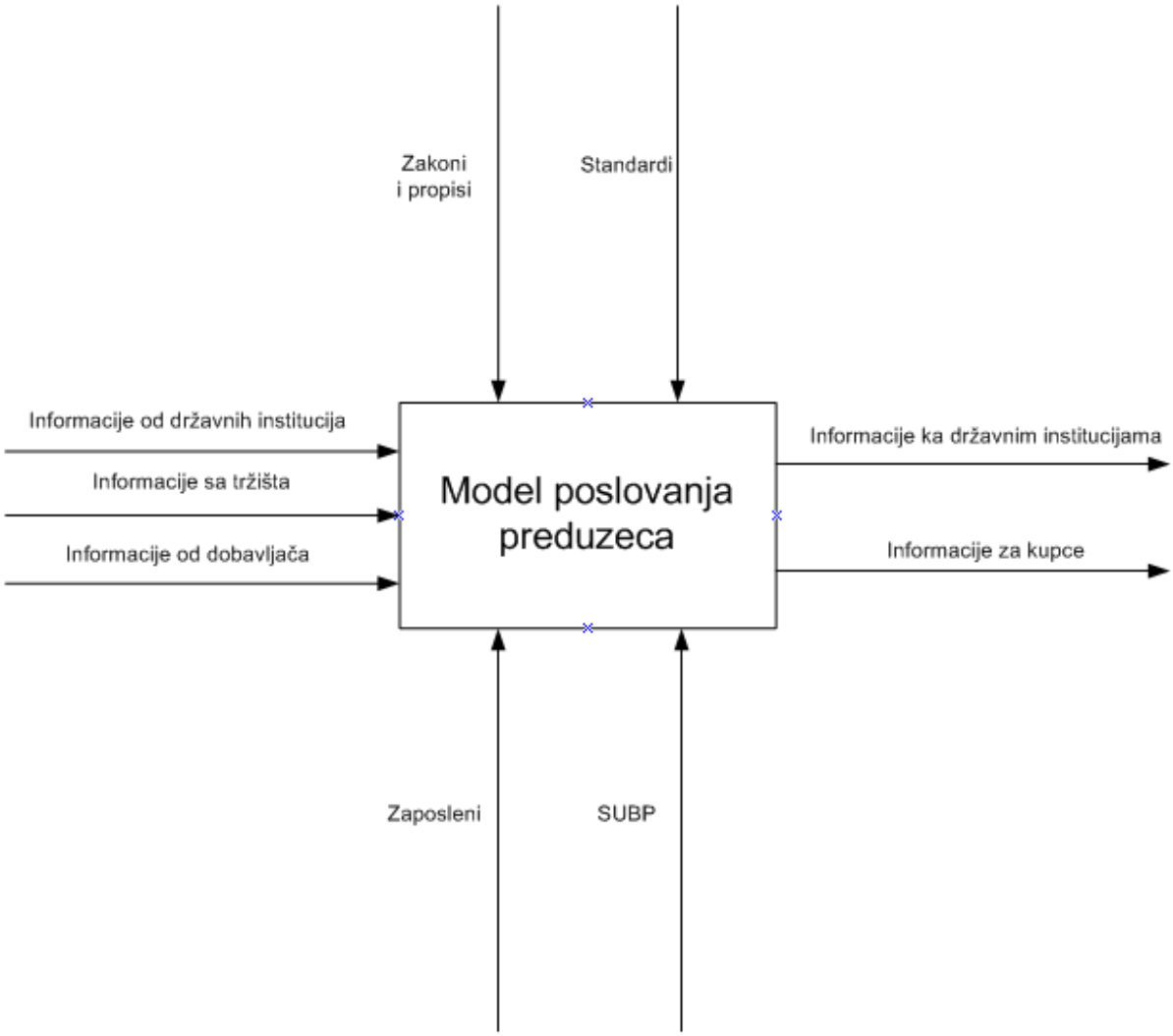
3. Контекстни дијаграм

Дефинисањем дијаграма контекста поставаља се граница између система који се посматра и његове околине. Дијаграм контекста је дефинисан једним правоугаоником (носи број 0) и представља границу модела који се проучава. У моделу се ток информација између система и околине, а такође и унутар самог система, одвија преко стрелица. Дијаграм контекста је највиши ниво апстракције посматраног система, који

се декомпозиционим дијаграмима преводи у нижи ниво апстракције. При дефинисању модела система треба почети од дефинисања излазних стрелица, па се померати према улазима, механизмима и контролама.[1]

3.1 Контекстни дијаграм за документ "Модел пословања предузећа"

Активност A0, која се појављује у контекстном дијаграму, описује оквире модела и мора бити одређена активном глаголском фразом, као, нпр. "Модел пословања предузећа" (слика 4).



Слика 4 - Контекстни дијаграм за активност "Модел пословања предузећа"

Са дијаграма контекста уочава се следеће:

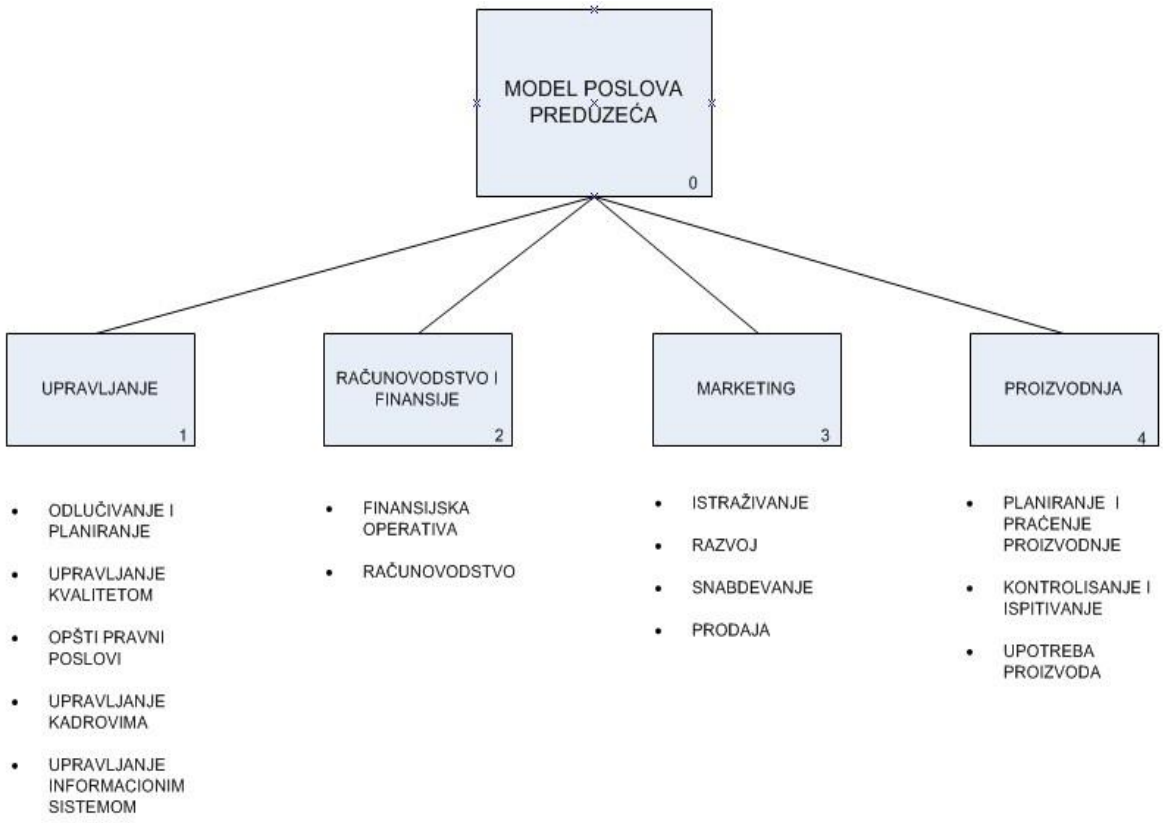
- Улазне информације су информације са тришта, информација од купца од кога се добија повратна информација о употреби производа и информације од пословних парнтера који утичу на снабдевање, а тиме и на ланац вредности организације.
- Излазне информације су информације за пословног партнера (информисаност о набавци и продаји) и информације за купца које треба да задовоље потребе купца за квалитетним производом.
- Контроле су закони и прописи (односе се на функцију финансија) и стандарди (везани за производњу и маркетинг).
- Механизми представљају систем за управљање базама података,запослени и др.

4. Дефинисање стабла активности

Стабло активности се дефинише применом методе top-down којом се полазна активност развија у хијерархију подређених активности. Структура овако развијених активности подсећа на стабло гдје корен стабла садржи полазну активност, док листови, тј. чворови који немају потомке, садрже простије активности које у потпуности решавају полазну сложену активност. Успостављају се вертикалне (хијерархијске) везе између активности, уз истовремено занемаривање веза (стрелица) између активности које се налазе на истом нивоу. Полазна активност (root) која се налази у корену стабла најчешће се означава са 0. Активност 0 се декомпонује на 1, 2, 3 итд. Активност 1 се декомпонује на 11, 12, 13 итд. Надређена активност се зове “родитељ” (parent), а подређене активности су “деца” (children).[1]
Разбијање активности “родитељ” на своју децу треба да има од две до шест подређених активности. Ако је више од шест подређених активности, то значи покушај да се смести превише детаља на један ниво.

4.1 Дефинисање стабла активности за модел пословања предузећа

На основу свега што је досад речено, треба дефинисати стабло активности за активност "Модел пословања предузећа" (слика 5).



Слика 5 - Стабло активности за контекстну активност "Модел пословања предузећа"

Приликом формирања овог стабла активности пошло се од опште чињенице да се овај модел може поделити на:

- Управљање
- Рачуноводство и финансије
- Маркетинг
- Производња

Активност 1 - Управљање може се поделити на подређене активности:

- Активност 1.1. Одлучивање и планирање (Где се дефинишу даљи стратешки планови како би се побољшало пословање фирме)

- Активност 1.2. Управљање квалитетом (Где се дефинише квалитет услуга које наше предузеће мора да оствари)

- Активност 1.3. Општи правни послови (Где се дефинишу и решавају правни послови, тј. решавају правни послови у складу са законом)

- Активност 1.4. Управљање кадровима (Дефинисање граница које кадровска служба обавља)

- Активност 1.5. Управљање информационим системима (Дефинисање информационих система који обезбеђују информације неопходне за сврхе доношења одлука и решавање проблема).

Активност 2 - Рачуноводство и финансије обухвата:

- Активност 2.1. Финансијска оператива (Где се регулишу и дефинишу сва средства плаћања)

- Активност 2.2. Рачуноводство (Дефинисање, планирање и контрола финансијских операција и трансакција)

Активност 3 - Маркетинг обухвата:

- Активност 3.1. Истраживање (Врше се истраживања која омогућују развој предузећа)

- Активност 3.2. Развој (Дефинишу се мере које ће омогућити развој фирме)

- Активност 3.3. Снабдевање (Дефинишу се планови за боље снабдевање производа у предузећу)

- Активност 3.4. Продаја (Дефинишу се рекламни производи)

Активност 4 - Производња обухвата:

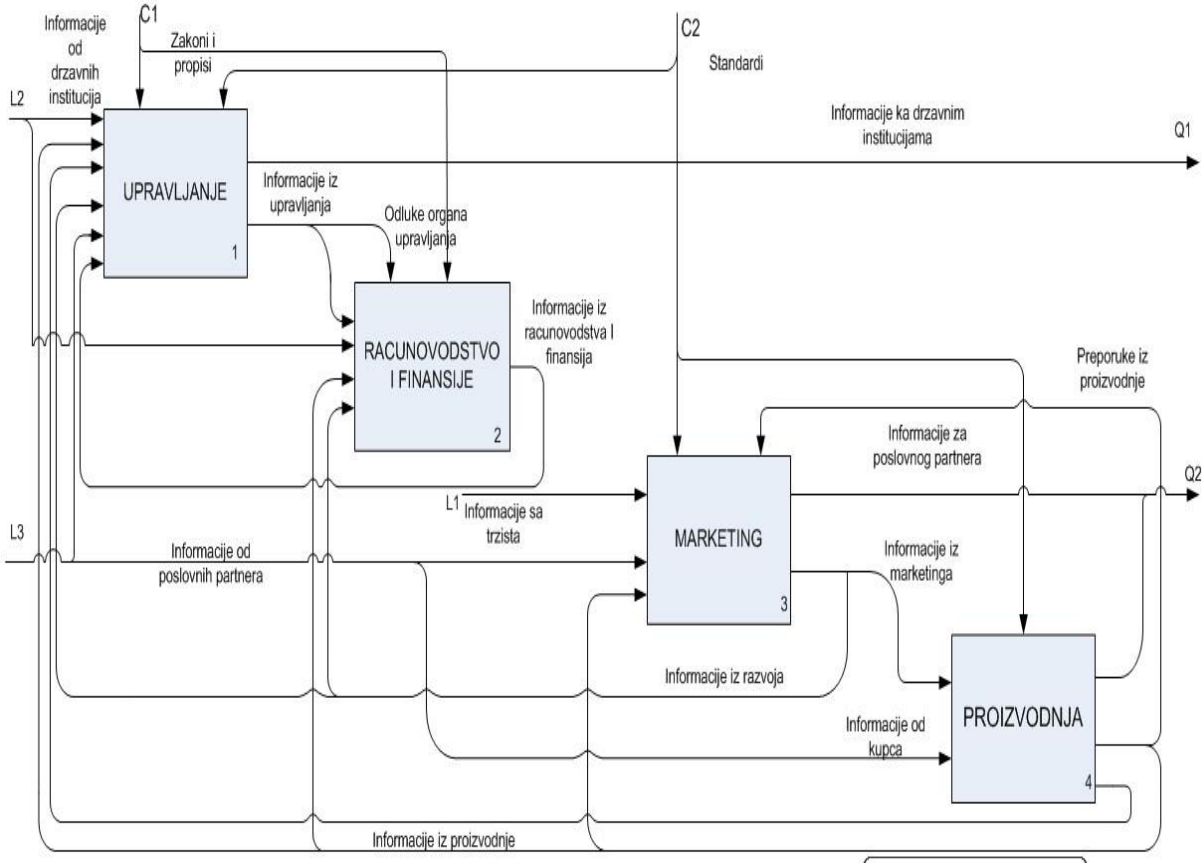
- Активност 4.1. Планирање и праћење производње (Прављење планова производње)

- Активност 4.2. Контролисање и испитивање (Испитивање и контролисање полупроизвода и готових производа)

- Активност 4.3. Употреба производа (Крајња провера производа пре слања купцу)

5. Дефинисање декомпозиционог дијаграма

На слици 6 се приказује декомпозициони дијаграм модела пословања предузећа и уочавају се четири глобална процеса: управљање, финансије, маркетинг и производња. За разлику од DTP-а где се процеси не могу директно повезивати, код IDEF0 стандарда то није случај. Наиме, стрелице које излазе из процеса могу директно улазити у друге процесе, с тим што треба обратити велику пажњу да уколико те излазне стрелице представљају улазе у други процес, треба да се и прикажу на страни улаза, а не на страни контрола или механизама.



Слика 6 - Декомпозициони дијаграм A0 за "Модел пословања предузећа"

За цртање декомпозиционог дијаграма коришћен је MS Visio.

5.1 MS Visio

Microsoft Visio (лого - слика 7) служи за израду пословних цртежа и дијаграма, као што су: организацијске шеме, дијаграми токова, мрежни дијаграми итд.
Развијен 1992.г у компанији Shapeware Corporation, док га 1999.г купује Microsoft.

Слика 7 - Microsoft Visio лого [7]

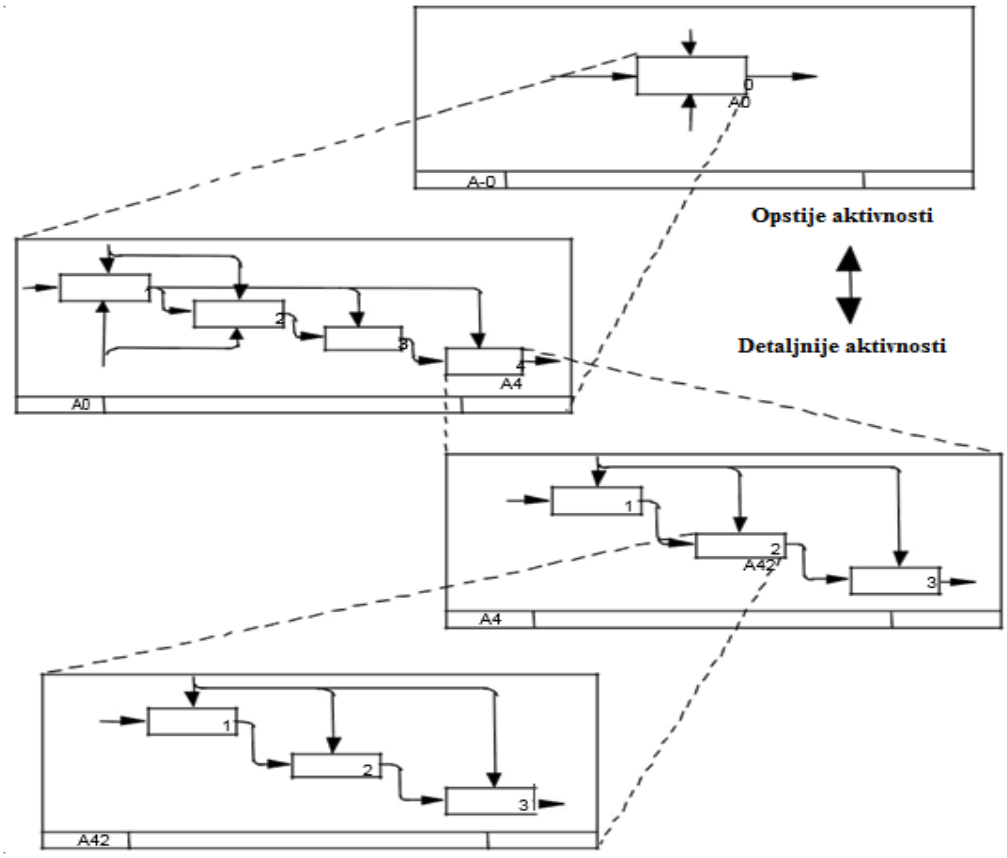
У зависности од потреба корисника, постоје 3 различите верзије овог програма:

- Стандардна,
- Професионална,
- Премијум

MS Visio садржи графичке могућности, интуитивно корисничко окружење, и ради по drag & drop принципу.

5.2 Правоугаоници у декомпозиционом дијаграму

Израдом декомпозиционог дијаграма успостављају се хоризонталне везе између активности истог нивоа. Активности су смештене у правоугаоницима који се цртају у дијагоналном смеру, од горњег левог угла стране ка доњем десном углу. Свакој активности мора се доделити назив у облику глаголске фразе, те свака активност мора имати најмање једну контролну и једну излазну стрелицу.[1]
На слици 8 приказана је структура формирања декомпозиционог дијаграма. Полази се од контекстног дијаграма (описан у претходном поглављу) који се дефинише на највишем нивоу, па се изводи декомпоновање у подређене (child) дијаграме. Свака од подактивности подређеног дијаграма може креирати свој дијаграм на нижем нивоу. На тај начин се дефинишу различити нивои апстракције, тј. на вишим нивоима су општије активности и груписане стрелице, које се на нижим нивоима декомпонују и детаљније описују.



