Висока технолошка школа струковних студија Аранђеловац

**Кристина Б. Булатовић**

**Информациони систем библиотеке**

Завршни рад

Информационе технологије

Аранђеловац, 2017

Проф. др , ментор

Професор струковних студија Високе технолошке школе струковних студија у Аранђеловцу

Проф. др , члан Комисије

Професор струковних студија Високе технолошке школе струковних студија у Аранђеловцу

Проф. др , члан Комисије

Професор струковних студија Високе технолошке школе струковних студија у Аранђеловцу

Датум одбране:

Висока технолошка школа струковних студија (ВТШСС)

Аранђеловац

**Завршни рад: пројектни задатак**

**Име студента, број индекса**: Кристина Булатовић IT-02-11/2014

**Име ментора**: Мирослав Марковић

**Наслов завршног рада**: Информациони систем библиотеке

**Опис и циљеви рада**:

Први циљ је креирати Windows апликацију која приказује податаке о књигама и о члановима библиотеке, као и њихово додавање, измену и брисање у бази. Треба урадити вођење евиденције изнајмљивања књига, статистику књига које се највише изнајмљују и издавање потврда члановима. Циљ је да се поједноставе свакодневне обавезе вођења библиотеке на брз и ефикасан начин.

Други циљ је креирати Веб страницу која приказује податке о књигама у библиотеци и претрага која омогућава да се књиге претражују по удк броју, наслову, аутору, издавачу, категорији и језику. Циљ Веб странице је да студентима омогући брз увид у стање књига.

**Очекивани резултати**:

1. Приказати потребне податке о књигама, као и члановима библиотеке
2. Омогућити рад са базом библиотеке: додавање, измена и брисање жељених података
3. Креирање статистике књига које се највише изнајмљују
4. Издавање потврда члановима
5. Креирање веб странице са претрагом књига по удк броју, наслову, аутору, издавачу, категорији и језику

У Аранђеловцу, дана 05.09.2017. Ментор:

Садржај

[1. Увод 5](#_Toc490910043)

[1.1. C# програмски језик 5](#_Toc490910044)

[1.2. Microsoft Visual Studio развојно окружење 6](#_Toc490910045)

[1.3. Начин функционисања клијент-сервер сисема 6](#_Toc490910046)

[1.4. MySQL база података 7](#_Toc490910047)

[2. Windows апликација – ВТШ Библиотека 7](#_Toc490910048)

[2.1. Форма за регистрацију 8](#_Toc490910049)

[2.1.1. Функција LoginRead() 9](#_Toc490910050)

[2.2. Форма за промену лозинке 10](#_Toc490910051)

[2.2.1. Функција PasswordUpdate() 11](#_Toc490910052)

[2.3. Главна форма 12](#_Toc490910053)

[2.3.1. Претраживање 14](#_Toc490910054)

[2.4. Форма за задуживање књига 17](#_Toc490910055)

[2.4.1. Функција changeState() 20](#_Toc490910056)

[2.4.2. Функција InsertBorrowingBooks() 21](#_Toc490910057)

[2.5. Форма за раздуживање књига 22](#_Toc490910058)

[2.5.1. Функција DeleteBorrowStudent() 25](#_Toc490910059)

[2.6. Форма за евиденцију чланова 25](#_Toc490910060)

[2.6.1. Форма за издавање потврде 28](#_Toc490910061)

[2.7. Форма за евиденцију књига 28](#_Toc490910062)

[2.7.1. Форма за евиденцију аутора 31](#_Toc490910063)

[2.7.2. Форма за статистику 32](#_Toc490910064)

[2.7.3. Функција EmptyAllRows() 34](#_Toc490910065)

[3. Веб страница школске библиотеке ВШАР 35](#_Toc490910066)

[3.1. Функција filterTable() 37](#_Toc490910067)

[4. База података vts\_biblioteka 37](#_Toc490910068)

[5. Закључак 41](#_Toc490910069)

[Референцe 42](#_Toc490910070)

# Увод

Овај рад је подељен на два дела: израда Windows апликације и израда Веб странице. Windows апликација је рађена у C# програмском језику у развојном окружењу Visual Studio. Апликација омогућава брзо и једноставно манипулисање подацима из базе, koja je део ове апликације, њихов приказ, измену и брисање. Поред тога и могућност изнајмљивања књига члановима и потпуни увид у евиденцију изнајмљивња. Апликација садржи и статистику књига које су се највише пута изнајмљивале, као и издавање потврда студентима које чине доказ да не дугују ниједну књигу библиотеци. Веб страница која се састоји од низа хтмл докумената којима се може приступити уз помоћ веб-читача пружа једноставну и брзу претрагу књига. Клијент преко мреже шаље захтев серверу, сервер их обрађује и на исти начин враћа одговор клијенту. То значи да када кориник у поље за претрагу унесе неки текст, сервер му враћа резултате претраге која се врши по удк броју (удк је универзална децимална класификација), наслову, аутору, издавачу, категорији или језику и те податке исписује у табели. Windows апликација, као и Веб страна су повезане истом MySQL базом података.

## C# програмски језик

C# (Си шарп) спада међу млађе програмске језике с обзиром да је његова прва верзија премијерно приказана 2002. године као саставни део Мicrosoft развојног окружења. У унапређеним верзијама овог програмског језика додаване су бројне функције које корисницима омогућавају програмирање без велике употребе изворног кода.

Овакав начин писања програма знатно је олакшао посао програмерима који раде у C#. Поред тога, важно је истаћи да је овај програмски језик опште примене и намењен је креирању софтвера за .NET Framework платформу.

C# је настао као унапређена верзија програмских језика Јава, C, и C++. Највећа предност програмирања у C# у односу на језике из којих је проистекао је знатно лакша употреба. Све што је потребно је да се напише један програм у C# је обичан текстуални едитор попут Нотепада, као и инсталиран програм .NET Framework на вашем рачунару. Овај програм представља велику библиотеку алата коју можете користити уз помоћ преко 20 различитих програмских језика.

С обзиром да је .NET Framework прилагођен за бројне оперативне системе, писање кодова у С# омогућиће вам да креирате софтвере за Windows, Linux, iОS, ОSX, Аndroid и BSD окружење.

Такође, важно је напоменути и да С# спада међу case sensitive језике, који праве разлику између малих и великих слова. Осим тога, овај програмски језик припада и групи такозваних strongly typed, односно строго типизираних језика, који захтевају експлицитно идентификоване типове у коду.

## Microsoft Visual Studio развојно окружење

Microsoft Visual Studio представља револуцију у развоју апликативних решења и софтверских пројеката. Не постоји ни једно интегрисано развојно окружење за програмера толико удобно, интуитивно, унапређено, user-friendly, квалитетно и што је најважније стабилно. Са Microsoft Visual Studio-м можете да буквално испрограмирате све што видите у Windows-у, на интернету али такође можете програмирати и апликације за мобилне телефоне, таблете или правити игрице за Xbоx конзоле. Microsoft Visual Studio је програмерско окружење највише заступљено пре свега у свим школама, едукативним и другим институцијама, затим у највећем броју у малим, средњим и великим фирмама у Србији.

## Начин функционисања клијент-сервер сисема

Сервер процес започиње на неком рачунару (на којем је смештен), покреће се, а затим прелази у sleep modе и чека да га неки клијент процес контактира и затражи неку услугу од њега. Основна предност двослојне клијент-сервер архитектуре је та што базу података чини независном од апликација, јер интегритет базе података осигурава њену независност од апликација које над њом раде.

У области информационих технологија сервер је рачунарски систем који пружа услуге другим рачунарским системима – клијентима. Комуникација између сервера и клијента одвија се преко рачунарске мреже. Када се под појмом сервер подразумева рачунар, то се углавном односи на рачунар који обавља серверске послове.

Клијент је рачунарски систем који приступа сервису на другом рачунару (серверу) преко неке врсте телекомуникационе мреже или неког медијума друге врсте.

Клијент/сервер модел је базиран на дистрибуцији функција између два типа независних и аутономних процеса: сервера и клијента. Клијент је било који процес који захтева специфичне услуге од сервер процеса. Клијент-сервер модел обраде података је модел дистрибуиране обраде података, где се функције једног корисничког програма расподељују на најмање два процеса који међусобно комуницирају. Та два процеса обраде података су: клијентски и серверски.

## MySQL база података

SQL је језик који се користи за писање упита унутар МySQL базе података. Првобитан власник МySQL-а је била шведска компанија МySQL АB, док је данашњи власник Оracle. Прва верзија МySQL базе података изашла је 1995. године. МySQL база података је најпопуларнија база међу wеб апликацијама и користи LAMP платформу. Апликације које користе МySQL базу података су TYPO3, MODx, Joomla, Wordpress, phpBB, MyBB, Drupal као и многи други софтвери. МySQL такође користе и најпознатији wеб сајтови као што су  Facebook, Twitter, Flickr, Youtube i Google(не укључујући претрагу).