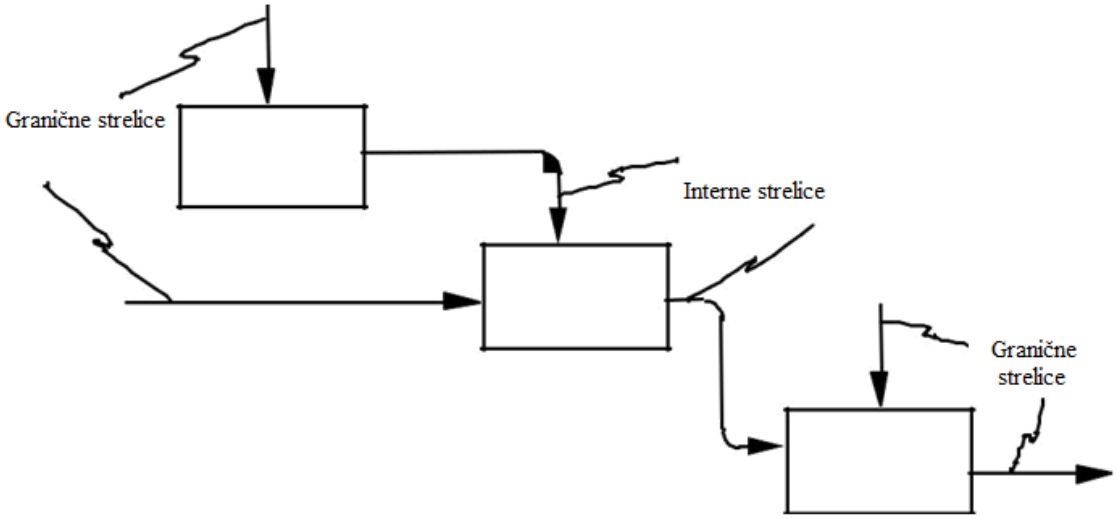
Слика 8 - Декомпозициона структура IDEF0 методологије [1]

5.3 Стрелице у декомпозиционом дијаграму

Стрелице у декомпозиционом дијаграму омогућују хоризонтално повезивање дефинисаних активности.

Као што се може видети, на претоходној слици стрелице дефинисане на контекстном дијаграму се преносе у подређени декомпозициони дијаграм. Дакле, стрелице дефинисане у активности која претходи ("родитељ") појављују се у подређеном декомпозиционом дијаграму као граничне стрелице (boundary arrows), тј. као стрелице које настају ван оквира посматраног дијаграма (слика 9).

У оквиру декомпозиционог дијаграма дефинишу се тзв. експлицитне или интерне стрелице које повезују активности (слика 9). Декомпозициони дијаграм без унутрашњих стрелица указује на организациони приступ декомпозицији, а не функционални.[1]



Слика 9 - Стрелице у декомпозиционом дијаграму [1]

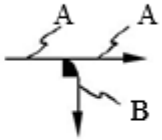
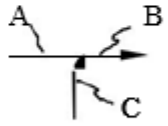
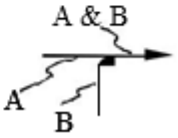
Улазне граничне стрелице које долазе из надређеног дијаграма у подређени дијаграм могу се делити у више специфичних стрелица и обрнуто, док излазне граничне стрелице из подређеног декомпозиционог дијаграма се групишу и излазе у надређени дијаграм.[1]

Стрелице које се користе у оквиру декомпозиционог дијаграма могу се поделити на:

- Стрелице које се гранају или удружују,
- Повратне стрелице и
- Скривене стрелице.

5.4 Стрелице које се гранају или удружују

Стрелице представљају својеврсну мрежу која се може гранати и удруживати. На следећој слици дат је приказ стрелица, њихово грањање и удруживање.[1]

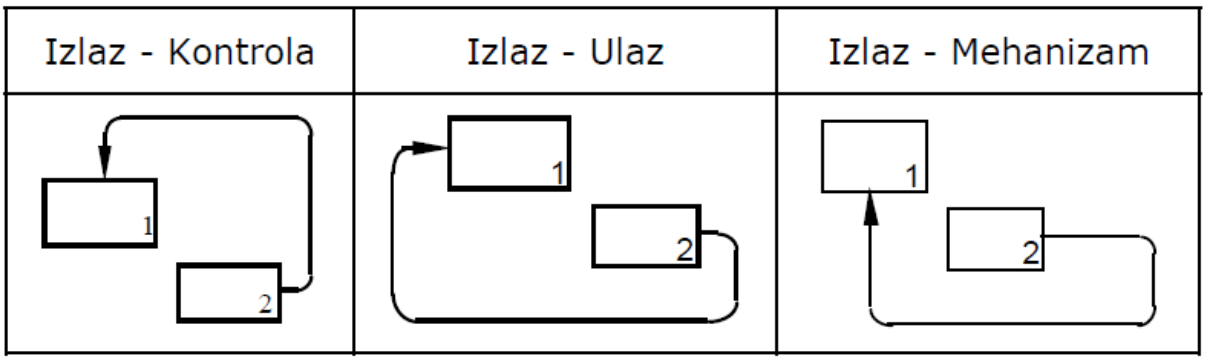
Puni skupovi informacija u svim granama.	
Podela na pun skup u prvoj i podskup informacija u drugoj grani	
Podela na različite podskupove informacija	
Integracija strelica	

Слика 10 - Структура рачвања и удруживања стрелица [1]

У првом случају, приказаном на слици 10, долази до рачвања улаза А у исте скупове информација, док се у другом случају одваја нови подскуп В. У трећем случају се А грана у В и С, а у четвртом се А и В интегришу у А&В. Препоручује се да се дефинише назив сваке стрелице приликом израде декомпозиционог дијаграма да не би дошло до двосмислености.

5.5 Повратне стрелице

Повратна петља је појава када излаз из активности представља улаз, контролу или механизам неке раније активности у декомпозиционом дијаграму. Овакви излази су, обично, неке негативне карактеристике.[1]



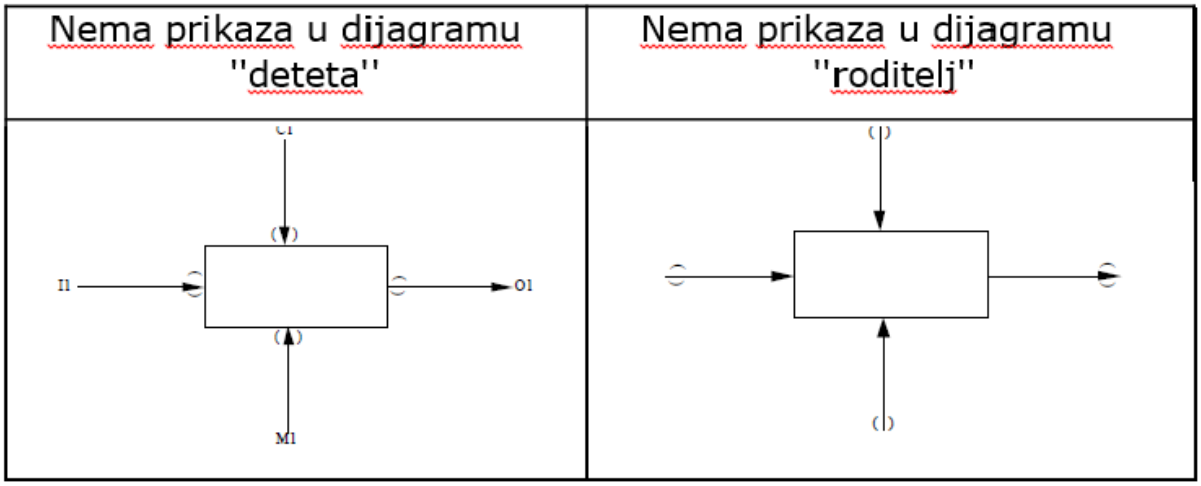
Слика 11 - Повратне стрелице [1]

У првом и другом случају приказаном на слици 11. повратна петља и улазна повратна петља као контрола представљају итерацију, или рекурзију, док у трећем случају

повратна петља Излаз - Механизам има улогу подршке на нивоу ресурса за извођење одређене активности.

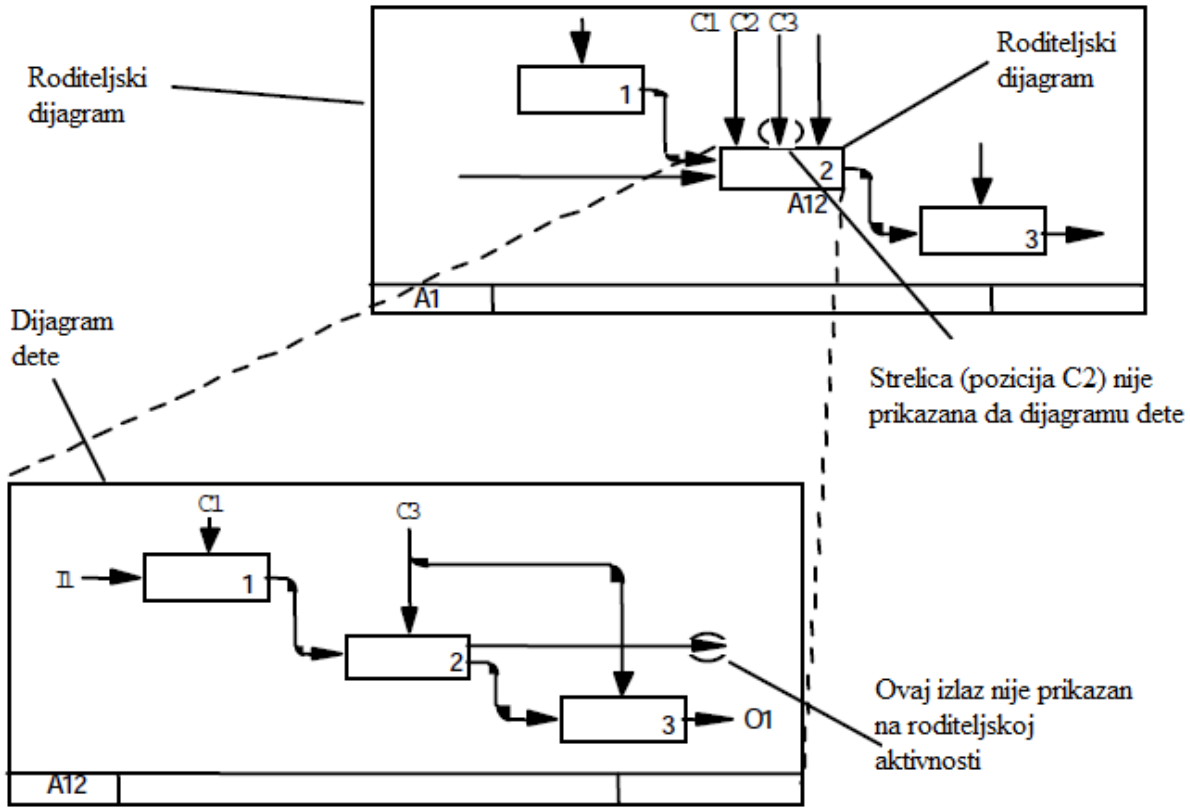
5.6 Скривене стрелице

Скривене стрелице су посебан тип стрелица које настају ако се жели да стрелице не буду виђене на надређеном или подређеном декомпозиционом дијаграму. То је ситуација када је дијаграм пренатрпан, па се због јасноће избегава његово приказивање. Друга ситуација је ако се жели приказ будућих догађаја.
Може бити и случај да стрелица представља контролу која се примењује на свакој активности у декомпозицији, а да је била пропуштена ради јасноће у нижим нивоима. Ознака да је стрелица сакривена су заграде (слика 12).[1]



Слика 12 - Врсте скривених стрелица [1]

Да би се ово појаснило приказаћемо још један пример скривених стрелица на следећој слици. На слици 13 се види да уколико је скривена стрелица сакривена на родитељском дијаграму онда се она неће видети на дијаграму детета. Док уколико је скривена стрелица нацртана на дијаграму детета, на родитељском дијаграму се неће приказивати.



Слика 13 - Пример скривених стрелица [1]

6. Дијаграми тока података

Структурна системска анализа користи тзв. дијаграме тока података (Data Flow Diagrams) који графички приказују проток и трансформацију података у систему. Дијаграми тока података сликовито приказују пословне процесе у систему и информационе везе међу њима. На њима се могу уочити одређени типови података који су произашли из једне активности, а касније употребљени у другој. Те информације представљају излаз из једног процеса, а улаз у неки други процес. DTP су презентације протока података у-из-између појединих поступака тј. процедура, затим подсистема или система. DTP приказују проток стварних података између постојећих активности и логичких база података, тако да нас обавештавају о начину на који су те активности међусобно повезане.[2]

Коришћењем универзалних симбола, формирају се дијаграми тока података, који омогућују представљање било ког система из било које области људског деловања. Такође, могуће је приказивање система на различитим нивоима уопштености. Систем се може презентовати глобално или детаљније, може се узети у обзир само део система и томе слично.

На основу дијаграма тока података се не може сазнати ништа о временском трајању догађаја, али се могу уочити извори података, њихове дестинације тј. одредишта, токови, базе података као и трансформације.

DTP су најприкладнији за приказивање система оријентисаних на проток података и врло добро истичу логичку основу система.

6.1 Симболи дијаграма тока података

У дијаграмима тока података користе се само четири примитивна концепта која су представљена одговарајућим симболима:[1]

Процес - представља извршење неког задатка који се односи на податке. Та активност улазне податке трансформише у излазне податке. Графички се процеси представљају кругом.

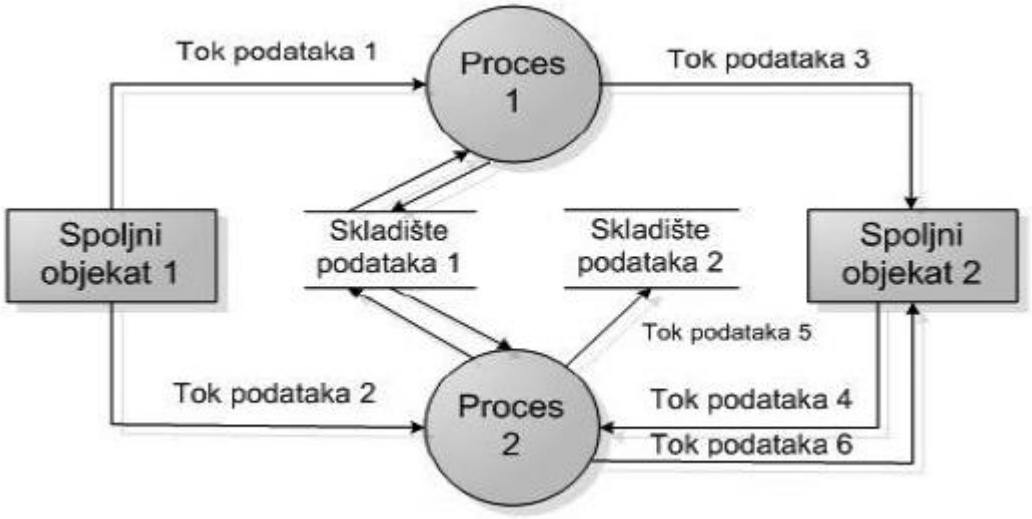
База података - или складиште података представља место задржавања, односно чувања података између процеса. Ту се податак смешта након изласка из процеса и након тога може бити употребљен од стране тог истог или неког другог процеса. Представља се путем правоугаоника без страна, са уписаном ознаком и називом базе.

Спољни објекат - треба да изврши повезивање са објектима ван контекста посматраног система и представља поноре и/или изворе токова података. Иначе, представља се правоугаоником, у који се уписује име објекта.

Ток података - описује кретање којим протичу групе података (документи, формулари, обрасци) који показује између којих елемената се одвија ток података. Ток података се представља путем стрелице, на којој се наводи назив структуре података.
Комбиновањем ових примитивних концепата описују се токови података у реалном систему, њихова трансформација и архивирање. Дијаграми тока података се, по правилу, допуњавају објашњењима и попуњеним примерцима докумената. Циљ давања објашњења је да се дефинише распоред и услови извршавања активности. Услове извршавања активности је потребно посебно дефинисати, јер у дијаграму тока података, процеси се представљају као да се сви безусловно извршавају. Опис редоследа и услова извршавања активности се даје путем слободног текста или неких других дијаграма којима може да се изрази динамика система.

6.2 Синтакса (правила креирања) дијаграма тока података

На следећој слици приказана је синтакса самог DTP-а (слика 14).



Слика 14 - Основне компоненте DTP-а [2]

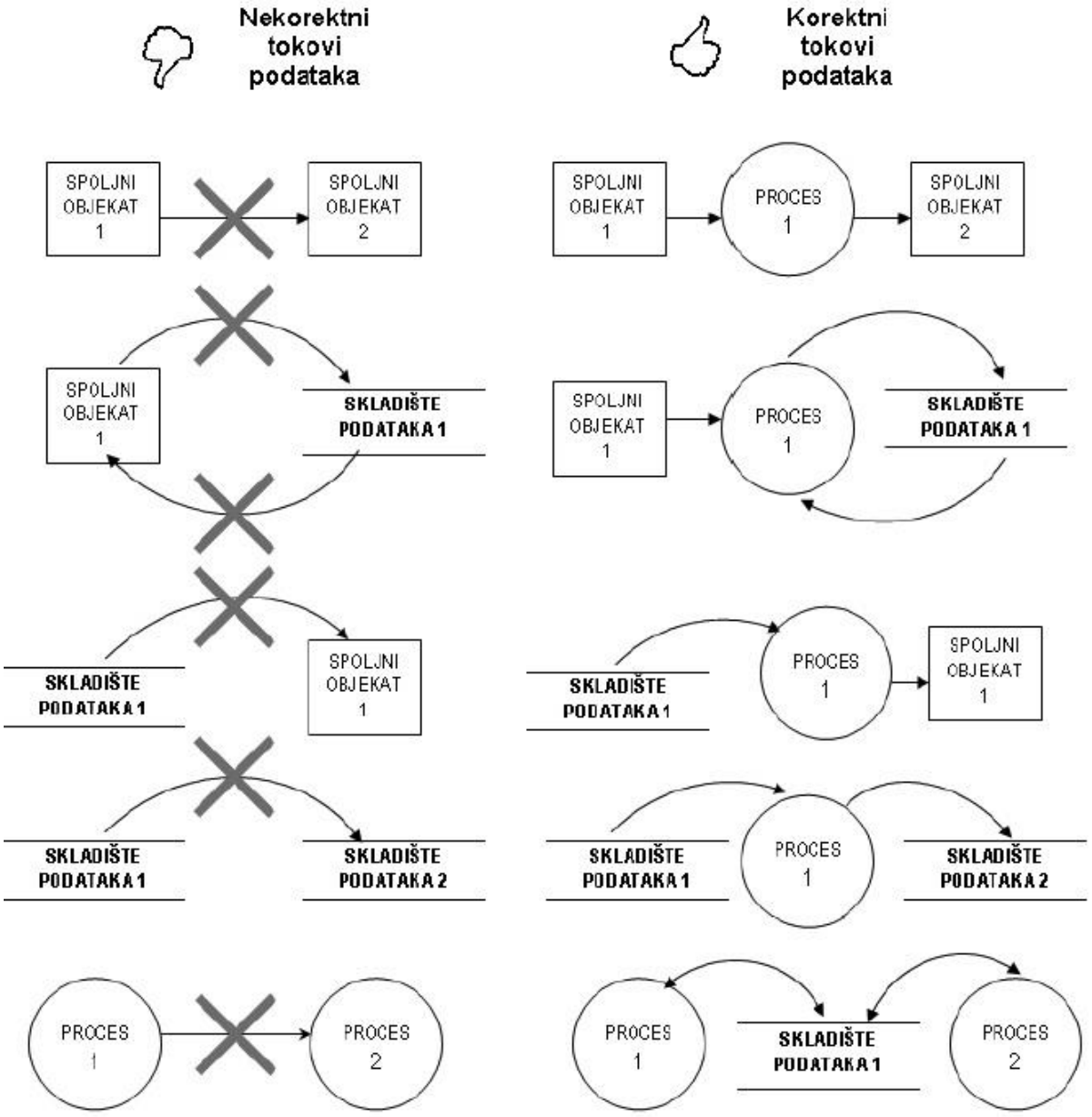
Код креирања дијаграма тока података важе следећа основна правила:

1. Ток података представља податак у покрету. Ток података представља улаз података у процес или излаз података или информација из процеса (то могу бити различита документа, формулари, текстови, књига и слично). Они се такође користе за приказивање креирања, брисања, или мењања података у бази података или датотеци. Ток података остварује везу између осталих компоненти система и на DTP-у се представља именованом, орјентисаном линијом.

2. Сваки ток података мора да има свој извор и понор. Међутим, за један ток, било извор, било понор (или оба) мора да постоји процес. На слици 15 се приказују непрописне ситуације токова података за које се могу дати следећи коментари:

- Директно повезивање спољних објеката, објеката ван система, неким током података није дозвољена, јер представља опис односа објеката ван посматраног система, па није од интереса. Ако би таква веза била од интереса, односно остваривала се преко посматраног система, тада би се морао дефинисати процес у посматраном систему који такву везу успоставља.
- У реалном систему спољни објекти немају директан приступ складиштима података, већ то обављају процеси система.
- Уколико је неопходно да се подаци са једног складишта података пренесу на друго складиште података, та веза се не може остварити директном везом између складишта података, већ се мора формирати процес преко кога ће се вршити то премештање.