МySQL омогућава приступ бази података уз помоћ већине програмских језика. МySQL сервер и подржане бибилиотеке писане су у C и C++ програмским језицима. МySQL ради на многим системским платформама као што су AIX, BSDi, FreeBSD, HP-UX, eComStation, i5/OS, IRIX, Linux, OS X, Microsoft Windows, NetBSD, Novell NetWare, OpenBSD, OpenSolaris, OS/2 Wrap, QNX, Oracle Solaris, Symbian, SunOS, SCO OpenServer, SCO UnixWare, Sanos i Tru64. Портови МySQL-а према OpenVMS такође постоје.

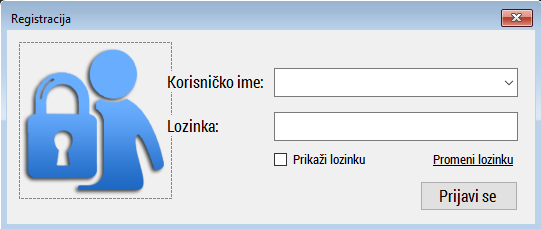
# Windows апликација – ВТШ Библиотека

Информациони систем за пословање библиотеке подржава све функције које су потребне у свакодневном раду библиотекара. Програм покрива све везанo за обраду књига, чланова, аутора, издавача, категорије и језика.

У раду се нећемо обазирати на споредне делове кода већ само на најбитније делове (само битне функције). У овом раду сви кодови су писани у програмском језику C#.

## Форма за регистрацију

Када се покрене апликација отвара се прозор за регистрацију. Он захтева унос корисничког имена и лозинке, и изгледа као на Слици 2.1.



*Слика 2.1. Форма за регистрацију*

Најбитнија функција у овој форми је btnLogin\_Click која се покреће притиском на дугме „Prijavi se“.

1. **private** **void** btnLogin\_Click(**object** sender, EventArgs e)
2. {
3. **try**
4. {
5. sql.Connection();
6. **bool** t = sql.LoginRead(comboUsername.Text, txtPassword.Text, "prijavljivanje");
7. **if** (t == **true**)
8. {
9. **this**.Hide();
10. library.ShowDialog();
11. }
12. **else** MessageBox.Show("Unesite ispravno korisničko ime i lozinku!", "Greška!", MessageBoxButtons.OK, MessageBoxIcon.Error);
13. sql.Close();
14. }
15. **catch**
16. {
17. MessageBox.Show("Greška pri konekciji! Proverite internet!", "Informacija", MessageBoxButtons.OK, MessageBoxIcon.Information);
18. }
19. }

У 3. линији кода дефинисана је try-catch конструкција која је специјално намењена за хватање, спречавање и исправљање грешака до којих може доћи уколико није остварена конекција са интернетом. У случају такве грешке исписаће се порука „Greška pri konekciji! Proverite internet!“.

У конструкторској методи дефинисан је објекат sql класе MySQL и у 5. линији кода позива се његова функција Connection, која омогућава повезивање са базом података. После повезивања у 6. линији кода позива се функција LoginRead (Више речи о њој у секцији 2.1.1.), којој се прослеђују 3 параметра. Први параметар је корисничко име, други лозинка и трећи назив табеле у бази која је намењена регистрацији (О бази више речи у секцији 4.). Променљивој t додељује се вредност коју враћа функција LoginRead, та вредност је bool и може бити true или false (тачно или нетачно). Ако је вредност true форма за регистрацију више неће бити видљива уз помоћ функције Hide и покренуће се нова форма која се зове Library (више детаља у секцији 2.3.) уз помоћ функције ShowDialog. Aко је вредност false отвара се алерт са поруком „Unesite ispravno korisničko ime i lozinku!“. У 13. линији кода затвара се база уз помоћ функције Close.

### Функција LoginRead()

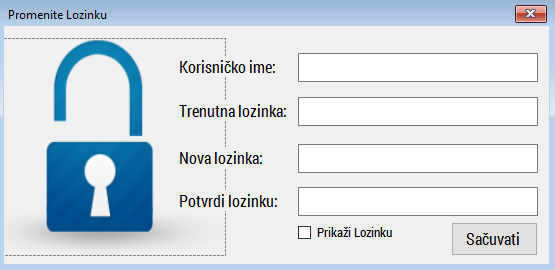
Ова функција се налази у класи MySQL. Та функција изгледа овако:

1. **public** **bool** LoginRead(**string** username, **string** password, **string** table)
2. {
3. **bool** result = **false**;
4. cmd.CommandText = "select \* from " + table;
5. conn.Open();
6. MySqlDataReader reader = cmd.ExecuteReader();
7. **if** (table == "prijavljivanje")
8. {
9. **while** (reader.Read())
10. {
11. **if** (username == reader[1].ToString() && password == reader[2].ToString())
12. {
13. result = **true**;
14. }
15. }
16. }
17. **return** result;
18. }

Већ је споменуто да функција прихвата 3 параметра, корисничко име, лозинку и назив табеле. У 3. линији кода дефинисана је променљива result типа bool и додељена јој је вредност false. У 4. линији кода прави се упит који се шаље бази. „select \* from " + table " значи да се селектује све што се налази у тој табели. У првом if-у проверава се да ли је вредност табеле која је прослеђена преко параметра једнака вредности "prijavljivanje", ако јесте улази се у while петљу. While петља пролази кроз све вредности које је упит вратио и у другом if-у проверава се да ли су те вредности једнаке парамерима корисничко име и лозинка који су прослеђени. Ако су обе вредности једнаке, променљивој result која је на почетку дефинисана и има вредност false, мења се вредност у true. У 13. линији кода наредбом return излази се из функције и враћа се вредност променљиве која је true или false.

## Форма за промену лозинке

Овој форми се приступа из форме за регистрацију кликом на линк „Promeni lozinku“. Као што сам назив каже у овој форми корисник може променити своју лозинку. Потребно је унети корисничко име, тренутну лозинку, нову лозинку и потврду нове лозинке, та форма изгледа као на Слици 2.2.



*Слика 2.2. Форма за промену лозинке*

Најбитнија функција у овој форми је btnSave\_Click која се покреће притиском на дугме „Sačuvati“.

1. **private** **void** btnSave\_Click(**object** sender, EventArgs e)
2. {
3. **try**
4. {
5. sql.Connection();
6. **bool** t = sql.LoginRead(txtUsername.Text, txtCurrentPassword.Text, "prijavljivanje");
7. sql.Close();
8. **if** (t == **true**)
9. {
10. **if** (txtNewPassword.Text == txtConfirmPassword.Text)
11. {
12. sql.Connection();
13. **bool** updatePass = sql.PasswordUpdate(txtUsername.Text, txtNewPassword.Text, "prijavljivanje");
14. sql.Close();
15. **if** (updatePass)
16. {
17. MessageBox.Show("Uspešno ste promenili lozinku!!!", "Informacija", MessageBoxButtons.OK, MessageBoxIcon.Information);
18. }
19. **else** MessageBox.Show("Greška pri menjanju lozinke!", "Greška!", MessageBoxButtons.OK, MessageBoxIcon.Error);
20. txtUsername.Text = "";
21. txtCurrentPassword.Text = "";
22. txtNewPassword.Text = "";
23. txtConfirmPassword.Text = "";
24. Close();
25. }
26. **else** MessageBox.Show("Greška pri potvrdi lozinke!", "Greška!", MessageBoxButtons.OK, MessageBoxIcon.Error);
27. }
28. **else** MessageBox.Show("Uneli se pogrešnu lozinku!!!", "Greška!", MessageBoxButtons.OK, MessageBoxIcon.Error);
29. sql.Close();
30. }
31. **catch**
32. {
33. MessageBox.Show("Greška pri konekciji! Proverite internet!", "Informacija", MessageBoxButtons.OK, MessageBoxIcon.Information);
34. }
35. }

Функција има try-catch конструкцију и у случају да дође до грешке при конекцији са интернетом исписаће поруку „Greška pri konekciji! Proverite internet!“. У 5. линији кода остварена је конекција са базом, а у 6. линији кода позвана је функција LoginRead из класе MySQL. Ова функција је објашњена у секцији 2.1.1. Прослеђују јој се корисничко име, лозинка, као и назив табеле. Функција проверава да ли корисник постоји у бази и ако постоји враћа вредност true, ако не постоји враћа false. Само ако је функција вратила true проверава се да ли су поље „Nova lozinka” и поље “Potvrdi lozinku” једнаки.

Ако је и тај услов испуњен наставља се даље, пошто је након LoginRead-а у 7. линији кода затворена конекција поново се конекција отвара у 12. линији кода и у 13. линији кода се позива нова функција PasswordUpdate из класе MySql. Ова функција враћа bool вредност и прихвата 3 параметра, корисничко име, нову лозинку и назив табеле (Више речи о њој у секцији 2.2.1.). Ако је функција вратила вредност true значи да је лозинка успешно промењена и исписује се порука „Uspešno ste promenili lozinku!!!“.

Ако функција врати вредност false лозинка није успешно промењена и исписује се порука „Greška pri promeni lozinke!“. На крају се вредности из поља бришу тако што су им додељени празни стрингови и у 24. линији кода уз помоћ функције Close затвара се форма.

### Функција PasswordUpdate()

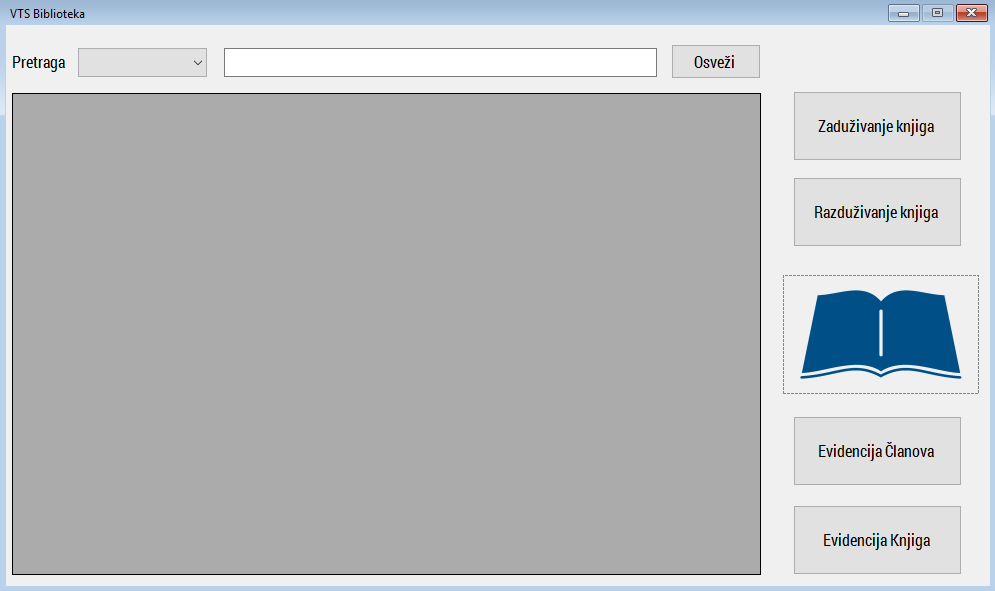
Ова функција је из класе MySQL.

1. **public** **bool** PasswordUpdate(**string** username, **string** newPassword, **string** table)
2. {
3. **try**
4. {
5. cmd.CommandText = "update " + table + " set Lozinka = '" + newPassword + "' where KorisnickoIme = '" + username + "'";
6. conn.Open();
7. MySqlDataReader reader = cmd.ExecuteReader();
8. **return** **true**;
9. }
10. **catch**
11. {
12. **return** **false**;
13. }
14. }

Као што је већ речено функција прихвата 3 параметра, корисничко име, нову лозинку и назив табеле која је везана за регистрацију корисника. Функција има try-catch конструкцију и у случају било какве грешке при раду са базом података функција ће вратити вредност false. У 5. линији кода се налази упит "update " + table + " set Lozinka = '" + newPassword + "' where KorisnickoIme = '" + username + "'" што значи да се мења табела чији је параметар прослеђен и на место Lozinka додељује се нова вредност која је прослеђена преко параметра „newPassword“ и да то уради на месу где је KorisnickoIme једнако парамеру „username“ који је прослеђен. Корисничко име је јединствено, што значи да се не може поновити два пута. Након писања упита у 6. линији кода отвара се конекција, у 7. линији кода упит се извршава, након чега у случају да није дошло ни до какве грешке током извршавања упита функција враћа вредност true.

## Главна форма

Када се корисник успешно региструје отвара се главни део апликације. У овој форми виде се подаци о свим књигама које се налазе у библиотеци, удк број, наслов, аутор, издавач, категорија, језик и стање. Ту се налази и претрага књига (Више о претрази у секцији 2.3.1.). Са десне стране налазе се 4 дугмета која кликом на њих отварају нове форме. Прво дугме отвара форму везану за задуживање књига, друго за раздуживање, треће отвара форму у којој се води евиденција чланова и четврто, последње дугме отвара форму у којој се води евиденција књига. Главна форма изгледа као на Слици 2.3.



*Слика 2.3. Главна форма*

Ова форма има доста функција, али две битније функције су SearchBooks и Library\_Load. SearchBooks функција не враћа никакву вредност и прихвата један параметар. Код те две функције изгледа овако:

1. **public** **void** SearchBooks(**string** query)
2. {
3. **string** connString = "Server=localhost;Port=3306;Database=vts\_biblioteka;Uid=root;password=''";
4. MySqlConnection conn = **new** MySqlConnection(connString);
5. MySqlCommand cmd = conn.CreateCommand();
6. conn.Open();
7. cmd.CommandText = query;
8. MySqlDataReader reader = cmd.ExecuteReader();
9. conn.Close();
10. DataTable dt = **new** DataTable();
11. MySqlDataAdapter da = **new** MySqlDataAdapter(cmd);
12. da.Fill(dt);
13. dataGridView1.DataSource = dt;
14. }
16. **private** **void** Library\_Load(**object** sender, EventArgs e)
17. {
18. **string** query = "select knjige.UDKBroj, knjige.Naziv , autori.ImePrezime as Autor, izdavaci.Naziv as Izdavač, kategorije.Naziv as Kategorija, jezik.Naziv as Jezik, knjige.Stanje from knjige inner join autori on knjige.SifraAutora = autori.Sifra inner join izdavaci on knjige.SifraIzdavaca = izdavaci.Sifra inner join kategorije on knjige.SifraKategorije = kategorije.Sifra inner join jezik on knjige.SifraJezika = jezik.Sifra ORDER BY knjige.Naziv asc";
19. SearchBooks(query);
20. }

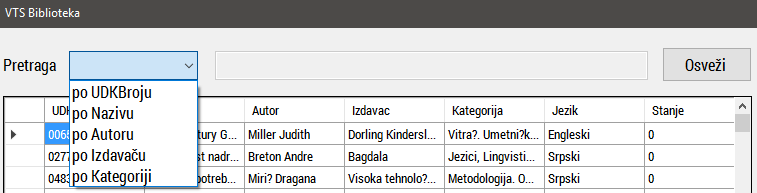
Креирана је променљива типа string и додељена јој је вредност која је потребна за конекцију са базом. Објекту класе MySqlConnection прослеђена је променљива connString која је дефинисана у констукторској методи и позивом функције Open у 6. линији кода остварена је конекција са базом података. Преко cmd-а позвана је функција CommandText којој је додељена вредност параметара који je прослеђен преко функције, та вредност је упит који се у 8. линији кода извршава. Након извршавања упита затвара се конекција функцијом Close и праве се два нова објекта. Први објекат је dt из класе DataTable, а други da из класе MySqlDataAdapter. Ови објекти су битни за приказ података из базе у dataGridView-у, то је део у форми где се може видети подаци из базе, у овом случају подаци о књигама. У 12. линији кода позива се функција објекта da која се назива Fill и прослеђује јој се објекат dt. У 13. линији кода dataGridView-у се додељује вредност dt. У dt-у се налазе вредности прослеђене након извршавања упита, тако да се након овог дела кода на екрану исписују жељене вредности из базе.

Испод ове функције у 16. линији кода налази се функција Library\_Load која се покреће чим се форма отвори. У функцији се налазе две линије кода, упит и позив функције SearchBooks којој се прослеђује тај упит. 18. линија кода представља упит и он изгледа овако:

"select knjige.UDKBroj, knjige.Naziv , autori.ImePrezime as Autor, izdavaci.Naziv as Izdavač, kategorije.Naziv as Kategorija, jezik.Naziv as Jezik, knjige.Stanje from knjige inner join autori on knjige.SifraAutora = autori.Sifra inner join izdavaci on knjige.SifraIzdavaca = izdavaci.Sifra inner join kategorije on knjige.SifraKategorije = kategorije.Sifra inner join jezik on knjige.SifraJezika = jezik.Sifra ORDER BY knjige.Naziv asc". То значи да се селектује удк број и наслов књиге, назив аутора, назив издавача, назив категорије, назив језика и стање књига, пошто се у бази налазе више табела, табела за књиге, табела за аутора, издавача, категорију и језик користи се inner join како би се спојиле све те табеле. И на крају ORDER BY је кључна реч која се користи за сортирање резултата упита по некој колони, у овом случају по називу и asc значи да ће се резултати сортирати по растућем редоследу.

### Претраживање

У целој апликацији претраживање функционише на веома сличан начин, у овој секцији биће представљено како функционише у главној форми. Прво је битно да се изабере по чему ће се претрага вршити, у главној форми претрагу можемо вршити по: удк броју, наслову, аутору, издавачу и категорији. Након тога уноси се појам који корисник жели да пронађе. За унос упита равномерно се користе мала и велика слова. Потпуно су еквивалентни упити: „Kovačević“ и „kovačević“.



*Слика 2.4. Сортирање претраге*

Функција која је задужена за претрагу је txtSearch\_TextChanged и она се покреће након сваке измене у пољу за унос текста који се претражује.