- Процеси се не могу директно повезивати, већ уколико на основу резултата које даје један процес, а чији садржај се смешта у складиште података, почиње да ради други процес, онда он узима те резултате из складишта у који су смештени и на тај начин се веза између процеса остварује путем складишта података.



Слика 15 Правилан и неправилан начин приказивања DTP компонената [2]

3. Сваки ток података на DTP-у мора имати име, које треба да одражава значење података које он носи. Ова имена треба да буду природна, а не нека специфична, кодирана, претерано скраћена имена, како би се остварила читљивост и разумљивост DTP-а.

Изузетак су токови који иду ка, односно од складишта података који не морају бити именовани. Уколико ток између процеса и складишта није именован, подразумева се да ток носи целокупан садржај и структуру података тог складишта.

4. Ток података се може гранати (слика 16). Истоимени токови на DTP-у у суштини представљају гранање једног тока, па морају имати заједнички извор, а могу имати различите поноре.

Слика 16 - Грањање тока података [2]

5. Процес обраде података је активна компонента система која врши трансформацију структуре и садржаја улазног у излазни или излазних токова у улазне токове података. Сваки процес има назив и ознаку. Назив процеса треба прецизно да означава функцију коју он обавља.

6. Сваки процес мора да има барем један улазни и барем један излазни ток података. Процес који има улазе, а нема излазе је тзв. црна рупа, јер изгледа као да је податак једноставно нестао. Процес који има излазе, а нема улазе, је такође немогућ.

7. Свако складиште података би требало да има барем један улазни и барем један излазни ток. Међутим, дозвољава се да складиште нема улазни ток, подразумевајући да се формира и ажурира у неком другом систему, односно да нема излазни ток, подразумевајући да посматрани систем формира и ажурира складиште које се користи у неком другом систему.

8. Сваки спољни објекат (спољни агент) мора да има барем један, било улазни, било излазни ток података, иначе би био изолован од осталог дела система.

9. Да би се избегло непотребно пресецање линија, дозвољава се да се складиште или спољни објекат, на једној слици, вишеструко понови.

7. Моделирање података

Моделирање података је техника за организовање и документовање структуре података система. Моделирање података се понекад зове моделирање базе података јер се модел података на крају имплементира као база података. Моделирање података је наше апстрактно виђење стања реалног система.
Модел података је поједностављено представљање реалног система преко скупа објеката (ентитета), веза између објеката и њихових атрибута. Модел података (у литератури дефинисан као модел објекти-везе (МОВ), или Entity Relationship - ER модел), преко скупа података и њихових међусобних веза, представља стање система у једном тренутку времена и садржи скуп информација о прошлости и садашњости система која је потребна да се под дејством будућих познатих улаза могу одредити његови будући излази.
Постоји неколико нотација модела објекти-везе. Многи су именовани према њиховим творцима (нпр. Chen, Martin, Bachman, Merise) или према објављеним стандардима (нпр. IDEF1X). Сви „језици“ моделирања података подржавају исте фундаменталне концепте и конструкције.

7.1 Основни појмови модела података

Сви системи садрже податке. Подаци описују неке „ствари“. Посматрајмо један школски систем. Школски систем садржи податке који описују следеће ствари Ученик, Учитељи, Часови, Учионице итд. Конкретни подаци који описују ученика су име, адреса, телефон, датум рођења, пол, смер, успех итд. Концепт који се користи за апстрактно представљање ствари са сличним подацима је објекат.
Објекат је класа особа, места, објеката, догађаја или концепата о којима треба да прикупљамо и складиштимо податке. Објекат је нешто што се може видети, додирнути или другачије осетити, који има своја својства и понашања и о коме корисници могу да складиште податке. Посматрајте окружење у коме се тренутно налазите. Погледајте около. Који објекти су присутни унутар вашег окружења? Можда видите врата, прозоре, књиге, столице или столове.
Све су то објекти који се могу видети и додирнути, па чак и ви представљате један видан објекат. Пошто смо дефинисали објекат као нешто што је могуће видети, додирнути или другачије осетити, поставља се питање који су то објекти који се могу на неки други начин осетити. На пример, уколико очекујете телефонски позив, то је нешто што можете да осетите. Уколико сте на пословном састанку и пословни састанак је такође нешто што се може идентификовати и на коме се може учествовати. То значи да су и телефонски позив и пословни састанак такође објекти.
Типови објеката се могу класификовати у особе, места, ствари или догађаје. У оквиру типа објекта особе могу се сврстати радници, клијенти, продавци, студенти и др. Одељење, агенција, складишта, зграде, кампус су примери типа објеката места. Примери типа објеката ствари укључују књигу, производ, возило, машину, сировине, софтверски пакет, опрему, DVD и др.

На крају објекти догађаја укључују поруџбину, плаћање, рачун, апликацију, регистрацију или резервацију и сл. Графички приказ објекта по Chen-овој нотацији је у облику правоугаоника.

Објекти имају свој идентитет који је дефинисан својствима или атрибутима. Атрибути су подаци који представљају карактеристике објекта. На пример, претпоставимо да смо заинтересовани за следеће атрибуте објекта клијент: шифра клијента, назив клијента, адреса, тип клијента, телефон, статус на рачуну. Не приказују се сви атрибути који описују дати објекат, већ само они за које је корисник заинтересован да држи у бази података. Неки атрибути се могу логички груписати у сложени атрибут. Сложени атрибут се састоји од више примитивних атрибута. На пример, адреса је сложени атрибут јер се састоји од следећих примитивних атрибута: улица, број, град, поштански број, држава.

За сваки атрибут се углавном дефинишу: тип податка, домен и дифолтна вредност. Тип податка дефинише која класа података може бити складиштена у тај атрибут. Примери типова података су дати на слици 17. Домен дефинише које вредности може да има један атрибут. На пример, испитна оцена студента узима вредности из домена 5 и 10. Дифолтна вредност је она вредност која ће бити ускладиштена за дати атрибут уколико је корисник не промени.

Људи радо класификују ствари, не би ли нашли сличности међу њима и према томе их груписали. Ствари са сличним својствима се групишу заједно. На пример, Петар Петровић и Марија Марјановић имају своја имена, адресе и занимања и пошто обоје раде у истом предузећу, можемо их сврстати у објекат запослених радника. Конкретан објекат се назива примерком или инстанцом (инстанце) објекта. На пример, Петар Петровић је инстанца објекта Запослени.

Logički tip podatka	**Logičko značenje**
NUMBER	Bilo koji decimalni ili celobrojni broj.
TEXT	String karaktera uključujući i brojeve sa kojima se ne može obavljati aritmetička operacija ili operacija poređenja.
MEMO	Kao i TEXT ali sa neodređenom veličinom.
DATE	Bilo koji oblik datuma.
TIME	Bilo koji oblik vremena, odnosno časova.
YES/NO	Atribut koji može da ima samo ove dve vrednosti.
VALUE SET	Skraćeni skup vrednosti. (Na primer, JR=junior, SR=senior).
OLE OBJECT	Na primer slika.
HYPERLINK	URL adresa (npr., *e-mail*)

Слика 17 - Логички типови података за атрибуте [2]

Један објекат има више инстанци. На пример, објекат Студент ће имати и неколико хиљада унешених студената слика 18. Свака инстанца објекта мора да има јединствени кључ по коме ће се претраживати у бази података. На пример, с обзиром да атрибут Број индекса задовољава особине кључа, јер јединствено идентификује сваку инстанцу објекта Студент (сваки студент има јединствени број индекса и не може га делити са другим студентом), тада ће Број индекса представљати кључ објекта Студент.

Objekat: STUDENT

Broj indeksa	Ime	Prezime	Adresa	PttBr	Broj telefona
122/03	Ana	Anić	Nušićeva 2	11000	011/123456
130/04	Pera	Perić	Nemanjina 3	21000	011/234567
140/02	Lea	Leić	Bul. umetn. 9	18000	018/456901

← Atributi

Instance

Слика 18 - Приказ атрибута и инстанци за објекат Студент [2]

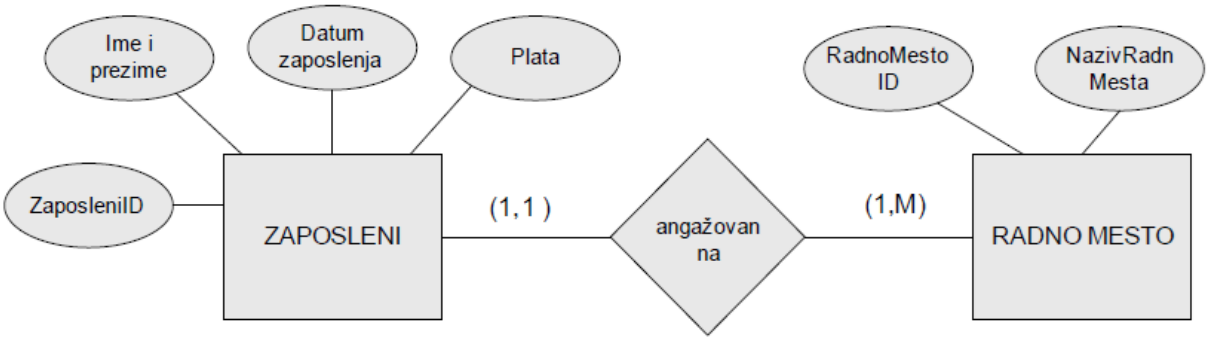
Некада је неопходно да више од једног атрибута јединствено идентификује објекат. Та група атрибута која јединствено идентификују објекат се зове сложени кључ. На пример, уколико би пројектовали информациони систем неког видео клуба, који има више копија касета, сложени кључ би био шифра наслова и број копије видео касете. Оба ова атрибута јединствено идентификују конкретну видео касету.
Да би неки атрибут могао да буде примарни кључ, он мора да задовољи следеће услове:

- једнозначност - не смеју да постоје две појаве за истим вредностима свих кључних атрибута;
- минималност - да не постоји подскуп атрибута кључа који је такође једнозначан (нпр., сложени примарни кључ објекта Особе не може бити комбинација атрибута ЈМБГ и Презиме, јер Презиме нема особину једнозначности);
- одређеност - постојање вредности у тренутку стварања инстанце;
- стабилност - отпорност на промене током времена;
- расположивост - доступност свим корисницима;
- неутралност - према значењу вредности кључа (самоговореће шифре).

Објекат може имати више атрибута који задовољавају карактеристике примарног кључа. На пример, објекат Радник се може јединствено идентификовати преко матичног личног броја или преко шифре запосленог или преко е-маил адресе. Сваки од ових атрибута се називају кандидати за кључ. Кандидати за кључ су кандидати за примарни кључ. Примарни кључ (Primary key) је кандидат за кључ који ће се најчешће користити да јединствено идентификује дати објекат. Примарни кључ се дефинише као један или више атрибута чија вредност јединствено идентификује једну инстанцу објекта. Примарни кључ мора да задовољи особине јединствености и нередундантности.

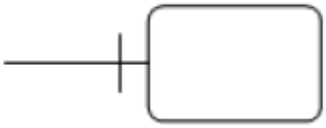
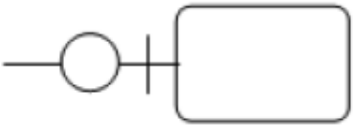
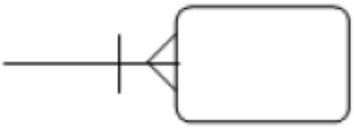
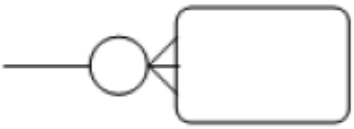
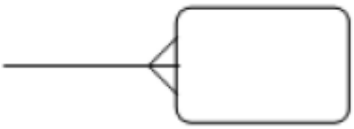
Дифолтна вредност примарног кључа је Not Null, односно кључ не сме да буде празно поље, јер онда неће моћи да јединствено идентификује дату инстанцу објекта. Сви други кандидати за кључ који нису изабрани за примарни кључ се зову алтернативни кључеви.

Објекти не егзистирају сами већ морају бити у некој релацији или вези са другим објектима. Уколико посматрамо објекте Запослени и РадноМесто, једна од релација између ова два објекта ће бити Ангажован, што значи да је запослени ангажован на одређено радно место у некој организацији. На слици 19 је приказана релација ова два објекта са својим атрибутима, према Chen-овој нотацији.



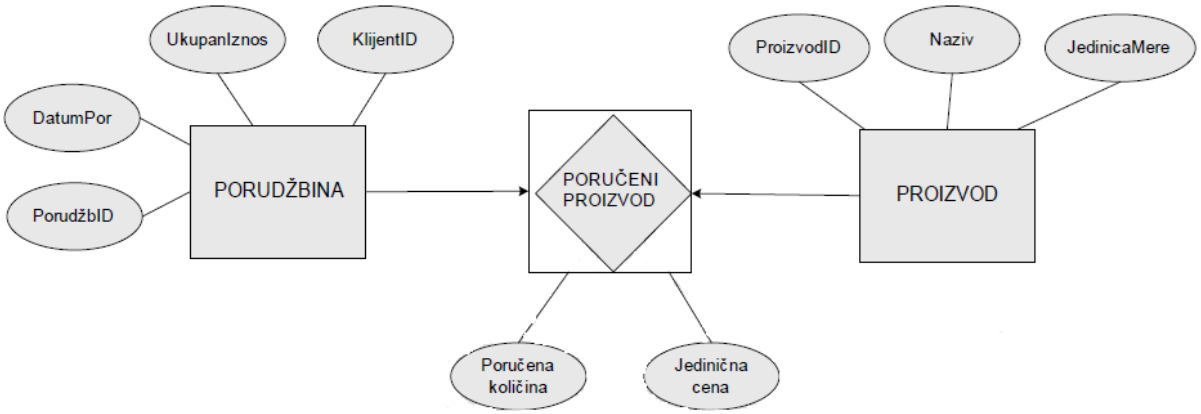
Слика 19 - Релација објеката Запослени и РадноМесто са својим атрибутима [2]

Једна од битних карактеристика веза између објеката је кардиналност (тип везе). Кардиналност дефинише минимални и максимални број догађаја једног објекта који се налази у конкретној релацији са другим објектом.
Пошто су све релације двосмерне, кардиналност се мора дефинисати за оба смера. Као што је приказано на слици 19, у заградама, конкретан запослени (нпр., Петар Петровић) мора да буде ангажован на само једном радном месту. На пример, Петар је ангажован на радно место програмера. Међутим, на одређено радно место може бити минимум један (1), а максимално више запослених (M). На пример, на радно место консултанта може бити ангажовано више запослених, односно у организацији је запослено више консултаната. Графичка нотација кардиналности зависи од нотација модела објекти везе. На слици 20, је приказана нотација кардиналности према Мартиновој нотацији ЕР модела.

Interpretacija kardinalnosti	Minimum	Maksimum	Grafička notacija
Tačno jedan	1	1	
Nula ili jedan	0	1	
Jedan ili više	1	>1	
Nula, jedan ili više	0	>1	
Više od jedan	>1	>1	

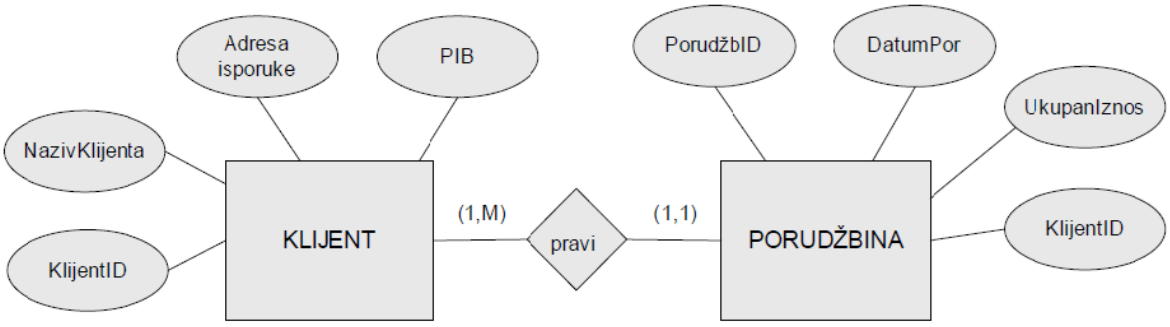
Слика 20 - Мартин-ова нотација кардиналности [2]

Агрегација истовремено представља и објекат и везу, односно удружени или асоцијативни објекат (associative entity), између два или више објекта слика 21. Уколико постоји потреба да се у току релације чувају додатни атрибути, тада релација постаје удружени објекат. Удружени објекат је објекат који наслеђује примарне кључеве својих тзв. „родитеља“. На пример, атрибути удруженог објекта Поручени производ су Поручена количина и Јединична цена 21, а њени кључеви би били атрибути PorudzbinaID и ProizvodID.



Слика 21 - Агрегација "Поручени производ" [2]

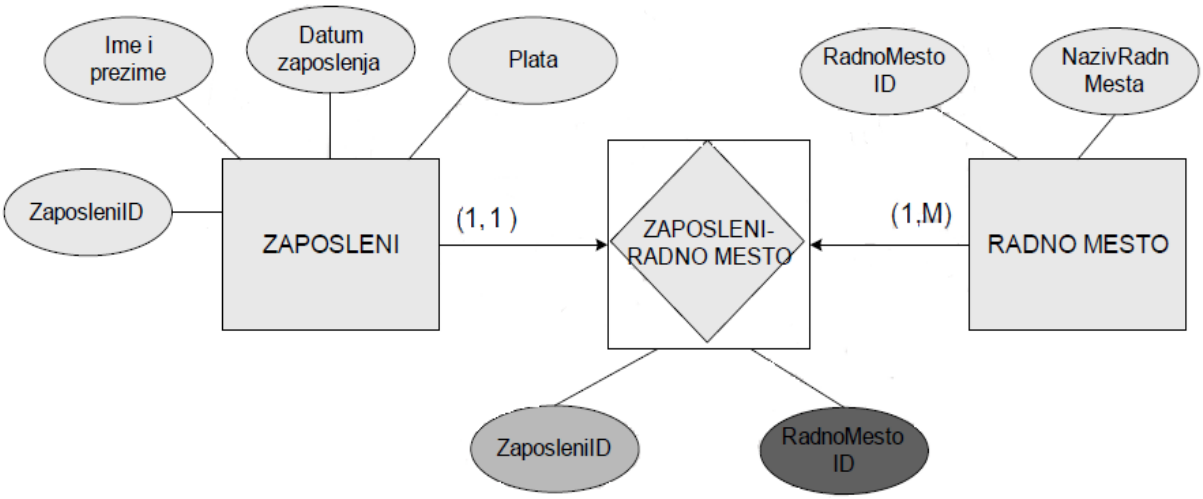
Објекти се повезују помоћу спољњег кључа. Спољни кључ (Foreign key) је атрибут или група атрибута једног објекта, чија се вредност користи за повезивање са вредношћу примарног кључа другог објекта. Односно, спољни кључ објекта 1 је примарни кључ објекта 2 са којим је објекат 1 у вези. Спољни кључеви и њима одговарајући примарни кључеви дефинисани су над истим доменом. На пример, погледајмо релацију Клијент и Поруџбина са слике 22. С обзиром да клијент може да има више поруџбина, тада ћемо примарни кључ објекта Клијент (у нашем примеру то је KlijentID) пренети у објекат Поруџбина, где ће тај атрибут представљати његов спољни кључ.
За конкретну поруџбину, постојањем спољног кључа KlijentID, знаће се ко је начинио конкретну поруџбину.



Слика 22 - Релација Клијент - Поруџбина [2]

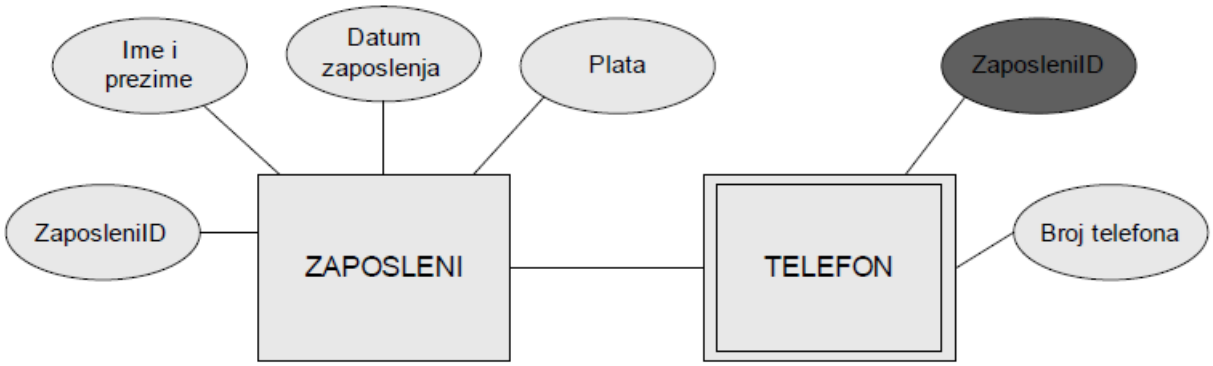
Да резимирамо, објекат Поруџбина ће имати примарни кључ PorudzbinaID, а спољни кључ је KlijentID који га повезује са објектом Клијент. Обично, где постоји кардиналност типа (1,1), онда тај објекат узима примарни кључ објекта са којим је у вези, а који ће представљати његов спољни кључ. Као што се види са слике 22, кардиналност код објекта Поруџбина је (1,1), тако да ће спољни кључ стајати на његовој страни. Код оних релација где постоји веза више према више, онда се убацује међу објекат, како би решио проблем вишезначне зависности. Уколико посматрамо претходни пример описан на слици 19, уочава се да постоји веза 1 према више, односно запослени може да буде ангажован на једном радном месту и на једном радном месту може да буде ангажовано више запослених.

У том случају се убацивањем удруженог објекта прави веза један према више (1,M) према овим објектима. Једини атрибути овог међу-објекта су примарни кључеви објеката које повезује (слика 23).



Слика 23 - Решавање проблема вишезначне зависности [2]

Проблем одређивања спољних кључева ће бити детаљније образложен кроз поступак нормализације. Уколико постоји потреба да се понављају вредности једног атрибута у оквиру одређеног објекта, тада се такав атрибут издваја и формира се слаб објекат. На пример, за објекат запослени који има атрибут телефон, желимо да памтимо више телефона. Тада ћемо направити слаб објекат телефон који ће наследити примарни кључ свог над објекта (Слика 24).



Слика 24 - Пример слабог ентитета [2]

Многе компаније прво обављају стратешко планирање где се развија стратешки план информационог система у коме се дефинишу визија и архитектура информационог система. Архитектура информационих система укључује модел података компаније. Модел података компаније идентификује основне објекте без описивања његових атрибута или кључева, а такође може да укључи, а не мора, релације између објеката.
Први задатак у моделирању података је да се одреде основни објекти система. Постоји неколико техника које се могу користити за идентификацију објеката:
Током интервјуа са власницима и корисницима система, треба обратити пажњу на кључне речи њихове дискусије. На пример, уколико корисник каже «Треба информисати наше сталне купце, о новом квалитету производа», приметићемо да су кључне речи у овој реченици Купци и Производ, а они су уједно и објекти система.
У току интервјуа треба питати власнике и кориснике система да идентификују оне ствари за које желе да прикупљају, складиште и добијају информације. Често су то објекти који треба да буду описани у моделу података.
Друга техника за идентификацију објекта је да се проуче постојећи формулари и картотеке. Неки формулари идентификују догађај објеката. На пример, поруџбина, уплата, депозит итд. Неки CASE алати такође могу да идентификују објекте. Када се идентификују објекти система, треба им дати једноставна, значајна, пословно-орјентисана имена. Други задатак у моделирању података је идентификовање атрибута објеката. Предлаже се следећи начин:
Многе компаније користе стандардна имена и скраћенице. Администратор података обично одржава такве стандарде. Пажљиво бирајте имена атрибута. Многи називи атрибута имају исту основу, нпр. име, адреса, датум, њих би требало раздвојити, као нпр., НазивДобављача, НазивКлијента и сл. Трећи корак је да се идентификују кључеви за сваки објекат. Предлаже се следеће:

- Вредност кључа не сме да се мења у току века трајања сваког објекта. На пример, Презиме особе се не може узети као кључ, јер особа може да промени своје презиме уколико се венча или разведе.
- Вредност кључа не може да буде празно поље.
- Контроле кључа морају да буду инсталиране, како би вредност кључа била увек валидна.

Четврти корак је повезивање објеката. Релација је веза између два или више објеката, тј. представља однос који постоји међу објектима, било у реалности или у мислима. Називи релација треба да буду глаголи, нпр, уписује, полаже, предаје итд. Објекат од кога је успостављена веза зове се “родитељ”(parent) или домен, а објекат ка коме је успостављена веза зове се “дете”(child) или кодомен. У току одређивања релација треба да се одреде и типови веза или кардиналности.
У току одређивања објеката система, њихових атрибута, кључева и релација исцртава се контекстуални (концептуални) модел података, накоме ће се јасно уочити објекти и њихове међусобне везе.
Логички модел података је потпуно одређен уколико су дефинисане следеће три компоненте:
Структура података - којима се дефинишу статичке карактеристике система (опис

објеката, атрибути и везе).
Ограничења - логичка ограничења на податке (правила интегритета) која се не могу дефинисати преко структуре модела података (структурна и вредносна ограничења) и односе се на дефинисање пословних правила.
Скуп оператора (операције) - дефинише динамичку интерпретацију података кроз њихову обраду (одржавање базе података и претраживање) има утицаја на дефинисање физичког нивоа модела и верификацију финалног дизајна.
Пети корак је нормализација релационог модела података. Нормализацијаје техника анализе података која организује атрибуте на начин како би се добили нередундантни, стабилни, флексибилни и прилагодљиви објекти. У току нормализације проверава се да ли су атрибути добро постављени и одређују се одговарајући спољни кључеви који повезују објекте.
Последњи корак је превођење логичког модела података у физички модел (шема базе података) у зависности од изабраног система за управљање базом података (Database Management System - DBMS).

7.2 IDEF1X стандард за моделирање података

Постоји много различитих верзија модела објекти везе, који се у већини случајева међусобно разликују само синтаксно. У пракси најчешће коришћена верзија је IDEF1X стандард америчке владе за моделирање података. IDEF1X је реализован кроз ERwin CASE алат. Код IDEF1X се користе појмови родитељ и дете. Објекат од кога је успостављена веза се зове родитељ, а објекат ка коме је успостављена веза зове се дете.
У IDEF1X дефинишу се две врсте објеката:

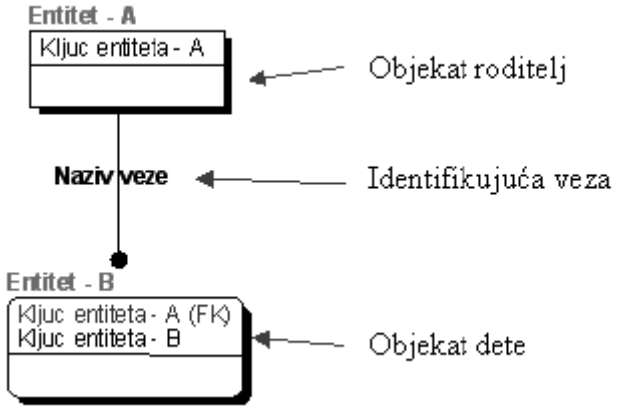
- независни објекти - објекти који могу постојати независно од других објеката у систему тзв. јаки објекти.
- зависни објекти - објекти чија егзистенција и идентификација зависе од другог или других објеката.

Постоје четири групе зависних објеката:

- слаб објекат - објекат који се понавља више пута за одређени независни објекат. На пример, објекти Фактура и СтавкаФактуре. За објекат СтавкаФактуре се каже да је слаб објекат, јер потпуно зависи од објекта Фактура.
- асоцијативни објекти - представљају везу више објеката;
- пројектни објекти - слични су асоцијативним, само што немају сопствене атрибуте;
- објекти категорије (генерализација /специјализација) - представљају под-категорију објекта.

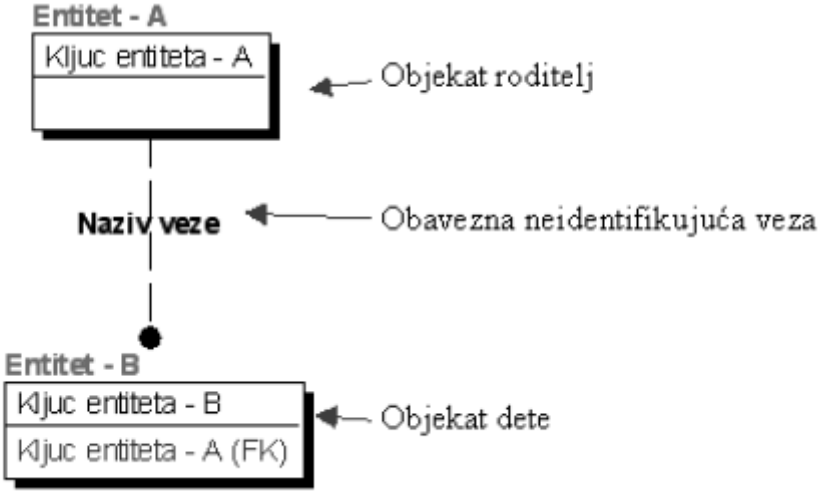
Код објекта типа категорије дефинишу се: надређени тип који има заједничке особине (нпр., Партнер) и подређени објекти (нпр., ФизичкоЛице и ПравноЛице), који се идентификују кључем надређеног и поседују своје специфичне особине.

Код IDEF1X стандарда разликују се следећи типови веза:
Идентификујуће везе које објекат дете идентификује кроз његову везу са објектом родитељ. Идентификујућа веза (слика 25) је приказана пуном линијом и повезује објекат родитеља са објектом дете са тачком на страни објекта дете. У идентификујућој вези објекат родитељ има свој независни примарни кључ (Кључ А), а објекат дете има сложени кључ који се састоји од свог кључа (Кључ В) и пренесеног родитељског кључа (Кључ А(FK)). Дакле, инстанце објекта родитељ се дефинишу независно, а инстанце објекта дете се не могу идентификовати без идентификатора објекта родитељ.



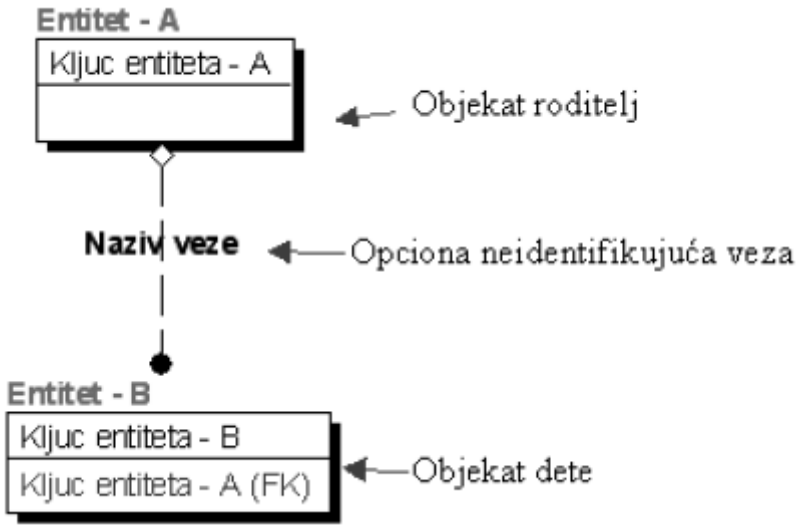
Слика 25 -Идентификујућа веза [1]

Неидентификујућа веза не идентификује дете преко идентификатора родитеља. Уколико се сваки примерак објекта дете може јединствено идентификовати без знања везе са примерком објекта родитељ, онда се таква веза дефинише као неидентификујућа веза. Неидентификујуће везе су приказане испрекиданом линијом која повезују објекат родитељ и објекат дете са тачком на страни објекта дете. Неидентификујућа или слаба веза зависи од начина дефинисања кључева од родитеља ка детету и то на два начина: обавезна и необавезна (опциона) неидентификујућа веза. Ако је веза обавезна (No Nulls, или Mandatory) из перспективе родитеља, онда је дете егзистенцијално зависно од родитеља (слика 26).



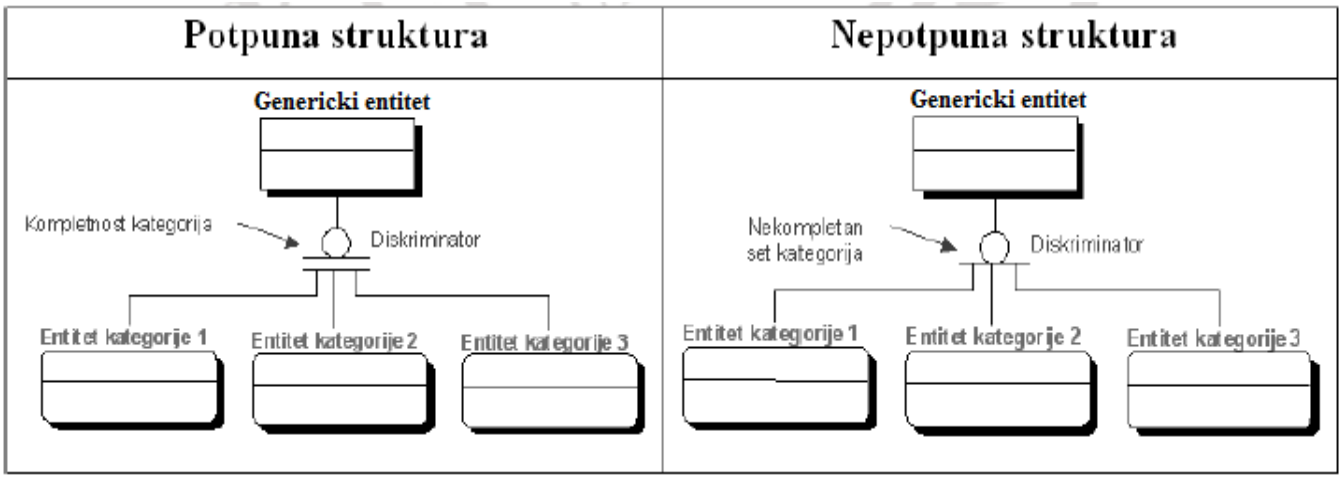
Слика 26 - неидентификујућа обавезна веза [1]

Ако је веза необавезна (Nulls Allowed или Optional) (слика 27), тада дете нити је егзистенцијално нити идентификационо зависно, али поштује ту везу.



Слика 27 - Неидентификујућа необавезна веза [1]

ERwin користи ромб (diamond) да назначи случај егзистенцијалне и идентификационе зависности. Ромб може постојати само у слабим везама (пошто је јака веза у оквиру примарног кључа, а примарни кључ не може да има Null вредност).
Веза категорије тј. везе према подтиповима је дефинисана за хијерархијску везу између надређеног генеричког објекта који садржи заједничке особине подређених објеката категорије (Слика 28). Овај тип везе се дели на комплетну категорију или тзв. потпуне структуре где су наведене све могуће категорије (подтипови) надређеног генеричког објекта и некомплетну категорију или тзв. непотпуну структуру, где су наведене само



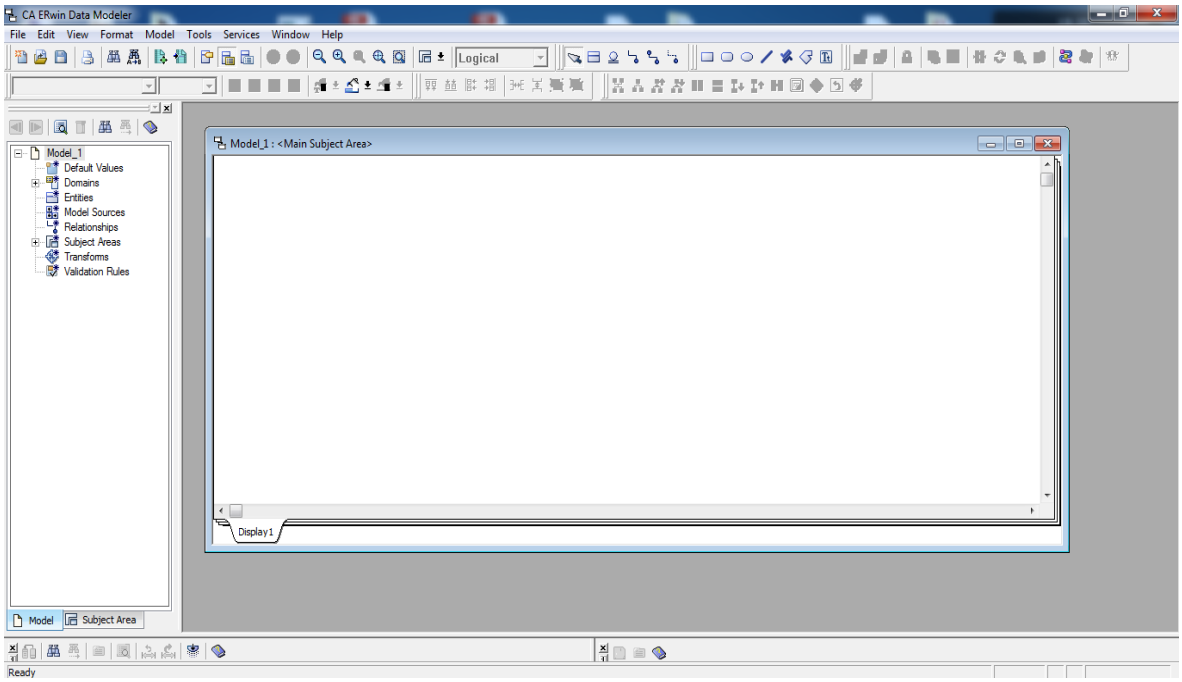
категорије (подтипови) надређеног генеричког објекта.

Слика 28 - Врсте ентитета категорије [1]

Потпуна структрура се дефинише за тачно одређени број ентитета категорије и не може се више ниједан укључити, док непотпуна структура оставља могућност укључивања других ентитета категорије.

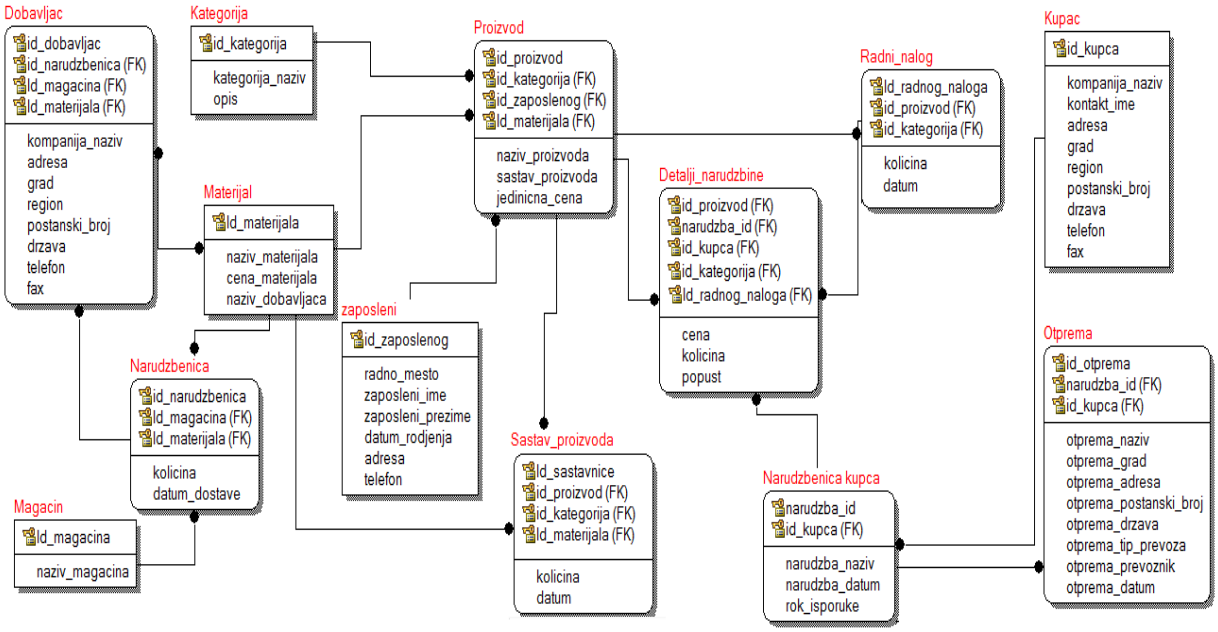
7.3 ERwin

ERwin алат, односно пуним називом AllFusion ERwin Data Modeler развило је предузеће Computer Associates International. Овај CASE алат служи за креирање и одржавање логичког и физичког модела података те за генерисање њима припадајуће физичке базе података. Интерфејс је кориснички оријентисан, међутим рад са ERwin-ом захтева високи ниво познавања поступака моделирања података. На слици 29. Приказан је почетни интерфејс за израду дијаграма.