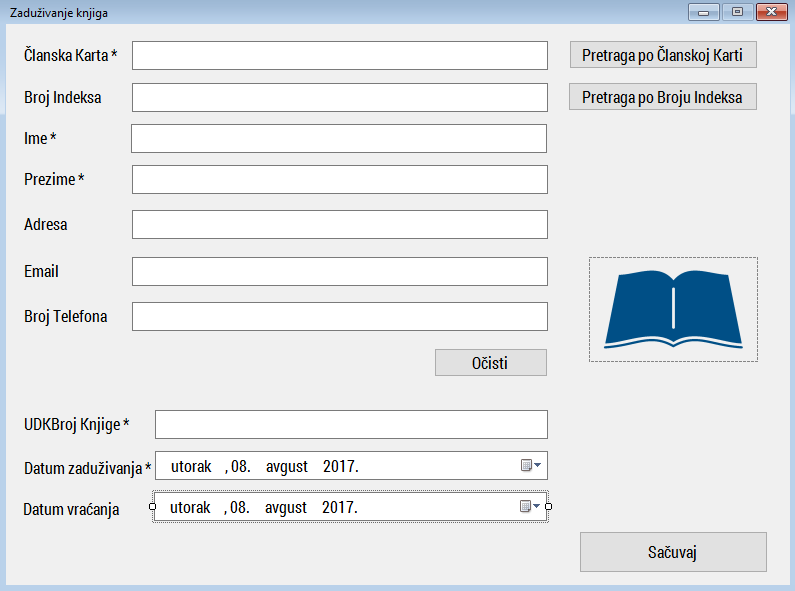
1. **private** **void** txtSearch\_TextChanged(**object** sender, EventArgs e)
2. {
3. **if** (!**string**.IsNullOrEmpty(txtSearch.Text))
4. {
5. dataGridView1.DataSource = **null**;
6. **string** search\_books = txtSearch.Text;
8. **if** (comboSearch.Text == "po UDKBroju")
9. {
10. **string** query = "select knjige.UDKBroj, knjige.Naziv , autori.ImePrezime as Autor, izdavaci.Naziv as Izdavač, kategorije.Naziv as Kategorija, jezik.Naziv as Jezik, knjige.Stanje from knjige inner join autori on knjige.SifraAutora = autori.Sifra inner join izdavaci on knjige.SifraIzdavaca = izdavaci.Sifra inner join kategorije on knjige.SifraKategorije = kategorije.Sifra inner join jezik on knjige.SifraJezika = jezik.Sifra where LOWER(knjige.UDKBroj) LIKE '%" + search\_books.ToLower() + "%'";
11. SearchBooks(query);
12. }
13. **else** **if** (comboSearch.Text == "po Nazivu")
14. {
15. **string** query = "select knjige.UDKBroj, knjige.Naziv , autori.ImePrezime as Autor, izdavaci.Naziv as Izdavač, kategorije.Naziv as Kategorija, jezik.Naziv as Jezik, knjige.Stanje from knjige inner join autori on knjige.SifraAutora = autori.Sifra inner join izdavaci on knjige.SifraIzdavaca = izdavaci.Sifra inner join kategorije on knjige.SifraKategorije = kategorije.Sifra inner join jezik on knjige.SifraJezika = jezik.Sifra where LOWER(knjige.Naziv) LIKE '%" + search\_books.ToLower() + "%'";
16. SearchBooks(query);
17. }
18. **else** **if** (comboSearch.Text == "po Autoru")
19. {
20. **string** query = "select knjige.UDKBroj, knjige.Naziv , autori.ImePrezime as Autor, izdavaci.Naziv as Izdavač, kategorije.Naziv as Kategorija, jezik.Naziv as Jezik, knjige.Stanje from knjige inner join autori on knjige.SifraAutora = autori.Sifra inner join izdavaci on knjige.SifraIzdavaca = izdavaci.Sifra inner join kategorije on knjige.SifraKategorije = kategorije.Sifra inner join jezik on knjige.SifraJezika = jezik.Sifra where LOWER(autori.ImePrezime) LIKE '%" + search\_books.ToLower() + "%'";
21. SearchBooks(query);
22. }
23. **else** **if** (comboSearch.Text == "po Izdavaču")
24. {
25. **string** query = "select knjige.UDKBroj, knjige.Naziv , autori.ImePrezime as Autor, izdavaci.Naziv as Izdavač, kategorije.Naziv as Kategorija, jezik.Naziv as Jezik, knjige.Stanje from knjige inner join autori on knjige.SifraAutora = autori.Sifra inner join izdavaci on knjige.SifraIzdavaca = izdavaci.Sifra inner join kategorije on knjige.SifraKategorije = kategorije.Sifra inner join jezik on knjige.SifraJezika = jezik.Sifra where LOWER(izdavaci.Naziv) LIKE '%" + search\_books.ToLower() + "%'";
26. SearchBooks(query);
27. }
28. **else** **if** (comboSearch.Text == "po Kategoriji")
29. {
30. **string** query = "select knjige.UDKBroj, knjige.Naziv , autori.ImePrezime as Autor, izdavaci.Naziv as Izdavač, kategorije.Naziv as Kategorija, jezik.Naziv as Jezik, knjige.Stanje from knjige inner join autori on knjige.SifraAutora = autori.Sifra inner join izdavaci on knjige.SifraIzdavaca = izdavaci.Sifra inner join kategorije on knjige.SifraKategorije = kategorije.Sifra inner join jezik on knjige.SifraJezika = jezik.Sifra where LOWER(kategorije.Naziv) LIKE '%" + search\_books.ToLower() + "%'";
31. SearchBooks(query);
32. }
33. }
34. **else**
35. {
36. dataGridView1.DataSource = **null**;
37. **string** query = "select knjige.UDKBroj, knjige.Naziv , autori.ImePrezime as Autor, izdavaci.Naziv as Izdavač, kategorije.Naziv as Kategorija, jezik.Naziv as Jezik, knjige.Stanje from knjige inner join autori on knjige.SifraAutora = autori.Sifra inner join izdavaci on knjige.SifraIzdavaca = izdavaci.Sifra inner join kategorije on knjige.SifraKategorije = kategorije.Sifra inner join jezik on knjige.SifraJezika = jezik.Sifra ORDER BY knjige.Naziv asc";
38. SearchBooks(query);
39. }
40. }

На почетку функције у 3. линији кода се проверава да ли је вреднст поља за унос текста који треба да се претражи празна, ако јесте празна улазимо у else где вредност dataGridView-a постављамо на null, прави се упит и позива се већ објашњена функција SearchBooks. Ако вредност поља за унос није празна опет dataGridView постављамо на null, затим у 6. линији кода креира се променљива којој је додељена вредност поља за унос. Након тога у 8. линији кода проверава се поље у којем је изабрано по чему ће се вршити претрага, када је то поље једнако вредности из if-a креира се упит и позива се функција SearchBooks којој се прослеђује тај упит. У сваком if-у сви упити су врло слични. Упит из првог if-a, када хоћемо да вршимо претрагу по удк броју изгледа овако: „select knjige.UDKBroj, knjige.Naziv , autori.ImePrezime as Autor, izdavaci.Naziv as Izdavač, kategorije.Naziv as Kategorija, jezik.Naziv as Jezik, knjige.Stanje from knjige inner join autori on knjige.SifraAutora = autori.Sifra inner join izdavaci on knjige.SifraIzdavaca = izdavaci.Sifra inner join kategorije on knjige.SifraKategorije = kategorije.Sifra inner join jezik on knjige.SifraJezika = jezik.Sifra where LOWER(knjige.UDKBroj) LIKE '%" + search\_books.ToLower() + "%'“ . У овом упиту врши се селекција по удк броју и наслову књиге, називу аутора, називу издавача, називу категорије, називу језика и стању књига. Inner join је већ објашњен да служи за спајање табела, LIKE и % пре и после променљиве која се претражује значи да није битно шта има пре и после текста који се претражује, на пример корисник унесе „čev“ он ће пронаћи „Kovačević“. Функција LOWER служи да преузете вредности из базе претвори у мала слова као и функција ToLower која вредност променљиве такође претвара у мала слова. И то омогућава равномерно коришћење малих и великих слова.

## Форма за задуживање књига

Ова форма се отвара из главне форме и служи да члановима библиотеке омогући да на брз и једноставан начин изнајме жељену књигу. Потребно је да обавезна поља буду попуњена (поља са звездицом), а то су чланска карта, име, презиме, удк број књиге и датум задуживања. Може се вршити и претрага по чланској карти или броју индекса тако што се унесе једна од те две вредности и након тога се притисне дугме које се налази поред поља у коме је текст унет. Уз помоћ претраге проверава се да ли члан постоји и у случају да постоји остала поља као сто су име, презиме, адреса, емаил и број телефона се сама попуне подацима од тог члана. Након тога је потребно унети само удк број књиге, датум издавања и датум враћања (који није обавезан). Притиском на дугме „Sačuvaj“ успешно изнајмљујемо књигу.

У случају да члан не постоји потребно је попунити сва поља и притиском на дугме „Sačuvaj“ прво се сачувава члан након чега се успешно сачувава и изнајмљивање изабране књиге. Форма за изнајмљивање књига игледа као на Слици 2.5.



*Слика 2.5. Форма за задуживање књига*

Ова форма садржи много функција, али најбитнија функција је btnSave\_Click која се покреће кликом на дугме „Sačuvaj”.