Слика 29 - Почетни интерфејс за израду дијаграма

У ERwin алату могуће је израђивати логички, физички или и логички и физички модел истовремено. Логички модел садржи ентитете, атрибуте који описују ентитете, везе међу ентитетима и примарне и секундарне кључеве. Физички модел садржи информације о физичкој бази подтака, као што су типови атрибута у релацијским шемама, ограничењима, индекси као везе на друге релационе шеме, денормализационе таблице и слично. Логичко - физички модел садржи елементе и једног и другог модела. На слици 30 приказан је пример логичког модела података израђен у ERwin-у за предузеће Фори Текстил.



Слика 30 - Логички модел података у ERwin-у за предузеће Фори Текстил

7.4 Анализа модела података

Уколико модел података ефективно изражава пословне захтеве за подацима, то уједно не значи да је такав модел и добар модел. Његова структура може бити таква да умањује флексибилност, проширљивост или ствара непотребну редундантност. Код креирања модела података треба се придржавати следећих критеријума:
Добар модел података је једноставан - атрибути једног објекта треба да описују само тај објекат. Добар модел података је у основи нередундантан. Добар модел података треба да буде флексибилан и адаптибилан за будуће потребе.
Техника која се користи за побољшање модела података у фази припреме за физички дизајн базе података се назива анализа података. Анализа података је процес који припрема модел података за имплементацију једноставне, нередундантне, флексибилне и адаптибилне базе података. Ова техника се назива нормализација.
Нормализација је техника анализе података која организује, групише атрибуте података тако да формирају нередундантне, стабилне, флексибилне и прилагодљиве објекте. Овде нећемо приказати математички приступ који обухвата релациону алгебру и релациони рачун, већ ћемо се осврнути само на примену нормалних форми. Правилно пројектована база података мора имати структуру која осигурава да у раду са њом не може доћи до нежељених аномалија, као што су на пример, уништавање одређених података или неусклађеност између меморисаних података као последица ажурирања базе података.

Творац нормализације је Codd који је дефинисао поступак који доводи пројектанта релационе базе података до жељеног резултата. Codd је дефинисао три нивоа нормализације, да би касније било доказано да у неким изузетним примерима, релације које су у трећој нормалној форми, још увек показују извесне аномалије.

На трећу нормалну форму надовезује се Bouce - Codd-ова нормална форма која се бави проблемима преклапајућих кандидата за кључ.
Да би се могао описати поступак нормализације, потребно је претходно описати појмове функционалне, потпуне функционалне, транзитивне и вишезначне зависности.

7.5 Функционална зависност

Функционална зависност се може дефинисати између једноставног атрибута и сложеног кључа (више атрибута). Ако је свакој вредности атрибута A у релацији R прикључена само једна вредност атрибута B у истој релацији, онда је атрибут B функционално зависан од атрибута A. Ако је могуће сваком пару вредност атрибута A и B релације R прикључити тачно једну вредност C исте релације, тада је атрибут C функционално зависан о састављеном атрибуту A и B. На основу дефиниције функционалне зависности, **потпуна функционална зависност** се дефинише на следећи начин:
Атрибут B је потпуно функционално зависан од атрибута A исте релације, ако је функционално зависан од атрибута A, а не од неког саставног дела атрибута A (у случају када је атрибут A састављен). Другим речима, потпуна функционална зависност се посматра када је кључ табеле искључиво састављен од више атрибута. Као пример показаћемо табелу ПословиЗапослених.

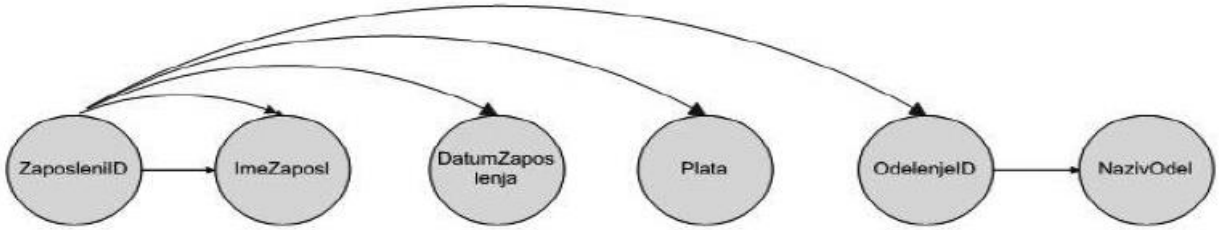
ZaposleniID	PosaoID	BrojČasova	OpisPosla
1001	202	40	Komerc
1005	222	35	Finans
1011	211	40	Komerc

Табела 1 - Послови запослених [2]

У овом примеру OpisPosla је функционално зависан од PosaoID (који је део кључа, па због тога није потпуно функционално зависан о кључу ове табеле). Атрибут BrojČasova је потпуно функционално зависан о целом кључу, те је број часова рада (40) одређеног радника (1001) на одређеном послу (202).

7.6 Транзитивна фукнционална зависност

Атрибут C је транзитивно функционално зависан од атрибута A, ако је функционално зависан од A и ако је функционално зависан од неког атрибута B који је и сам функционално зависан од A. Графички приказ опште дефиниције транзитивне функционалне зависности релације “запослени” се приказује на следећој слици.



Слика 31 - Пример транзитивне функционалне зависности за релацију запослени [2]

У овом примеру NazivOdel је функционално зависан и од примарног кључа ZaposleniID и од једног његовог некључног атрибута OdeljenjeID, што доводи до појаве транзитивне функционалне зависности. Транзитивна функционална зависност доводи до редундантности података у бази података.

Након дефинисања постојећих зависности, прелазимо на објашњавање поступка нормализације који започиње са првом нормалном формом. Процес нормализације представља трансформацију почетне табеле у једну или више коректних табела (релација) у којима су сви атрибути потпуно функционално зависни од кључа, а међусобно функционално независни. Циљ нормализације је да елиминише редундансу у атрибутима и са тиме омогући лакше одржавање интегритета података и једноставнију манипулацију.

7.7 Прва нормална форма

Основне операције одржавања базе података су: додавање нове н-торке у релацију, избацивање неке н-торке из релације и измена (ажурирање) вредности неког атрибута у релацији. Проблеми при извођењу ових операција као што су понављање ових операција више пута, избацивање једног логичког скупа података који доводи до нежељеног избацивања других података или логичко додавање се не може извршити, називају се аномалије у одржавању базе података.

Пре него што се крене у дефинисање табела и њихово нормализовање, пројектант информационог система у оквиру анализе полази од, на пример, неког формулара, као што су фактуре или досији радника и сл. Формирајући на основу тога једну релацију, пројектант приступа провери дате релације, односно приступа нормализацији, примењујући прву нормалну форму, која гласи: Релација је у првој нормалној форми уколико су све вредности њених атрибута атомске или другим речима нису дозвољени атрибути или групе атрибута “са понављањем”, тзв. “табеле у табели”.

Погледајмо још један пример ненормализоване релације “dobavljači” која је следећег облика:

DobavljacID	NazivDob	AdresaDob	Ptt_B	ProizvodID	NazivProizvoda
110	Sony	Decanska 10	11000	1234	Televizor
				1456	DVD
				2156	Walkman
				3522	Digitalne kamere
121	COOL	Makedonska	21000	2222	Muzička linija
				3321	Frižider
				2244	Usisivač
131	Tehnicom	Nusiceva 10	18000	3111	Notebook
				1111	Foto aparat

Табела 2 - Релација dobavljači [2]

Дата релација “dobavljači” показује извесне аномалије у одржавању базе података, као што су на пример: уколико се промени адреса добављача то се мора учинити на онолико места колико производа се набавља од тог добављача. Уколико се желе убацити у базу информације о новом производу, то се не може убацити све док га неки добављач не понуди итд. Ненормализовану релацију добављачи операцијом пројекције, декомпонујемо на следеће две:

DOBAVLJACI (DobavljacID, NazivDob, AdresaDob, Ptt_Br)
PROIZVOD (ProizvodID, NazivProizvoda, DobavljaID)

7.8 Друга нормална форма

Друга нормална форма (2НФ) - Ентитет или веза је у другој нормалној форми ако и само ако:

- је у првој нормалној форми, и
- ниједан његов атрибут који није део кључа не зависи само од неког дела тог кључа.

Другим речима, атрибути треба да зависи од целог скупа атрибута који учествују у кључу. Према томе, ако је кључ сведен на само један атрибут или ако садржи све атрибуте, ентитет или веза су обавезно у 2НФ. Ентитет или веза могу да буду у другој нормалној форми у односу на један кандидат за кључ, а да не буду у односу на неке друге кандидате.

Сл. 32 приказује ентитет Производ који описује производе које испоручују разни добављачи. Претпостављамо да једна добављач може да испоручи више производа и да један производ може да испоручује више добављача. У том случају ни opisProizvoda ни imeDobavljaca не могу сами да буду кључ ентитета Производ. Насупрот томе, кључ (opisProizvoda, imeDobavljaca) јесте идентификатор ентитета Производ. Међутим,

атрибут adresaDobavljaca зависи само од дела кључа - imeDobavljaca. Једно решење (неадекватно) је да се уведе нови атрибут idProizvoda који ће бити кључ, чиме ентитет прелази у 2НФ.

Сл. 32 - Пример превођења у другу нормалну форму [6]

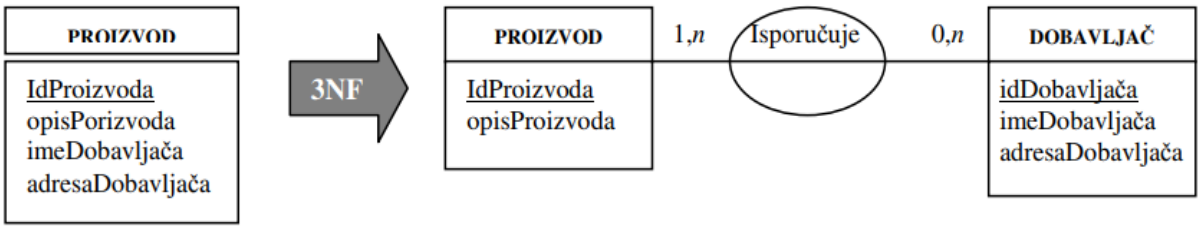
Овде претпостављамо да један добављач испоручује више производа и да више добављача може да испоручује један исти призвод.

7.9 Трећа нормална форма

Трећа нормална форма (3НФ) - Ентитет или веза је у трећој нормалној форми ако и само ако:

- је у другој нормалној форми, и
- њедан његов атрибути који није део кључа не зависи од неког скупа атрибута који нису у кључу.

Према томе, ако је ентитет или веза у другој нормалној форми и има само један атрибут који није у кључу он је обавезно у 3НФ. Ентитет или веза могу да буду у трећој нормалној форми у односу на један кандидат за кључ, а да не буду у односу на неке друге кандидате.
Током превођења у 3НФ могу се открити и формулисати неки скривени ентитети као што приказује Сл. 33.



Слика 33 - Пример превођења у трећу нормалну форму [6]

У овом примеру атрибут adresaDobavljaca зависи од imenaDobavljaca (заправо idDobavljaca)

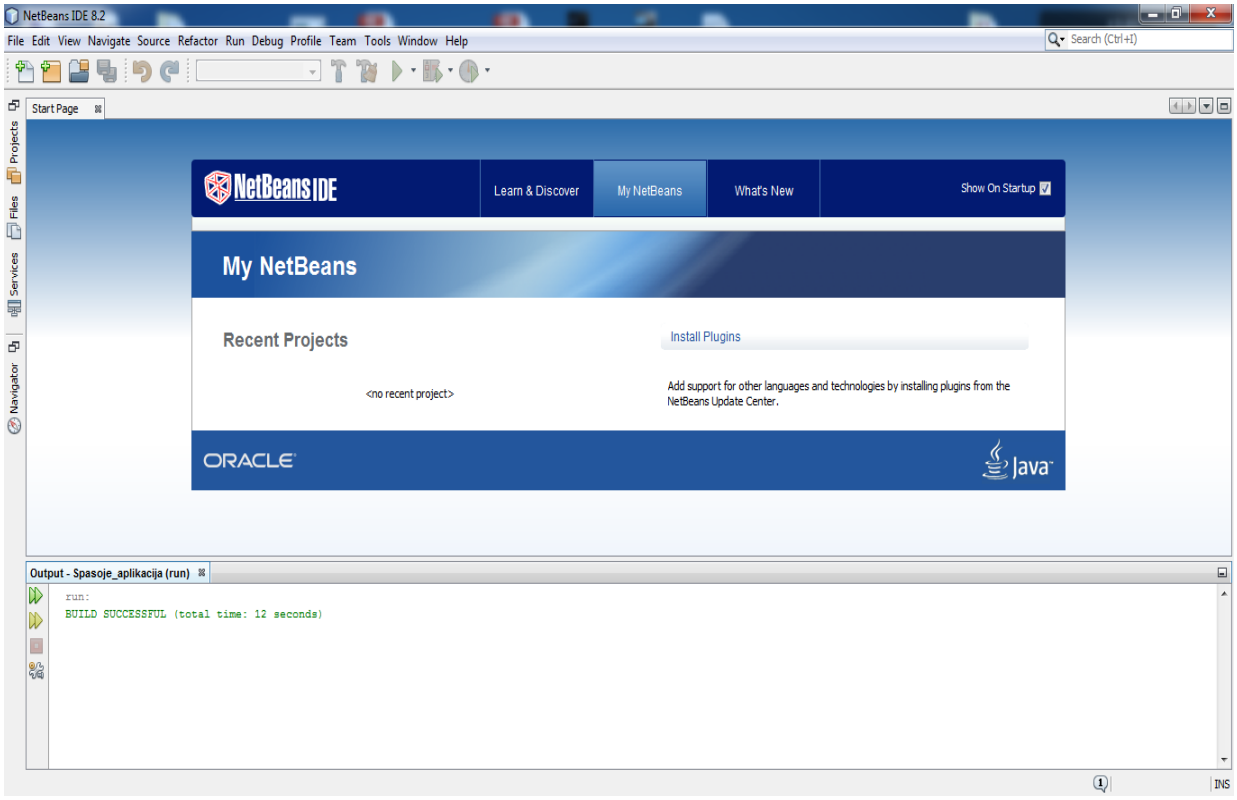
8. Јава апликација

8.1 Јава

Јава је програмски језик који се користи за израду и развој великог броја апликационих софтвера и његову имплементацију у најразличитија мултиплатформска окружења. Јава програмирање је једно од најзаступљенијих данас, посебно имајући у виду све већу примену у изради апликација за мобилне телефоне и таблет уређаје кроз Андроид оперативни систем. Јава платформе имају широку употребу у програмирању и примењеним софтверским решењима, које се крећу од најједноставнијих дигиталних уређаја, mp3 плејера, мобилних телефона, па све до комплексних веб сервера и корпоративних апликација.
Основне карактеристике и предности програмског језика Јава јесте то што је објектно-оријентисан програмски језик што подразумева лакше разумевање и једноставније проширење и одржавање, као и то што је машински независтан, односно може радити на сваком уређају независно од платформ.
Јава програм се не извршава директно на оперативном систему рачунара. Покреће се у стандардизованом окружењу које се зове Јава платформ. Последица тога је да се Јава програми могу покренути на широком спектру оперативних система.
Јава платформа се састоји од јава виртуелне машине (основна компонента) и Java application programming interface-а (API) - скуп компоненти који обезбеђују писање интерактивних апликација у Јави.

8.2 NetBeans

NetBeans је интегрисано развојно окружење (IDE) првенствено намењено развоју Јава технологија, али исто тако пружа доста додатних могућности које му омогућавају да се једнако ефикасно може користити за развој рачунарских програма и у осталим програмским језицима као што су C, C++, PHP, Fortran, Python, Rubi и други. NetBeans једнако добро ради на различитим платформама као што су Windows, Linux, BSD. Подражава различите технологије и алате који побољшавају развојни процес апликације. На слици 34 је приказан почетни прозор када се отвори радно окружење NetBeans-а.



Слика 34 - Почетни прозор радног окружења NetBeans-а

За потребе нашег рада одрађена је апликација која има за циљ да симулира принцип на коме предузеће "Фори Текстил" врши своје пословање. У даљем раду биће објашњен рад дате апликације.

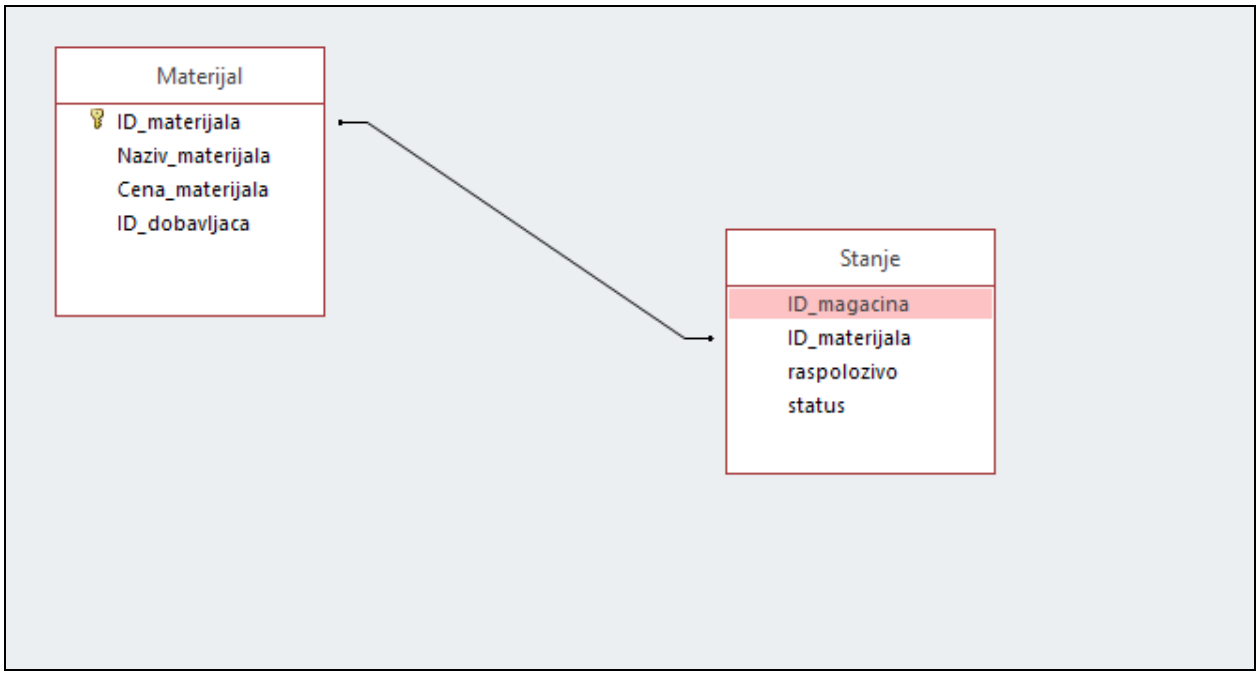
8.3 Јава апликација

База података се састоји од укупно 9 табела:

1. **Добављачи** - бележи све добављаче са основним подацима. Примарни кључ је ID.

2. **Купац** - информације о свим купцима који су регистровани. Могуће је пословати са фирмом само ако си регистрован купац, а ако ниси, потребна је регистрација.

3. **Материјал** - праћење материјала. Материјал је повезан са магацином (слика 35), тако да се у сваком тренутку прати стање материјала:



Слика 35 - Праћење стања материјала

4. **Отпремница** - на основу ID отпремнице се претражују сви послови који су завршени. Отпремница је директно повезана са купцем, и ту се бележе основни подаци о извршеним пословима, ценама и слично.