1. **private** **void** btnSave\_Click(**object** sender, EventArgs e)
2. {
3. sql.Connection();
4. **bool** membersRead = sql.MembersRead(txtMembershipCard.Text, "clanovi");
5. sql.Close();
6. **if** (membersRead)
7. {
8. **if** (changeState(txtUdk.Text, "knjige"))
9. {
10. sql.Connection();
11. **bool** result = sql.InsertBorrowingBooks(txtMembershipCard.Text, txtUdk.Text, dateBorrowing.Text, dateReturn.Text, "zaduzivanje\_knjiga");
12. sql.Close();
13. **if** (result)
14. {
15. sql.Statistic(txtUdk.Text, "statistika");
16. MessageBox.Show("Uspešno ste zadužili knjigu!", "Informacija", MessageBoxButtons.OK, MessageBoxIcon.Information);
17. Members("", "", "", "", "", "", "");
18. Close();
19. }
20. **else** MessageBox.Show("Greška pri zaduživanju!", "Greška!", MessageBoxButtons.OK, MessageBoxIcon.Error);
21. }
22. **else** MessageBox.Show("Knjiga se trenutno ne nalazi na stanju!", "Greška!", MessageBoxButtons.OK, MessageBoxIcon.Error);
23. }

У 3. линији кода се отвара конекција, затим се у 4. позива функција MembersRead. Ова функција прихвата два параметра, први параметар је чланска карта, а други назив табеле у којој се налазе чланови библиотеке. Функција проверава да ли члан постоји и ако постоји враћа вредност true, у супротном враћа false. Након тога затвара се конекција и у if-у се проерава да ли је враћена вредност true или false. Ако је вредност true улази се у следећи if у ком се позива функцију changeState. Ова функција прихвата 2 параметра удк број књиге и назив табеле у бази. Функција такође враћа bool вредност која може бити true или false (Више речи о овој функцији у секцији 2.4.1.). Ако је враћена вредност true наставља се даље са извршавањем кода. У 10. линији отвара се конекција, затим у 11. линији кода позива се функција InsertBorrowingBooks из класе MySQL. Ова функција омогућава унос података везаних за изнајмљивање књига у базу података, функција прихвата 5 параметара. Ти параметри су: чланска карта, удк број књиге, датум задуживања, датум враћања и назив табеле (Више детаља о функцији у секцији 2.4.2.). Ако није дошло ни до какве грешке функција ће вратити вредност true, у супротном враћа false. Та вредност се проверава и ако је true позива се функција Statistic (Више речи у секцији 2.7.2.), у 16. линији кода исписује се порука „Uspešno ste zadužili knjigu!“. А у 17. линији кода позива се функција Members која сва поља у форми враћа на почетно стање и на крају у 18. линији кода функцијом Close затвара се форма.

Већ смо рекли да се ово извршава ако функција MembersRead врати вредност true, а у случају да функција врати вредност false извршиће се следећи код:

1. **else**
2. {
3. sql.Connection();
4. **bool** result = sql.InsertMembers(txtMembershipCard.Text, txtIndexNumber.Text, txtFirstName.Text, txtLastName.Text, txtAddress.Text, txtEmail.Text, txtPhoneNumber.Text, "clanovi");
5. sql.Close();
6. **if** (result)
7. {
8. **if** (changeState(txtUdk.Text, "knjige"))
9. {
10. sql.Connection();
11. **bool** r = sql.InsertBorrowingBooks(txtMembershipCard.Text, txtUdk.Text, dateBorrowing.Text, dateReturn.Text, "zaduzivanje\_knjiga");
12. sql.Close();
13. **if** (r)
14. {
15. changeState(txtUdk.Text, "knjige");
16. MessageBox.Show("Uspešno ste zadužili knjigu!", "Informacija", MessageBoxButtons.OK, MessageBoxIcon.Information);
17. Members("", "", "", "", "", "", "");
18. Close();
19. }
20. **else** MessageBox.Show("Greška pri zaduživanju!", "Greška!", MessageBoxButtons.OK, MessageBoxIcon.Error);
22. }
23. **else** MessageBox.Show("Knjiga se trenutno ne nalazi na stanju!", "Greška!", MessageBoxButtons.OK, MessageBoxIcon.Error);
24. }
25. **else** MessageBox.Show("Greška pri unosu člana", "Greška!", MessageBoxButtons.OK, MessageBoxIcon.Error);
26. }
27. }

Отвара се конекција и позива се функција InsertMembers. Пошто члан не постоји, ова функција омогућава да се дода у базу. Ако је члан успешно додат функција враћа вредност true, у супротном враћа false. Након додавања члана затвара се конекција са базом и у if-у се проверава да ли је вредност коју је функција вратила true, ако јесте улази се у следећи if који је претходно већ објашњен. Остатак кода је исти као када је функција MembersRead вратила вредност true.

### Функција changeState()

Уз помоћ ове функције смањује се стање књига у библиотеци када их члан изнајми, да би остали чланови могли увек да имају увид у стварно стање тренутних књига које су им на располагању. Функција прихвата 2 параметра удк број књиге и назив табеле, и враћа bool вредност. Ако не дође ни до какве грешке током извршавања кода у овој функцији она ће вратити вредност true, у супротном враћа вредност false.

1. **public** **bool** changeState(**string** udk, **string** table)
2. {
3. **try**
4. {
5. sql.Connection();
6. **string** state = sql.ReadState(udk, table);
7. sql.Close();
8. **if** (state != "")
9. {
10. **int** s = Convert.ToInt32(state);
11. s--;
12. **if** (s >= 0)
13. {
14. **string** newState = s.ToString();
15. sql.Connection();
16. sql.UpdateBookState(udk, newState, table);
17. sql.Close();
18. **return** **true**;
19. }
20. **else**
21. {
22. **return** **false**;
23. }
24. }
25. **else** **return** **false**;
26. }
27. **catch**
28. {
29. **return** **false**;
30. }
31. }

Функција има try-catch конструкцију, а у овом случају то значи да ако дође до био какве грешке током извршавања кода функција ће вратити вредност false. Након дефинисане try-catch конструкције отвара се конекција и позива се ReadState функција. Ова функција прихвата 2 параметра, удк број књиге и назив табеле. Функција враћа тренутно стање књиге чији удк број је послат. Затвара се конекција и проверава да ли је стање различито од празног стинга. Ако јесте то стање се конвертује у број уз помоћ функције Convert.ToInt32. Када је пребачено стање из стринга у број смањује се за 1 и у if-у се проверава да ли је то стање веће или једнако 0. Ако јесте у 14. линији кода стање се поново конвертује само сада из броја у стринг уз помоћ функције ToString. Отвара се конекција и позива се функција UpdateBookState која замењује тренутно стање књига са стањем које је прослеђено кроз параметар функције. Функција прихвата 3 параметра, удк број књиге, ново стање и назив табеле. Након тога затвара се конекција и функција враћа вредност true. Ако у било ком делу кода неки услов не буде испуњен и оде се у else функција ће вратити вредност false.

### Функција InsertBorrowingBooks()

InsertBorrowingBooks је функција уз помоћ које се у базу убацују подаци о изнајмљеним књигама. Функција те податке прима као параметре, и то су чланска карта, удк број књиге, датум издавања књиге, датум враћања књиге и назив табеле у бази која је одговорна за изнајмљивање књига. Функција враћа bool вредност и изгледа овако:

1. **public** **bool** InsertBorrowingBooks(**string** membershipCard, **string** udk, **string** dateBorrowing, **string** dateReturn, **string** table)
2. {
3. **try**
4. {
5. cmd.CommandText = "insert into " + table + " (ClanskaKarta, UDKBroj, DatumZaduzivanja, DatumVracanja) values ('" + membershipCard + "', '" + udk + "' , '" + dateBorrowing + "', '" + dateReturn + "')";
6. conn.Open();
7. MySqlDataReader reader = cmd.ExecuteReader();
8. **return** **true**;
9. }
10. **catch**
11. {
12. **return** **false**;
13. }
14. }

Има try-catch конструкцију, тако да у случају да дође до неке грешке са убацивањем података у базу функција ће вратити вредност false. После дефинисане try-catch конструкције у 5. линији кода написан је упит, отвора се конекција са базом и у 7. линији кода тај упит се извршава. Упит изгледа овако: „insert into " + table + " (ClanskaKarta, UDKBroj, DatumZaduzivanja, DatumVracanja) values ('" + membershipCard + "', '" + udk + "' , '" + dateBorrowing + "', '" + dateReturn + "')“, и то значи да се у табелу, чији назив је прслеђен преко параметра функције, додају вредности које се налазе у загради. Те вредности су: ClanskaKarta, UDKBroj, DatumZaduzivanja, DatumVracanja, и њима се додељују вредности прослеђене преко параметра функције. Ако је изнајмљивање књига успешно евидентирано у бази функција ће вратити вредност true.

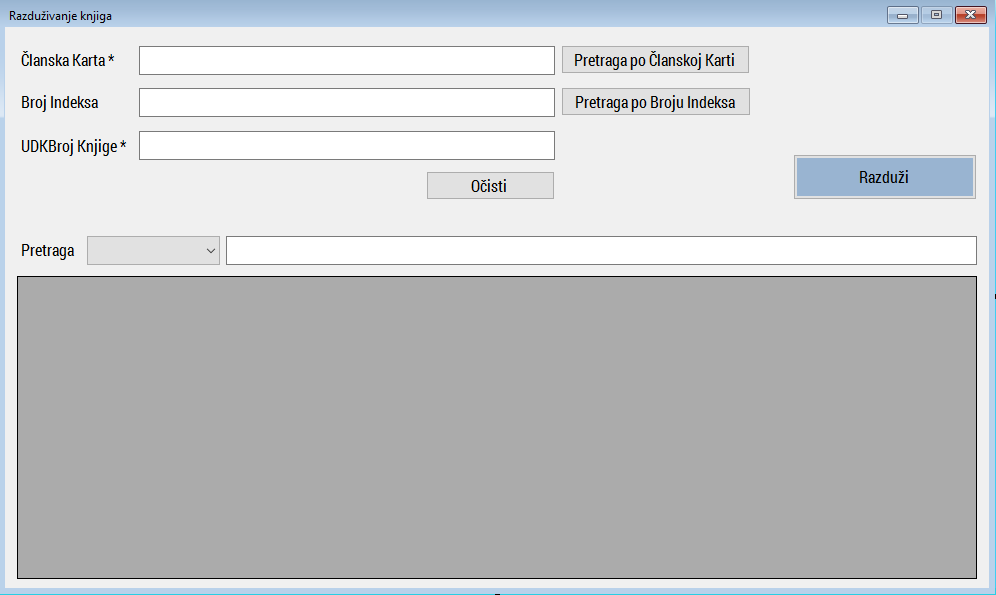
## Форма за раздуживање књига

Ова форма се отвара из главне форме, она се користи када чланови желе да врате изнајмљене књиге. Форма функционише на веома једноставан начин, што ће библиотекару уштедети доста времена. Садржи 3 поља од којих су 2 обавезна, а то су чланска карта и удк број књиге.

Члан се може пронаћи по чланској карти или по броју индекса ако је студент, када је члан пронађен потребно је унети удк број књиге коју је изнајмио и кликом на дугме „Razduži“ је цео процес обављен.

Дугме „Pretraga po Članskoj Karti“ и дугме „Pretraga po Broju Indeksa“ омогућавају да сe пронађе члан, такође ова форма има и претрагу свих изнајмљених књига са подацима код кога се та књига налази, датумом изнајмљивања и датумом до када књига треба да се врати.

Претрага није објашњена јер функционише на исти начин као и у главној форми, само у овој форми сe може претраживати по чланској карти, броју индекса, имену или презимену члана, удк броју књиге и наслову књиге.



*Слика 2.6. Форма за раздуживање књига*

Форма има доста функција од којих је најбитнија btnRediscover\_Click функција која се покреће кликом на дугме „Razduži“. Код функције изгледа овако:

1. **private** **void** btnRediscover\_Click(**object** sender, EventArgs e)
2. {
3. sql.Connection();
4. **string** id = sql.IDMembersRead(txtMembershipCard.Text, txtUDK.Text, "zaduzivanje\_knjiga");
5. sql.Close();
6. **if** (id != "")
7. {
8. sql.Connection();
9. **bool** result = sql.DeleteBorrowStudent(id, "zaduzivanje\_knjiga");
10. sql.Close();
11. **if** (result)
12. {
13. changeState(txtUDK.Text, "knjige");
14. txtUDK.Text = "";
15. dataGridView1.DataSource = **null**;
16. **string** search\_books = txtMembershipCard.Text;
17. **string** query = "select clanovi.ClanskaKarta, clanovi.BrojIndeksa, clanovi.Ime , clanovi.Prezime, clanovi.Email, clanovi.Telefon, knjige.UDKBroj, knjige.Naziv, autori.ImePrezime as Autor, izdavaci.Naziv as Izdavač, zk.DatumZaduzivanja, zk.DatumVracanja from clanovi inner join zaduzivanje\_knjiga as zk on zk.ClanskaKarta = clanovi.ClanskaKarta inner join knjige on zk.UDKBroj = knjige.UDKBroj inner join autori on knjige.SifraAutora = autori.Sifra inner join izdavaci on knjige.SifraIzdavaca = izdavaci.Sifra where clanovi.ClanskaKarta LIKE '%" + search\_books + "%'";
18. SearchBooks(query);
19. MessageBox.Show("Hvala Vam na vraćenoj knjizi!", "Informacija", MessageBoxButtons.OK, MessageBoxIcon.Information);
20. }
21. **else** MessageBox.Show("Greška pri razduživanju!", "Greška!", MessageBoxButtons.OK, MessageBoxIcon.Error);
22. }
23. **else** MessageBox.Show("Greška!", "Greška!", MessageBoxButtons.OK, MessageBoxIcon.Error);
24. }

Повезујемо се са базом, након чега се креира променљива типа стринг са називом id којој се додељује вредност коју враћа функција IDMembersRead. Позива се функција IDMembersRead из класе MySQL која прима 3 параметра, чланску карту, удк број књиге и назив табеле која је везана са изнајмљеним књигама. Функција проверава да ли се у бази налази члан, чија је чланска карта прослеђена као параметар функције, који је изнајмио књигу чији је удк број такође прослеђен као параметар функције. Ако тако изнајмљивање постоји функција враћа ид тог изнајмљивања, а ако не постоји враћа празан стринг. У 5. линији кода затвара се конекција са базом, у 6. линији кода у if-у проверава се да ли је id различит од празног стинга. Ако изнајмљивање постоји и ако је id различит од празног стринга улази се у if. У 8. линији кода опет се отвара конекција, затим у 9. Линији се позива функција DeleteBorrowStudent којој се прослеђују два параметра, ид изнајмљивања и назив табеле. Функција брише све податке о том изнајмљивању и ако није дошло ни до какве грешке током брисања података функција ће вратити bool вредност true, а ако је дошло до грешке враћа bool вредост false (Више речи о овој функцији у секцији 2.5.1.). Затвара се конекција и проверава се вредност коју је функција вратила, ако је вредност true наставља се даље са извршавањем кода. Позива се функција changeState која функционише на сличан начин као у секцији 2.4. у којој се задужују књиге, само код изнајмљивања књига стање књига се смањивало за 1, а у овој функцији, пошто чланови враћају књигу, стање се повећава за 1. Након тога поље где је унет удк број књиге поставља се на почетну вредност, односно додељује му се вредност празног стринга. dataGridView-у додељује се вредност null, затим се прави упит који је сличан упиту за претрагу књига из главне форме, само се претрага врши по чланској карти члана који је управо вратио једну књигу. Затим се позива функција SearchBooks која функционише на исти начин као у главној форми и прослеђује јој се упит који је дефинисан у 17. линији кода и смештен у променљиву query. Ово пружа податке о томе да ли члан дугује још неку књигу поред књиге коју је тренутно вратио. На крају се појављује порука „Hvala Vam na vraćenoj knjizi!“, која обавештава корисника да је све прошло како треба. У случају да неки од услова у овој функцији није испуњен исписаће се порука о грешци.

### Функција DeleteBorrowStudent()

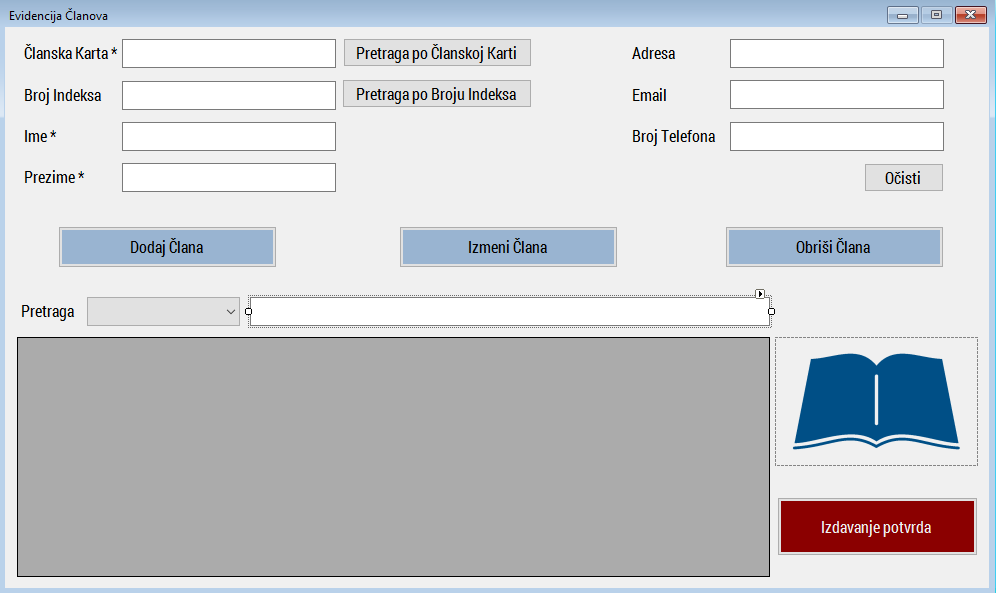
Ова функција је из класе MySQL и омогућава да се преко ид-ја обришу подаци о неком изнајмљивању. Функција прихвата 2 параметра ид изнајмљивања и назив табеле у базе где се то изнајмљивање налази. Такође функција враћа bool вредност, и она изгледа овако:

1. **public** **bool** DeleteBorrowStudent(**string** id, **string** table) {
2. **try**
3. {
4. cmd.CommandText = "delete from " + table + " where ID = " + id;
5. conn.Open();
6. MySqlDataReader reader = cmd.ExecuteReader();
7. **return** **true**;
8. }
9. **catch**
10. {
11. **return** **false**;
12. }
13. }

Креирана је try-catch конструкција и у случају било какве грешке при раду са базом података функција ће вратити вредност false. Затим у 4. линији кода направљен је упит „delete from " + table + " where ID = " + id“ који значи да се брише из табеле чији је назив прослеђен преко параметра функције, а бришу се подаци где је ид једнак такође прослеђеној вредности ид-ја. Након тога отвара се конекција и извршава се упит. Ако није дошло до грешке при извршавању упита функција ће вратити вредност true, у супротном враћа вредност false.