5. **Поручивање материјал** - праћење поруџбине материјала и директно је повезано са ID материјала и ID добављача.

6. **Поруџбина** - Праћење поруџбина, односно послова које је потребно одрадити.

7. **Производ** - Праћење свих произода који су регистровани у предузећу

8. **Стање** - приказ стања материјала у магацину

9. **Запослени** - праћење свих запослених у фирми. Свако од њих има привилегије (магацин, комерцијала, админ и слично) ради правилног функционисања апликације.

8.4 Опис кода апликације

Апликација се састоји из 3 главна пакета:

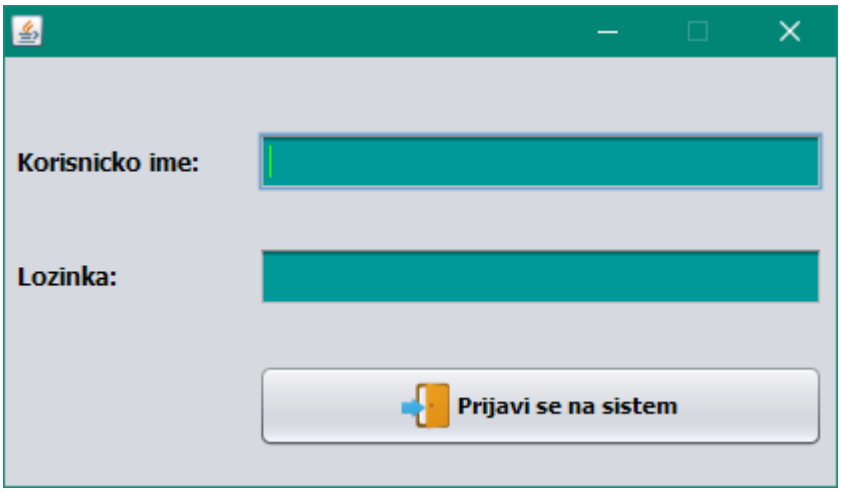
- Библиотеке
- Главни пакет и
- Иконе

Унутар првог пакета се налазе све библиотеке које су потребне за функционисање апликације, а пре свега се односе на библиотеке за рад са базом података. У овом случају се користи ucanaccess.jar као главна библиотека и преко које се остварује комуникација апликације са базом података.

Други пакет се састоји из 3 главна JABA фајла:

a) Glavni.java
b) Komercijala.java
c) Magacin.java

а) Унутар главног пакета се врши пријава на систем (слика испод)

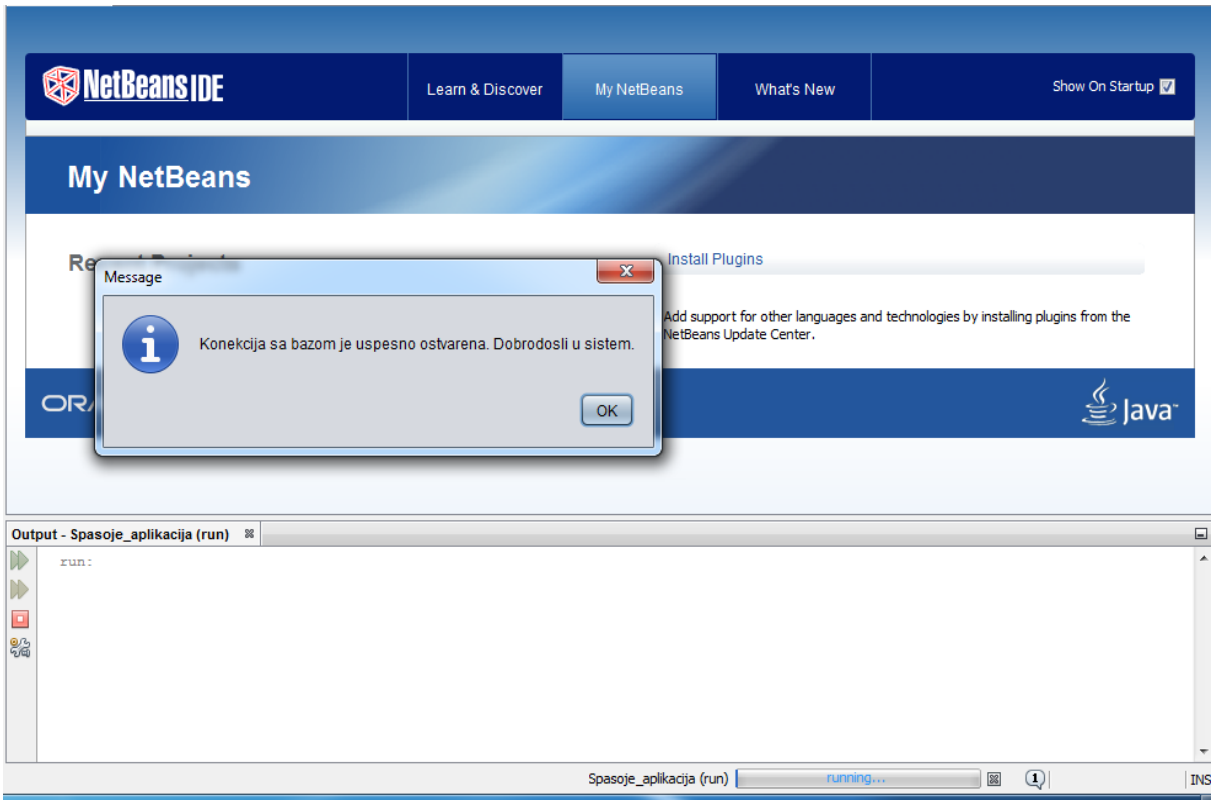


Слика 36 - Пријава у систем

Да би се извршила пријава, апликација мора да има контакт са базом података. Тај контакт се остварује употребом претходно поменутих библиотека и наравно кода у апликацији. Главна метода која се користи за остваривање конекције је :

```
public void konekcija()
   {
      try
      {
      connection                                                    =
DriverManager.getConnection("jdbc:ucanaccess://Spasoje_baza.accdb");
      statement = connection.createStatement();
      JOptionPane.showMessageDialog(rootPane, "Konekcija sa bazom je
uspesno ostvarena. Dobrodosli u sistem.");
      }
      catch (Exception e)
      {
         JOptionPane.showMessageDialog(rootPane, e);
      }
   }
```

Дакле, најпре се креира конекција, где се врши подешавања назива базе, у овом случају је Spasoje_baza.accdb. Уколико је конекција успешно остварена, издаје се порука кориснику (слика 37).



Слика 37 - Порука када се конекција оствари

Што се тиче препознавања корисника који је пријављује, то се врши преко методе која служи за преузимање вредности текстуалног поља и смештање у промењиве „корисничко“ и „лозинка“, након тога се врши селектовање свих података из базе на основу тражених података:

```
String korisnicko = jTextField1.getText();
    String lozinka = jPasswordField1.getText();
```

Унутар try-catch блока се налази sql упит, где се врши селектовање запосленог који задовољава те критеријуме претраге. Након тога се смешта у resultSet (rs) и уколико је услов претраге задовољен, односно ако је пронађен запослени са тим корисничким именом и лозинком, смешта се радно место у променљиву, и проверава се назив радног места. У зависности од радног места које се прикупи из базе отвара се одређени прозор, односно магацин или комерцијала.

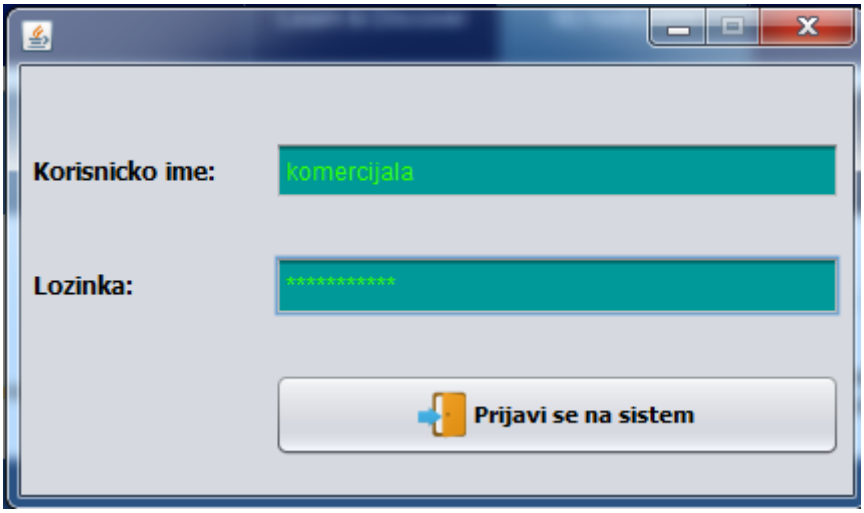
```
try
    {
        String upit = "Select * from Zaposleni where korisnicko_ime='"+korisnicko+"' and lozinka='"+lozinka+"'";
        rs = statement.executeQuery(upit);

        if (rs.next())
        {

            String radno_mesto = rs.getString("radno_mesto");
```

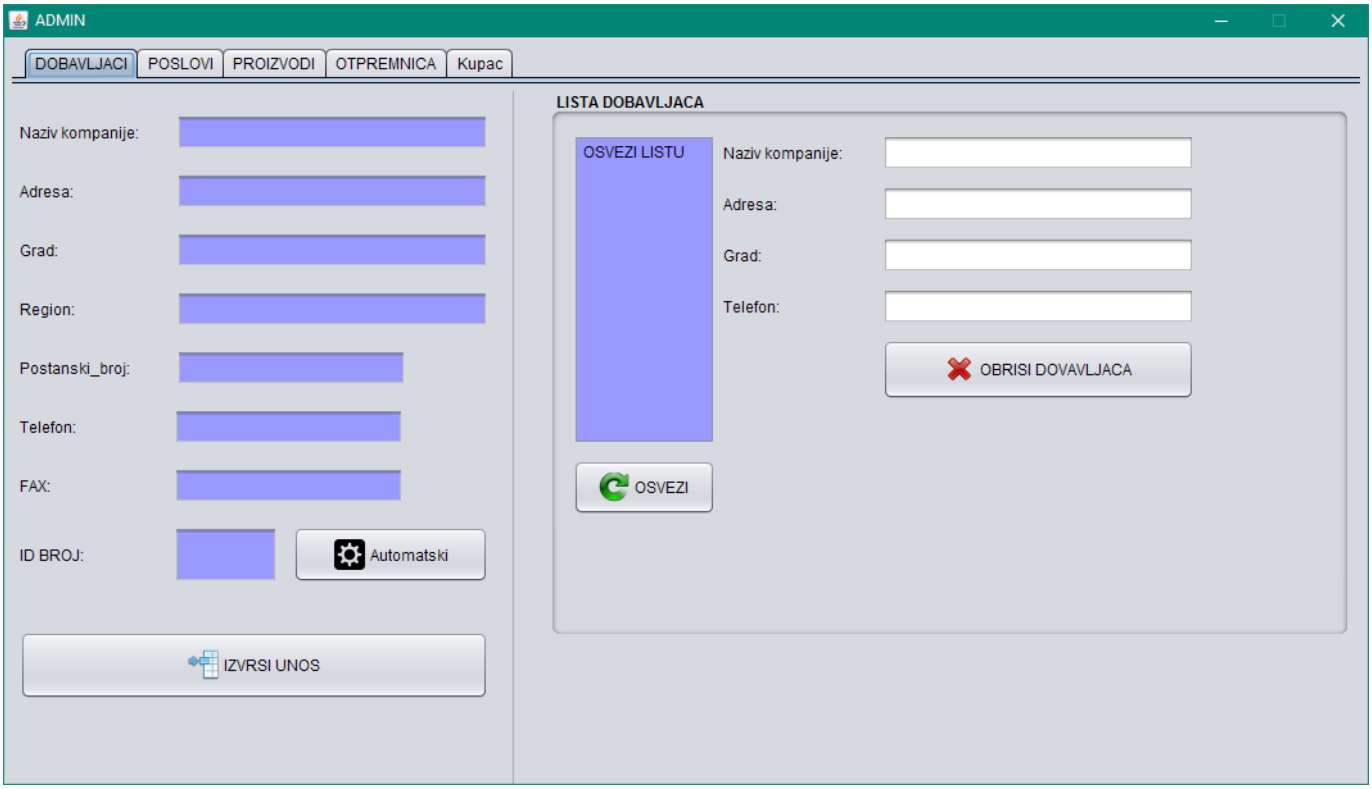
b) Комерцијала

Да би се пријавили на сектор комерцијале (слика 38), потребно је да унесемо корисничко име "комерцијала" што је уједно и лозинка. Када то укуцамо потребно је само да кликнемо на дугме "Пријави се на систем" и аутоматски нас пребацује на послове које обавља комерцијала.



Слика 38 - Пријављивање на сектор комерцијале

Унутар овог дела постоји неколико картица које је могуће користити (слика 39).



Слика 39 - Почетни прозор након логовања на сектор комерцијале

- **Добављачи**

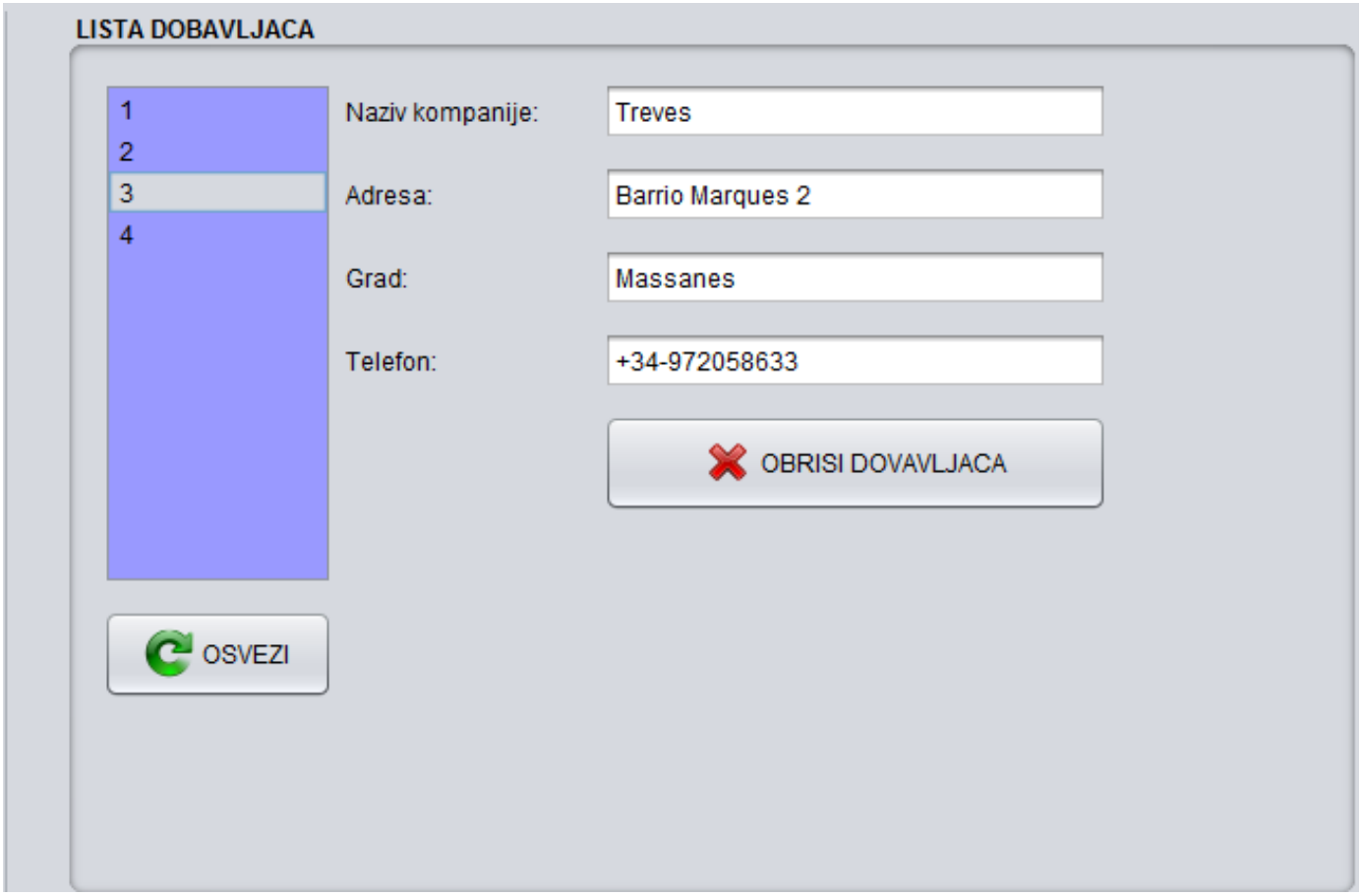
Унутар картице "добављачи" врши се додавање нових добављача, такође прикупљањем информација из поља са леве стране и коришћењем стандардног SQL упита за унос који изгледа:

```
String unos_u_bazu = "Insert into Dobavljaci values
("+id+",'"+naziv_dobavljaca+"','"+adresa+"','"+grad+"','"+region+"','"+postanski
+"','"+telefon+"','"+fax+"')";

        st.executeUpdate(unos_u_bazu);

        JOptionPane.showMessageDialog(rootPane, "Unos je uspesno izvrsen");
```

Након што унесемо добављаче, комплетну листу унетих добављача (слика 40) можемо прегледати кликом на дугме “освежи”. Након тога ми можемо да одаберемо унетог добављача. Податке о добављачима можемо сваког тренутка да променимо, као и да одбришемо уколико више нећемо сарађивати са њима.



Слика 40 - Приказ листе добављача

Дакле, у табелу "добављачи" се уносе вредности које одговарају редоследу у бази, а те вредности се прикупљају преко методе getText().
Унутар картице " добављачи " такође постоји и листа за приказ свих добављача. То се постиже преко следећег кода:

```
DefaultListModel df = new DefaultListModel();

    try
    {
       String preuzmi = "Select * from Dobavljaci";
       rs = st.executeQuery(preuzmi);

       while (rs.next())
       {
          String id = rs.getString("ID");
          df.addElement(id);
       }
```

```
    jList1.setModel(df);
}
catch (Exception e)
{
    JOptionPane.showMessageDialog(rootPane, e);
}
```

Дакле, креира се најпре DefaultList модел који је потребан као модел, односно главна карактеристика једне листе у Јава програмирању. Креира се SQL упит за преузимање свих добављача, али се унутар листе додаје само ID унутар WHILE петље. Дакле, докле год постоји неки добављач, у DefaultList модел се додаје нови елемент.

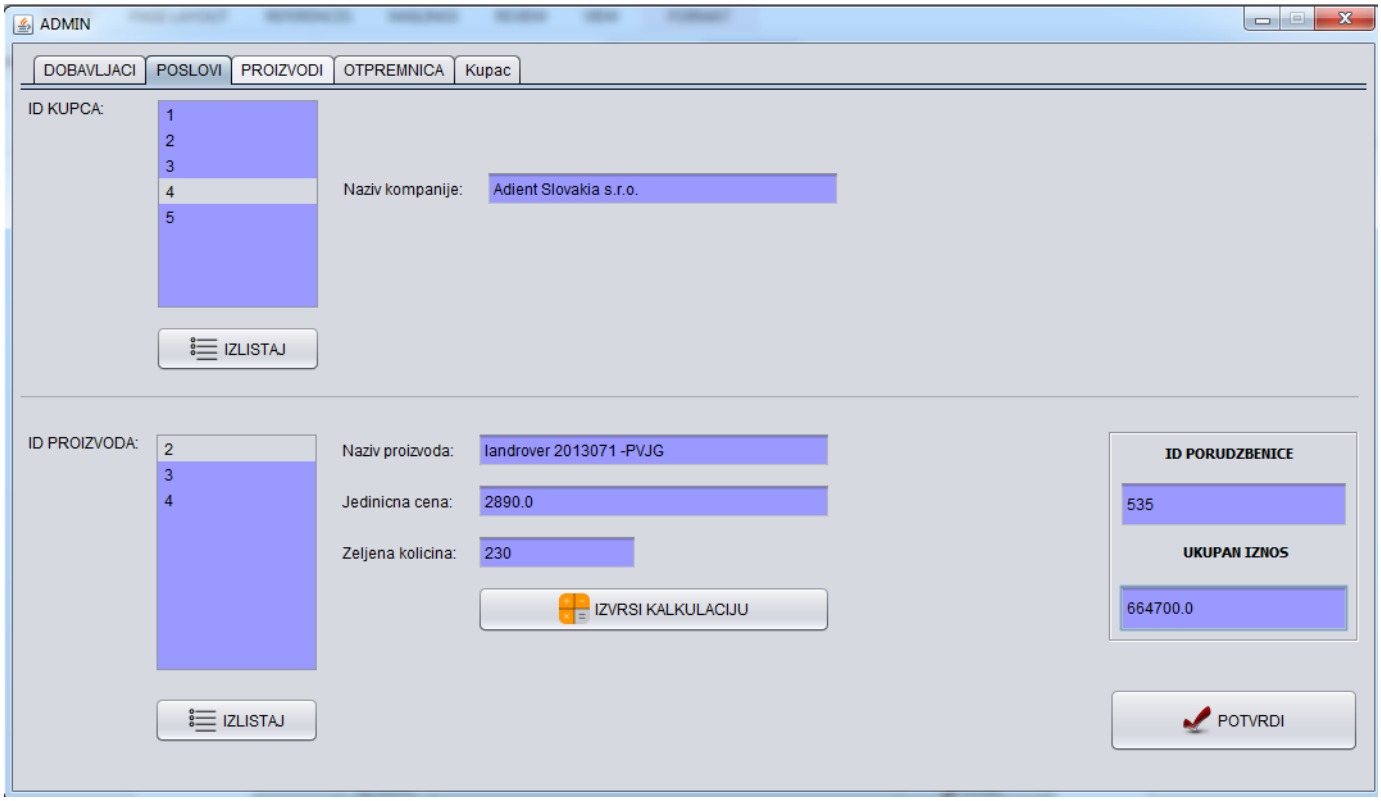
Брисање добављача се врши класичним SQL упитом:

```
String upit_brisanje = "Delete * from Dobavljaci where ID='"+selektovano+"'";
        st.executeUpdate(upit_brisanje);
```

Дакле, променљива „селектовано“ је оно што је кликнуто на листи, након што корисник одабере са листе ID добављача, селектовано добија нову вредност и брише се та врста у табели са тим ID.

- **Послови**

Унутар ове картице (слика 41) је могуће извршити листање свих доступних послова, односно праћење поруџбина, где је потребно одабрати купца и ID производа који тај купац жели.
У нашем примеру смо ставили да купац "Adient Slovakia s.r.o" жели да купи наше навлаке за наслон главе за модел Land Rover идента 2013071 - PVJG. Јединична цена за овај један наш производ је 2890 динара. Када купац наручи 230 наших навлака, наш систем ће израчунати да је укупна цена 664700 динара.



Слика 41 - Табела послови

Што се тиче програмирања у овом делу, врло је једноставно. Листа се попуњава на исти начин као и у претходном објашњењу, где се за дати ID у поље „назив компаније“ исписује назив компаније. На исти начин се и на основу ID производа исписују подаци о том проиводу. И на крају, се врши калкулација:

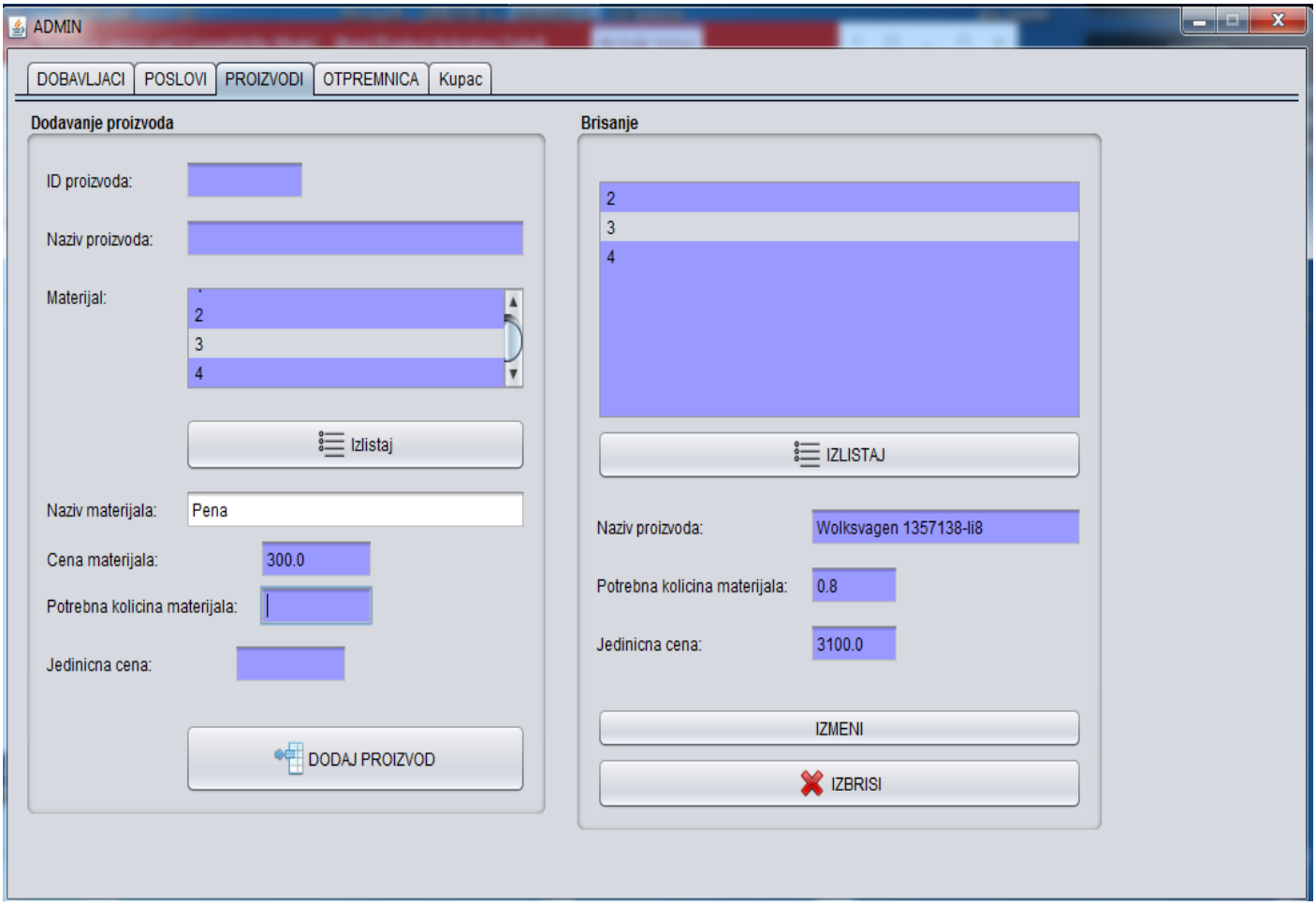
```
double x1 = Double.parseDouble(jTextField23.getText());
      double x2 = Double.parseDouble(jTextField24.getText());

      double ukupno = x1*x2;

      jTextField26.setText(""+ukupno);
```

- **Производи**

У наредној картици (слика 42) се врши додавање нових производа као и брисање.



Слика 42 -Табела производи

Дакле, додељује се ID новом производу, и додељује се колико је потребно материјала за дати производ. Ово је битан параметар, јер се на основу овога повлачи материјал из магацина при пуштању производа у производњу.
Други део ове картице се користи за брисање производа из базе. Односно најпре се врши листање ID свих производа и на основу тих ID се врши листање података о датом производу. Такође, остављен је простор за измену и брисање производа на основу ID. Измена се врши такође на основу ID-а, и то преко sql упита и наредбе update:

```
String naziv = jTextField18.getText();
        double potrebna_kolicina = Double.parseDouble(jTextField20.getText());
        double cena = Double.parseDouble(jTextField19.getText());
        try
        {
            String update = "UPDATE Proizvod SET Naziv_proizvoda='"+naziv+"',
Potrebno_materijala='"+potrebna_kolicina+"',  Jedinicna_cena='"+cena+"'
WHERE ID_proizvoda='"+selektovan_proizvod+"'";
```

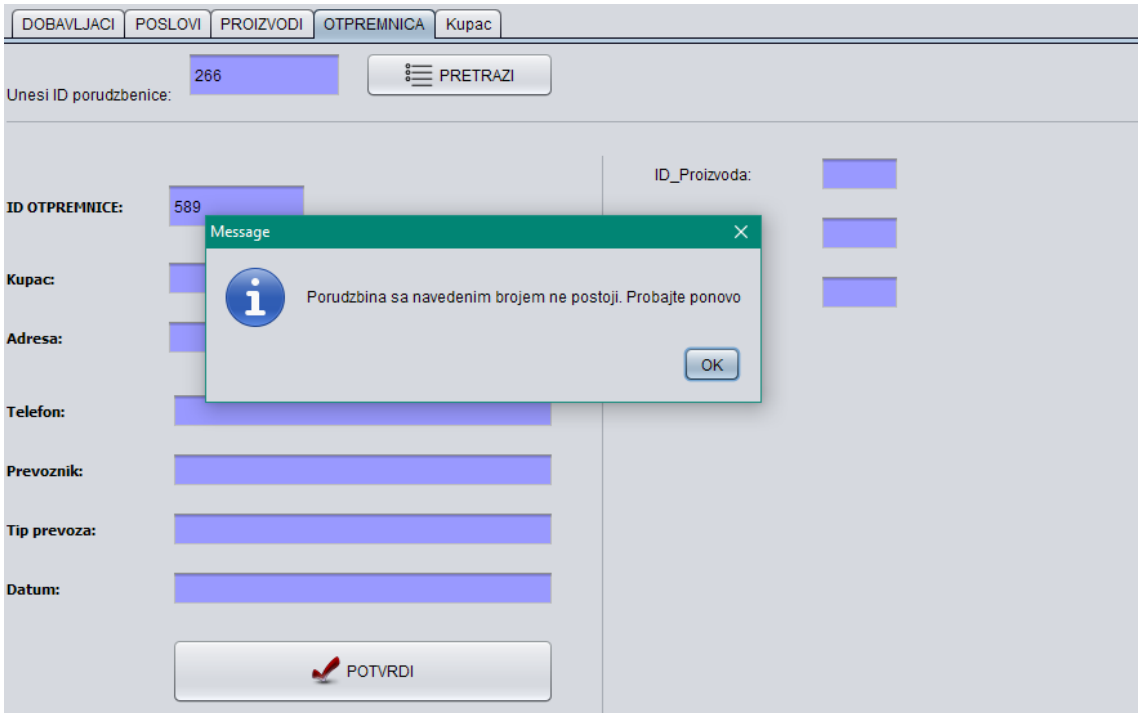
```
            st.executeUpdate(update);
            JOptionPane.showMessageDialog(rootPane, "Proizvod je uspesno
izmenjen");

        }
        catch (Exception e)
        {
            JOptionPane.showMessageDialog(rootPane, e);
        }
```

Дакле, врши се “update” производа и подешавање свих података поново. Уколико корисник изврши промену неких података врше се промене, уколико не, подаци остају исти.

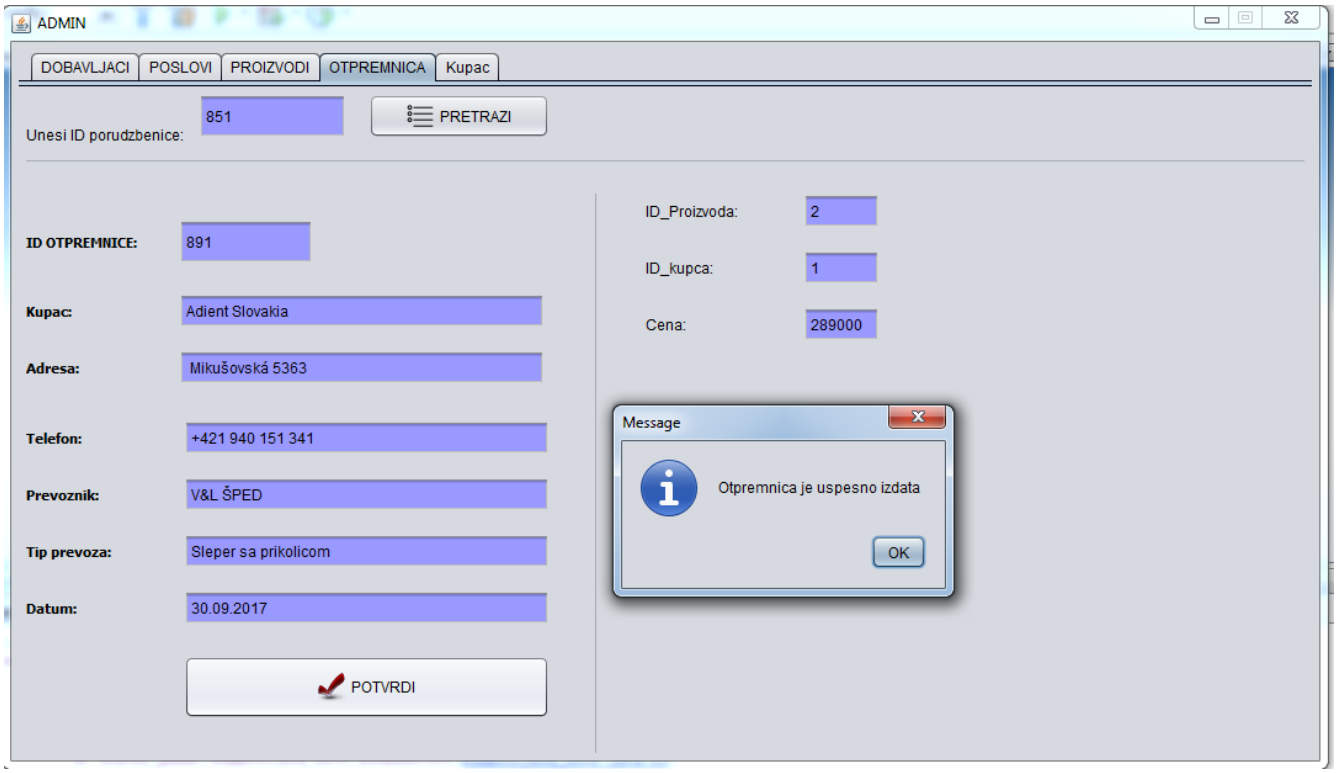
- **Отпремница**

У овој картици (слика 43)је могуће вршити претрагу отпремница, и издавање истих. Дакле, уколико се унесе број који није активан излази следећа грешка:



Слика 43 - Картица отпремница

Када правимо отпремницу морамо унети све податке о купцу, превознику, датум када робу шаљемо, као и обавезно на фактури уписати цену робе коју шаљемо купцу (слика 44).



Слика 44 - Попуњена картица отпремница

Што се тиче кода који се користи за прављење отпреме користи се класичан sql упит:

```
String selektovanje = "Select * from Porudzbina where ID_porudzbine
="+pretragaPorudzbine+" and Status ='proizvodnja'";
        rs = st.executeQuery(selektovanje);
```

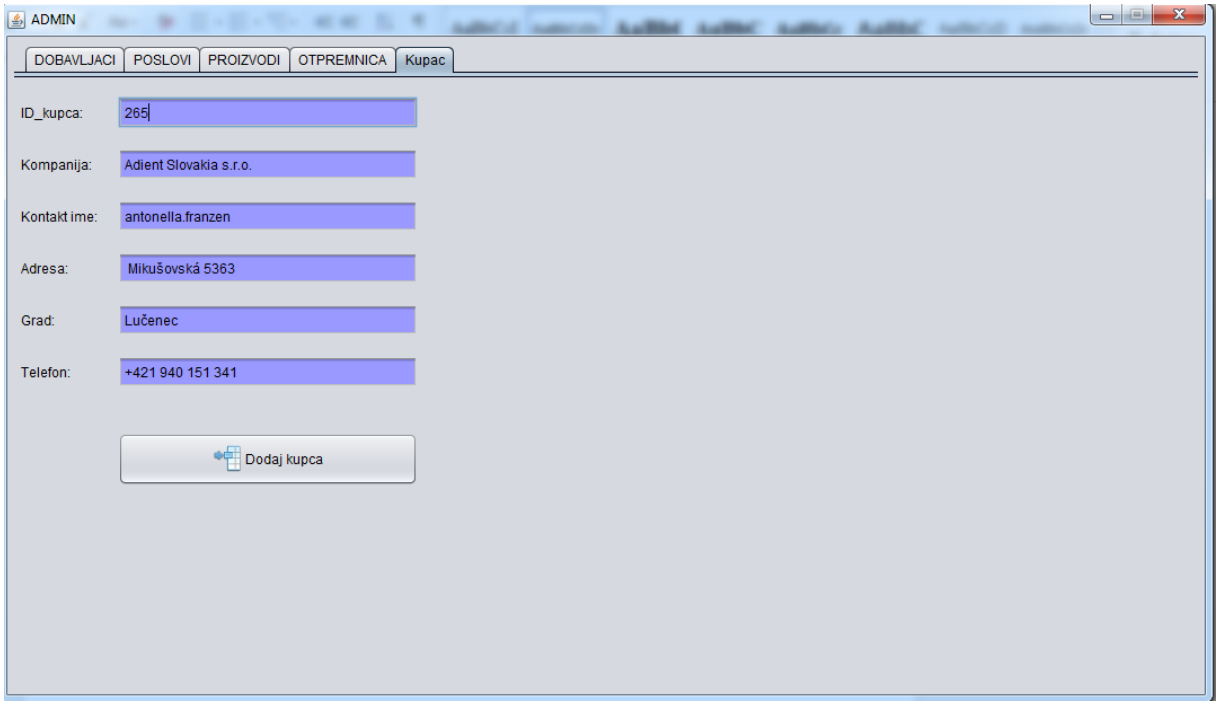
односно, селектују се све поруџбине за који одговара дати ID и где је статус производња, чиме се спречава да се испоручи производ који није ни ушао у производњу, то је такозвана техничка грешка.

- **Купац**

Последња картица је „купац“ где се врши једноставно додавање купца у базу података, где се уносе основни подаци о купцу (слика 45).

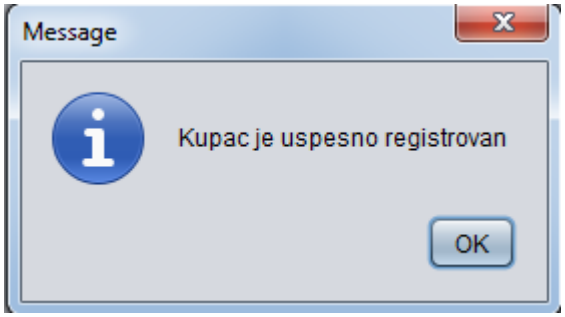
```
        String unos = "Insert into Kupac
values('"+id_kupca+"','"+kompanije+"','"+ime+"','"+adresa+"','"+grad+"','"+telef
on+"')";
```

```
st.executeUpdate(unos);
    JOptionPane.showMessageDialog(rootPane, "Kupac je uspesno
registrovan");
```



Слика 45 - Картица купац

Када кликнемо на додај купца, систем нам приказује следећу поруку (слика 46).