## Форма за евиденцију чланова

И ова форма отвара се из главне форме. Ово је форма у којој се води евиденција о члановима библиотеке. Омогућава додавање нових чланова, измену и брисање већ постојећих. Такође се могу видети и подаци о свим члановима библиотеке, као што су чланска карта, број индекса, име, презиме, адреса, емаил и број телефона. Да би додали или изменили члана потребно је да се попуне празна поља од којих су чланска карта, име и презиме обавезни. Док је за брисање члана потребно попунити само поље где се уноси чланска карта. Већ је познато како функционише дугме „Pretraga po Članskoj karti“ као и дугме „Pretraga po Broju indeksa“. Кликом на једно од та два дугмета врши се претрага свих чланова и у случају да постоји члан са том чланском картом или тим бројем индекса у осталим пољима се аутоматски попуњују његови подаци. У овој форми се такође налази претрага, која функционише на исти начин као и претрага из главне форме, само се овде чланови могу претраживати по чланској карти, броју индекса, имену или презимену. У форми се налази и дугме које се назива „Izdavanje potvrda“ и кликом на њега отвара се нова форма о којој ће више речи бити у секцији 2.6.1.



*Слика 2.7. Форма за евиденцију чланова*

Функција btnAdd\_Click служи за додавање чланова и покреће притиском на дугме „Dodaj Člana“. Код ове функције изгледа овако:

1. **private** **void** btnAdd\_Click(**object** sender, EventArgs e)
2. {
3. sql.Connection();
4. **bool** membersRead = sql.MembersRead(txtMembershipCard.Text, "clanovi");
5. sql.Close();
6. **if** (!membersRead)
7. {
8. sql.Connection();
9. **bool** result = sql.InsertMembers(txtMembershipCard.Text, txtIndexNumber.Text, txtFirstName.Text, txtLastName.Text, txtAddress.Text, txtEmail.Text, txtPhoneNumber.Text, "clanovi");
10. sql.Close();
11. **if** (result)
12. {
13. Members("", "", "", "", "", "", "");
14. **string** query = "select ClanskaKarta, BrojIndeksa, Ime, Prezime, Adresa, Email, Telefon from clanovi ORDER BY Prezime asc";
15. SearchMembers(query);
16. MessageBox.Show("Uspešno ste dodali člana!", "Informacija", MessageBoxButtons.OK, MessageBoxIcon.Information);
17. }
18. **else** MessageBox.Show("Greška pri dodavanju člana!", "Greška!", MessageBoxButtons.OK, MessageBoxIcon.Error);
19. }
20. **else** MessageBox.Show("Član već postoji!", "Greška!", MessageBoxButtons.OK, MessageBoxIcon.Error);
21. }

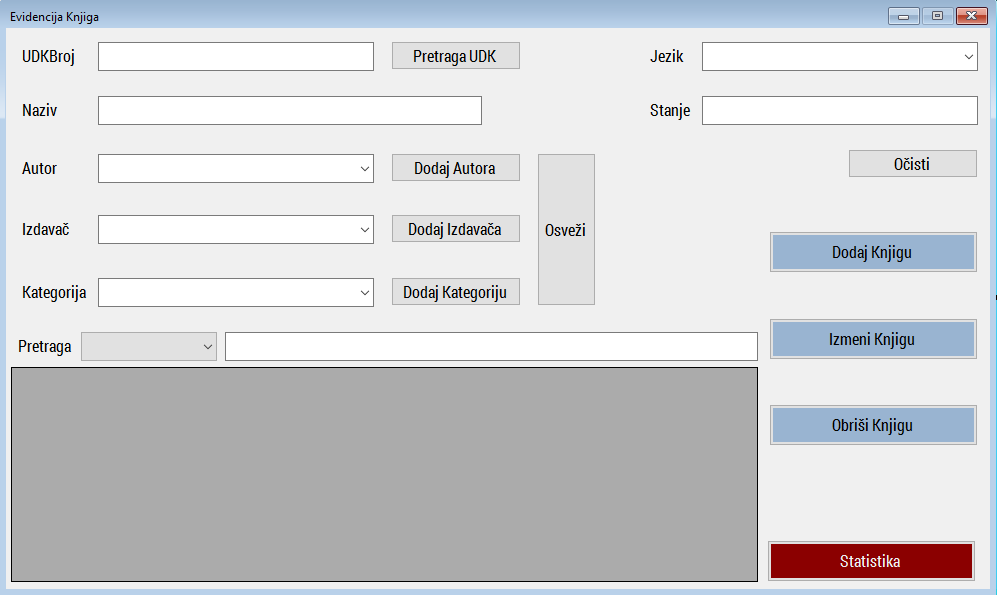
Отвара се конекција са базом података и позива се функцију MembersRead из MySQL класе. Функција прихвата 2 параметра, чланску карту и назив табеле у којој се чланови налазе. Она проверава да ли члан чија је чланска карта послата преко параметра функције постоји у бази и ако постоји враћа bool вредност true, а ако члан не постоји враћа bool вреднст false. Затвара се конекција и проверава се вредност коју је функција вратила, ако је вредност false, односно ако члан не постоји услов је испуњен и наставља се даље са извршавањем кода. Поново се у 8. линији кода отвара конекција, затим се у слиедећој линији кода позива функција InsertMembers из MySQL класе. Функција прихвата 8 параметара, чланску карту, број индекса, име, презиме, адресу, емаил, број телефона и назив табеле. Уз помоћ ове функције члан се додаје у базу и уколико не дође ни до каквог проблема приликом додавања функција враћа вредност true, у супротном враћа вредност false. Затвара се конекција и проверава се вредност коју је функција вратила, ако је вредност true услов је испуњен и улази се у if. У 13. линији кода позива се функција Members која сва поља враћа на почетно стање, у следећој линији кода пише се упит и позива се функцију SearchMembers која ради на истом принципу као функција SearchBooks која је већ објашњена. У 16. линији кода се исписује порука „Uspešno ste dodali člana!“, која служи као обавештење да је све прошло како треба. У случају да било који услов у овој функцији није био испуњен исписаће се порука о грешци.

### Форма за издавање потврде

Ова форма се отвара из форме за евиденцију чланова. На крају сваког семестра студенти су дужни да пре пријаве испита доставе потврду да не дугују ниједну књигу библиотеци. Ова форма омогућава библиотекару да то уради на једноставан и брз начин. Све што треба је да провери студента и уколико студент не дугује ниједну књигу да унесе његов индекс у одговарајуће поље у форми и кликом на дугме „Štampaj“ потврда ће бити иштампана.

## Форма за евиденцију књига

Ова форма се отвара из главне форме и функционише на сличан начин као и форма за евиденцију чланова из секција 2.6. Само у овој форми уносе се подаци о књигама као што су удк број, наслов, аутор, издавач, категорија, језик и стање. Кликом на дугме „Dodaj Autora“ отвара се нова форма у којој се води евиденција o аутору (Више речи у секцији 2.7.1.). Дугме „Dodaj Izdavača“ и „Dodaj Kategoriju“ такође отварају нове форме и функционишу на исти начин као и дугме „Dodaj Autora“. У овој форми се налази и претрага која се користи као и претраге у осталим формама. Књиге се могу претраживати по удк броју, наслову, аутору, издавачу или категорији. Постоји још једно дугме „Statistika“ које отвара нову форму (О којој ће више речи бити у секцији 2.7.2.). Форма за евиденцију књига изгледа као на Слици 2.7.