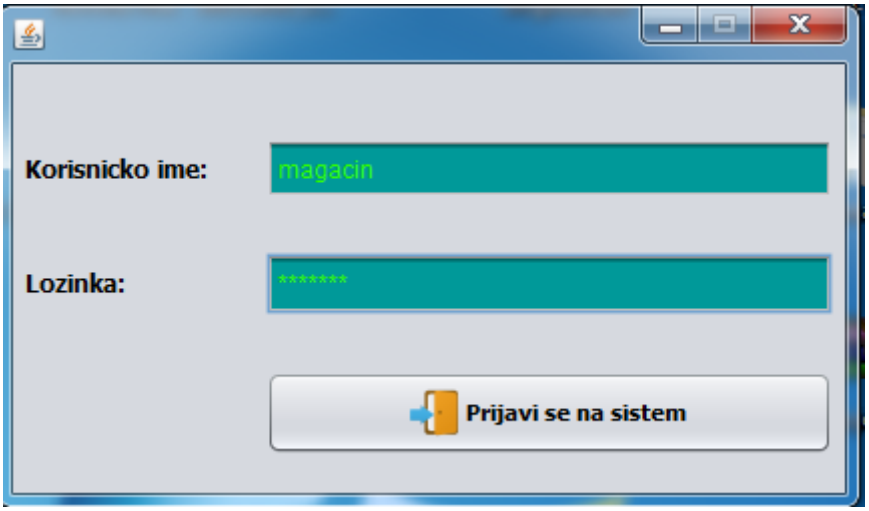
Слика 46 - Обавештење да је купац успешно регистрован

c) Магацин

Ако радно место одговара „магацин“ онда се отвара прозор магацин креирањем новог објекта класе Магацин, и поставља се прозор да буде видљив, у супротном се отвара комерцијала ако име одговара „комерцијала“ и тако редом. Корисничко име за приступање магацину је магацин, што је уједно и лозинка. Када то укуцамо потребно је само кликнути на пријави се на систем (слика 47).

```
if (radno_mesto.equals("magacin"))
            {
               Magacin magacin = new Magacin();
               magacin.setVisible(true);
            }

            if (radno_mesto.equals("komercijala"))
            {
               Glavni_paket.Komercijala_novo admin = new Komercijala_novo();
               admin.setVisible(true);
            }       }
      }
      catch (Exception e)
      {
         JOptionPane.showMessageDialog(rootPane, e);
      }
```



Слика 47 - Пријава на сектор магацина

Битно је напоменути да се конекција покреће одмах на почетку извршавања програма, односно када се отвори главни прозор, и то преко конструктора:

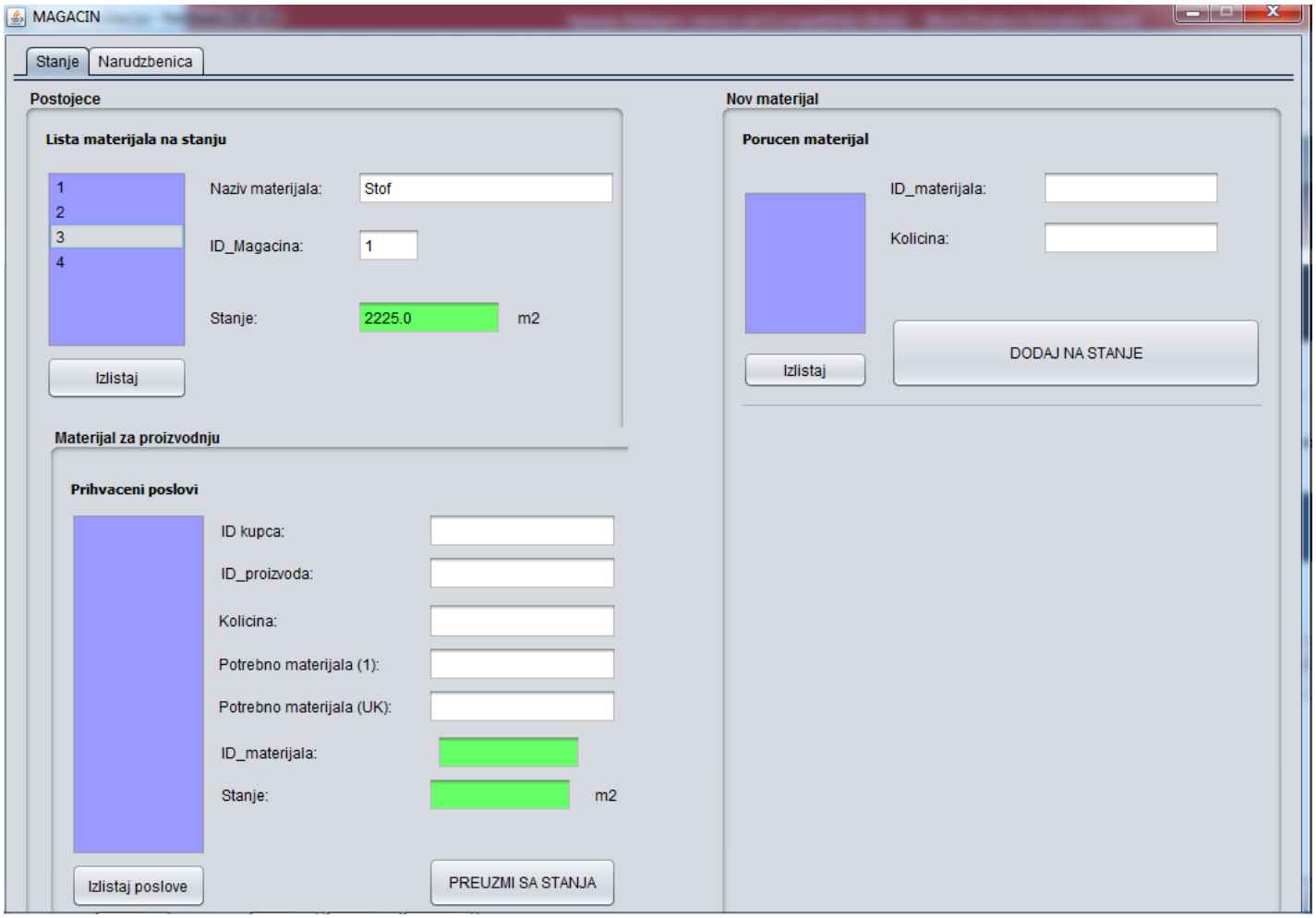
```
public Glavni_gui() {
      konekcija();
```

```
        initComponents();

    }
```

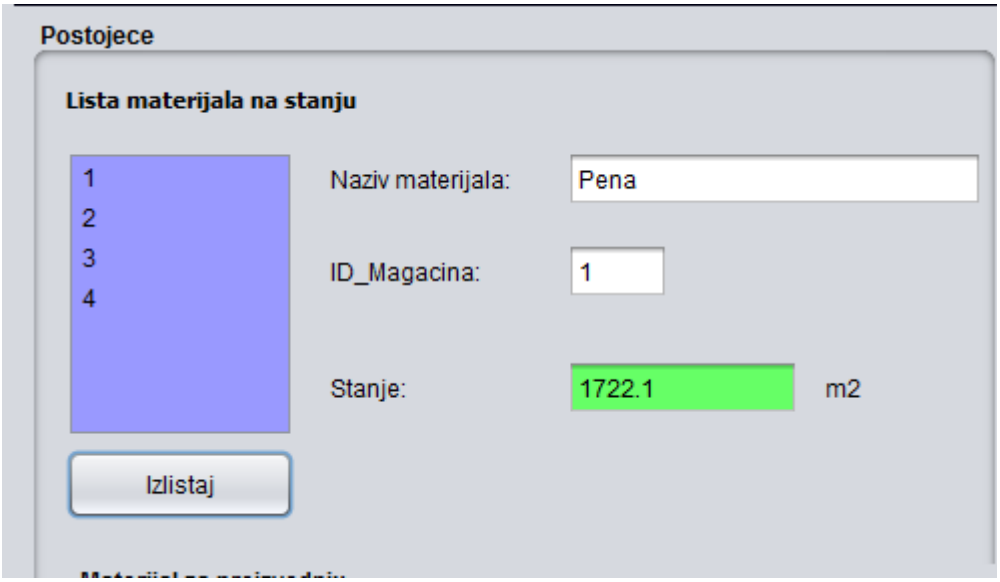
Прозор магацин (слика 48) има у себи 2 картице:

- Стање и
- Наруџбеница

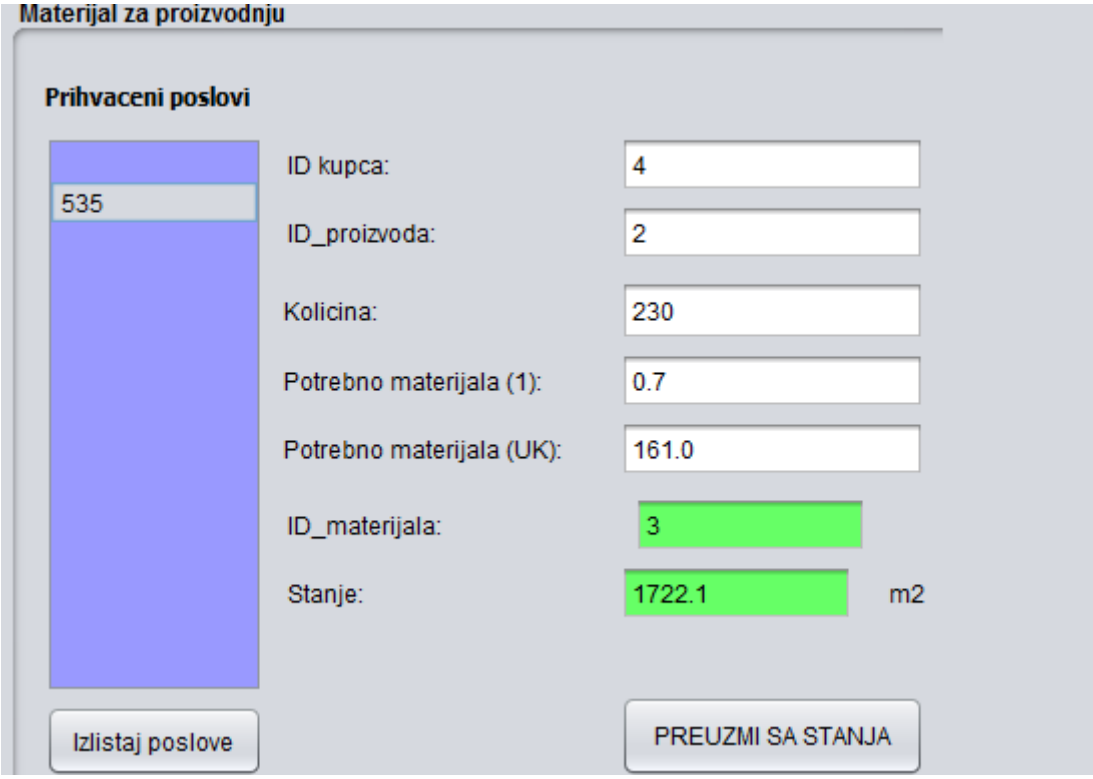


Слика 48- Картица стање

Преко картице стање, ми пратимо наше залихе материјала које су нам потребне за производњу. У овој листи се налазе тренутно 4 врсте материјала. А то су штоф, пена, винил и кожа. На следећој слици можемо видети да тренутно стање пене у магацину је 1722.1 m2.

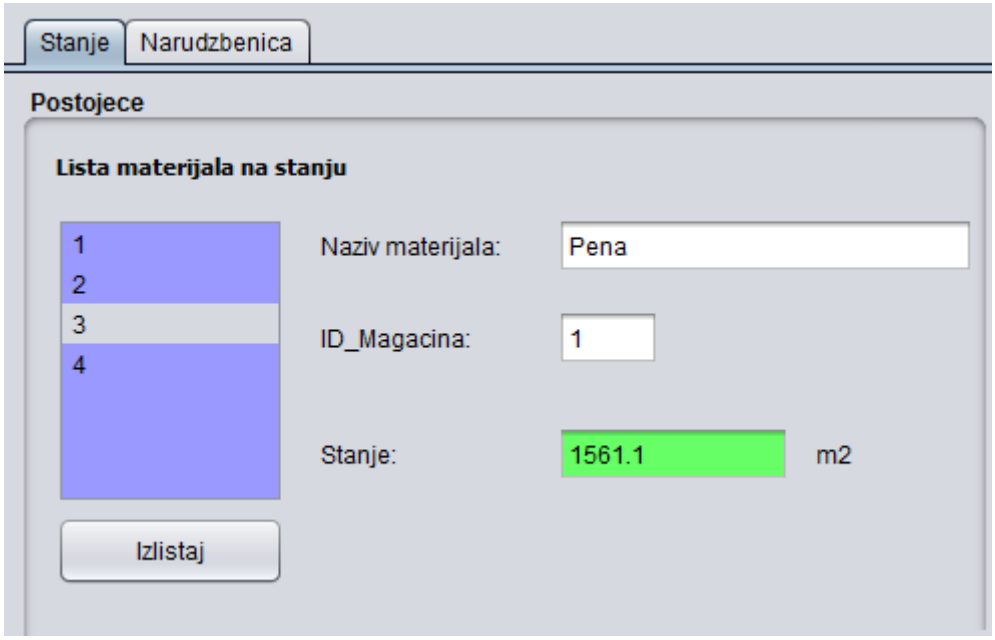


Слика 49 - Стање материјала (пене) на магацину



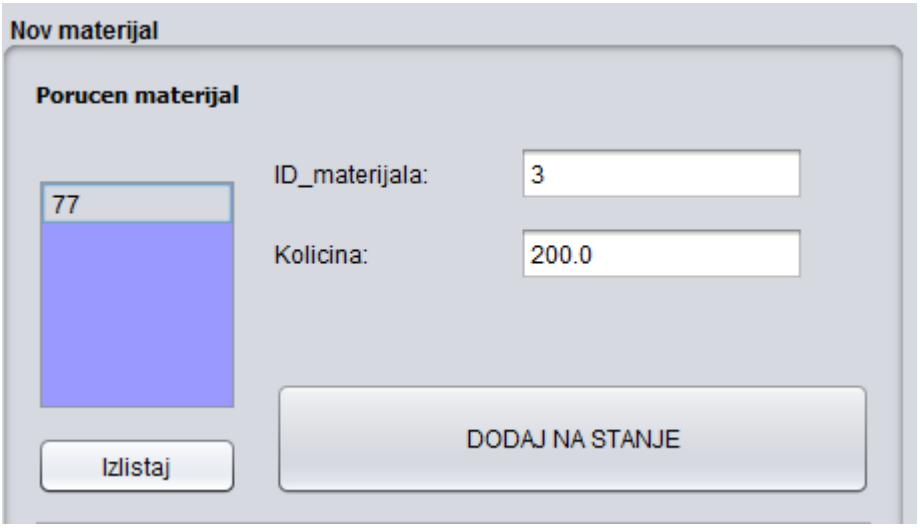
Слика 50 - Преузимање материјала из магацина

Када кликнемо на излистај послове, ми видимо да је за посао који производња треба да одради потребно 161 m2 пене. А пошто ми имамо на стању 1722.1 m2, ми ћемо им пребацити ту количину системски тако што кликнемо преузми са стања (слика 50).
Након што им пребацимо, стање у магацину што се материјала од пене тиче биће 1561,1 m2 (слика 51).



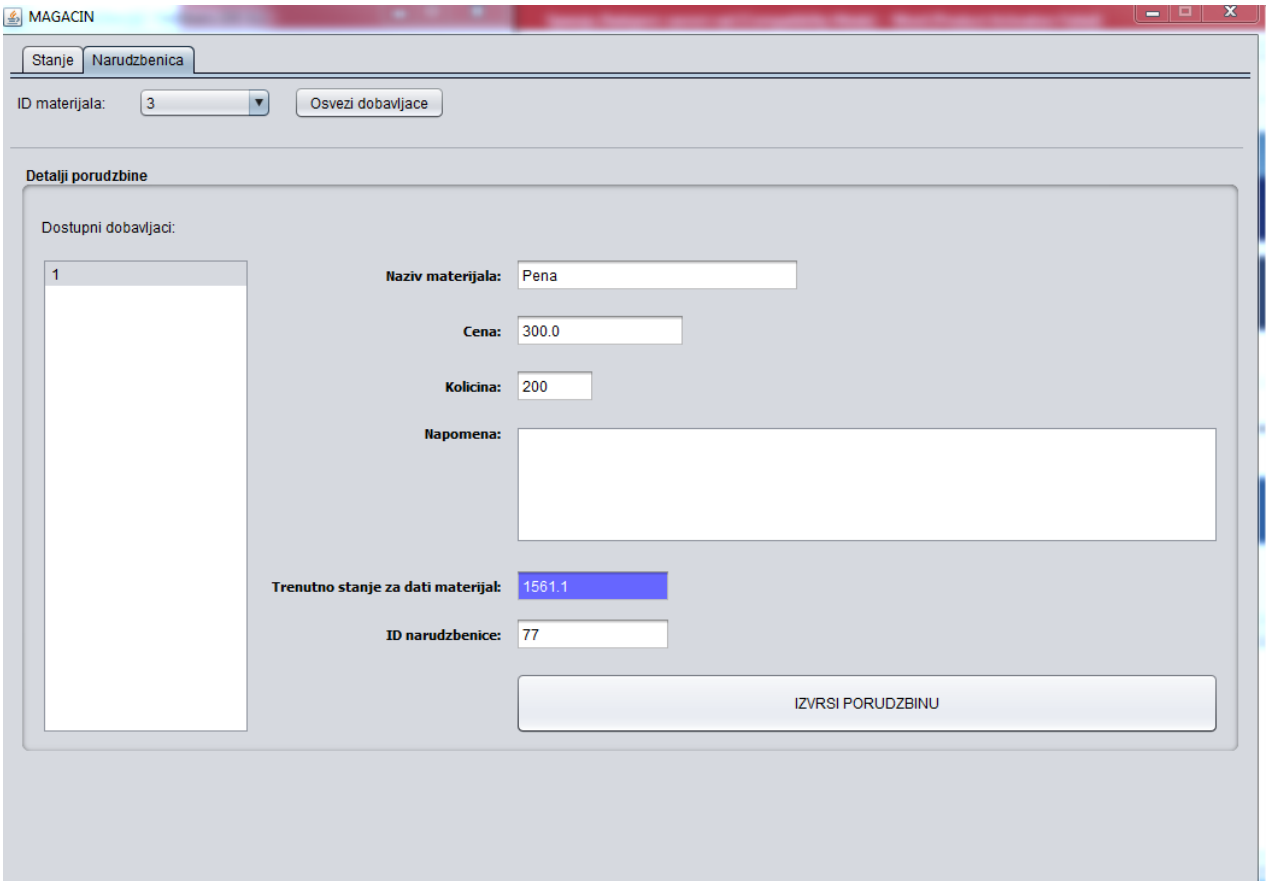
Слика 51 - Ново стање магацина

Такође можемо да додајемо нов материјал на стање магацина, као што је приказано на следећој слици.



Слика 52 - Додавање новог материјала на стање магацина

Али да би смо додали нову количину материјала на стање ми ћемо направити нову наруџбеницу за додатни материјал (слика 53).



Слика 53 - Прављење наруџбенице

9. Закључак

У раду је дат "Функционални модел рада једног предузећа" реализован употребом графичког језика IDEF0, програмом MS Visio. IDEF је графички језик који садржи скуп стандардизованих метода за функционално моделирање пословних процеса и објеката, а самим тим и унапређење процеса пословања. Урађени су дијаграм контекста, стабло активности. Информационо моделовање података извршено је дефинисањем потребних ентитета, атрибута и релација. Урађен је декомпозициони дијаграм модела предузећа где је јасно приказано пословање предузећа. Логички модел предузећа који је израђен у програму Erwin је такође приказан у раду. Исти тај модел урађен је и у MS Access-у што уједно представља и базу података за нашу апликацију у Net-Beans-у.

У мастер раду рађена је апликација која симулира рад система који се користи у предузећу Фори Текстил. Предност оваквог система је брже и ефикасније пословање. Базу података не чине само организоване табеле као носиоци података, већ је то комплетан програм који може прорачунавати, филтрирати податке, итд. Поред могућности уноса података у форме и подформе, апликација нам омогућава да вршимо прегледе свих добављача и купаца. Да увек имамо евиденцију колико материјала имамо на стању, да вршимо нове наруџбине и отпремнице и још много тога. Подаци су доступнији, лакше их је унети и селектовати.

10. Литература

[1] A.Veljović - Razvoj informacionih sistema i baze podataka. 1996 god.
[2] A. Njegus - Poslovni informacioni sistemi. 2009 god
*[3] Michael Lee, Gentry Bieker: **Mastering SQL Server 2008**, Wiley Publishing, Inc., Indianapolis, Indiana, 2009.*
[4] Rainer K., Turban E.: Introduction to Information Systems, John Wiley & Sons, Inc., 2009.
[5] C# базе података, CET Computer Equipment and Trade, Београд, 2003.

Интернет сајтови:

[6]http://poincare.matf.bg.ac.rs/~cvetana/Nastava/Materijal/RelacioniModel_1213.pdf
[Pristupljeno 01.09.2017]

[7] https://www.microsoft.com/en-us/*[Pristupljeno 03.09.2017]*