*Слика 2.7. Форма за евиденцију књига*

Ова форма има доста функција, али биће више речено о 2 функције, а то су: btnChange\_Click функција и btnDelete\_Click функција. btnChange\_Click функција се покреће кликом на дугме „Izmeni Knjigu“, док се btnDelete\_Click функција покреће кликом на дугме „Obriši Knjigu“. Код ових функција изгледа овако:

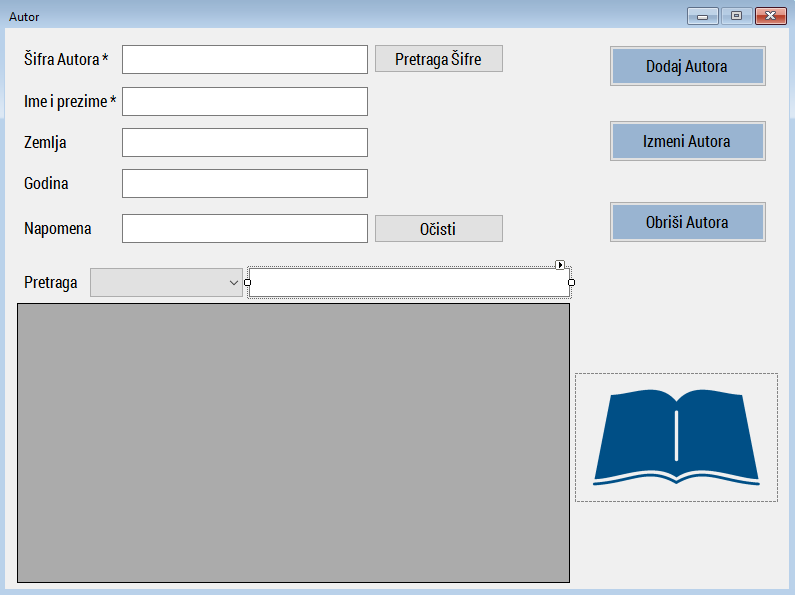
1. **private** **void** btnChange\_Click(**object** sender, EventArgs e)
2. {
3. sql.Connection();
4. **bool** bookRead = sql.BookRead(txtUDK.Text, "knjige");
5. sql.Close();
6. **if** (bookRead)
7. {
8. sql.Connection();
9. **bool** result = sql.UpdateBook(txtUDK.Text, txtName.Text, comboAuthor.Text, comboPublisher.Text, comboCategory.Text, comboLanguage.Text, txtState.Text, "knjige");
10. sql.Close();
11. **if** (result)
12. {
13. Book("", "", "", "", "", "", "");
14. **string** query = "select knjige.UDKBroj, knjige.Naziv , autori.ImePrezime as Autor, izdavaci.Naziv as Izdavač, kategorije.Naziv as Kategorija, jezik.Naziv as Jezik, knjige.Stanje from knjige inner join autori on knjige.SifraAutora = autori.Sifra inner join izdavaci on knjige.SifraIzdavaca = izdavaci.Sifra inner join kategorije on knjige.SifraKategorije = kategorije.Sifra inner join jezik on knjige.SifraJezika = jezik.Sifra ORDER BY knjige.Naziv asc";
15. SearchBooks(query);
16. MessageBox.Show("Uspešno ste izmenili knjigu!", "Informacija", MessageBoxButtons.OK, MessageBoxIcon.Information);
17. }
18. **else** MessageBox.Show("Greška pri izmeni knjige!", "Greška!", MessageBoxButtons.OK, MessageBoxIcon.Error);
19. }
20. **else** MessageBox.Show("Knjiga ne postoji!", "Greška!", MessageBoxButtons.OK, MessageBoxIcon.Error);
21. }
23. **private** **void** btnDelete\_Click(**object** sender, EventArgs e)
24. {
25. sql.Connection();
26. **bool** result = sql.DeleteBook(txtUDK.Text, "knjige");
27. sql.Close();
28. **if** (result)
29. {
30. Book("", "", "", "", "", "", "");
31. **string** query = "select knjige.UDKBroj, knjige.Naziv , autori.ImePrezime as Autor, izdavaci.Naziv as Izdavač, kategorije.Naziv as Kategorija, jezik.Naziv as Jezik, knjige.Stanje from knjige inner join autori on knjige.SifraAutora = autori.Sifra inner join izdavaci on knjige.SifraIzdavaca = izdavaci.Sifra inner join kategorije on knjige.SifraKategorije = kategorije.Sifra inner join jezik on knjige.SifraJezika = jezik.Sifra ORDER BY knjige.Naziv asc";
32. SearchBooks(query);
33. MessageBox.Show("Uspešno ste obrisali knjigu!", "Informacija", MessageBoxButtons.OK, MessageBoxIcon.Information);
34. }
35. **else** MessageBox.Show("Greška pri brisanju knjige!", "Greška!", MessageBoxButtons.OK, MessageBoxIcon.Error);
36. }

У функцији btnChange\_Click у 3. линији кода остварује се конекција са базом, затим се у 4. линији кода позива функција BookRead из класе MySQL. Функција прихвата 2 параметра, удк број књиге и назив табеле, затим проверава да ли књига, чији је удк број послат преко параметра функције, налази у бази. Ако се књига налази у бази функција враћа вредност true, у супротном враћа false. Након тога у 5. линији кода затвара се конекција функцојом Close. У 6. линији кода у if-у проверавамо да ли је враћена вредност true или false, ако је вредност true улазимо у if. Поново се отвара конекција, па у 9. линији кода позива функција UpdateBook из класе MySQL. Функција прихвата 8 параметара, удк број књиге, наслов, аутора, издавача, категорију, језик, стање и назив табеле. Ова функција налази књигу по њеном удк броју затим мења податке те књиге подацима који су прослеђени функцијом преко њених параметара. Ако је измена успешно обављена и није дошло до грешке током мењања података у бази функција ће вратити true, у супротном враћа false. У 10. линији кода затвара се конекција позивом функције Close. У 11. линији кода у if-у проверава се да ли је функција вратила true или false, ако је вратила true улази се у if. У 13. линији кода позива се функција Book која сва поља за унос поставља на почетно стање, тако што им преко параметара функције прослеђује празне стингове који им се касније додељују као вредности. У 14. линији кода написан је упит који се прослеђује као параметар у функцији SearchBooks написаној у 15. линији кода. У 16. линији кода позива се функција Show која отвара прозор са поруком „Uspešno ste izmenili knjigu!“. У случају да if из 6. или 11. линије кода врати вредност false исписаће се порука о грешци.

У функцији btnDelete\_Click у 25. линији кода остварује се конекција са базом, затим се у 26. линији кода позива функција DeleteBook из класе MySQL. Функција прихвата 2 параметра, удк број књиге и назив табеле, након чега функција покушава да обрише књигу преко удк броја који је послат преко параметара функције. Ако је књига са прослеђеним удк бројем успешно обрисана функција враћа вредност true, у супротном вратиће false. У 27. линији кода затвара се конекција са базом уз помоћ функције Close. У if-у се проверава да ли је вредност коју је функција вратила true или false, ако је вредност true улази се у if, у супротном ако је вредност false улази се у else и исписује порука „Greška pri brisanju knjige!“. У if-у у 30. линији кода позива се функција Book која је објашњена у претходној функцији btnChange\_Click. У 31. линији пишемо упит који прослеђујемо функцији SearchBooks написаној у 32. линији кода. На крају, у 33. линији кода позива се функција Show која отвара прозор са поруком „Uspešno ste obrisali knjigu!“.

### Форма за евиденцију аутора

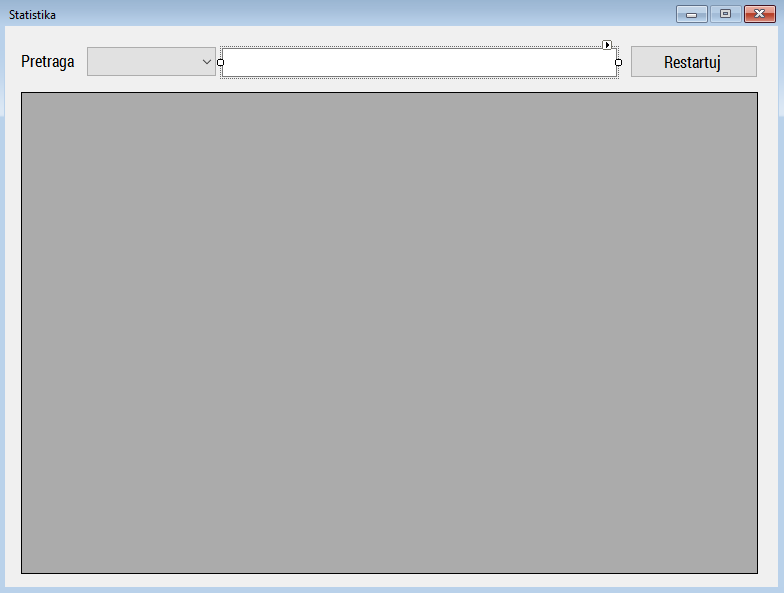
Ова форма отвара се из форме за евиденцију књига из секције 2.7., функционише као и остале форме за евиденцију. Форма садржи поља за унос података о аутору, а то су: шифра аутора, име и презиме, земља, година и напомена. Шифра аутора и име и презиме су обавезна поља која морају бити попуњена при додавању или измени. За брисање аутора потребно је попунити само поље у које се уноси шифра. Форма садржи и претрагу која функционише на исти начин као и остале претраге у апликацији, аутори се могу претраживати по шифри и по имену и презимену. Такође има 3 дугмета, „Dodaj Autora“, „Izmeni Autora“ и „Obriši Autora“, која раде на истом принципу као дугмићи из секције 2.7. (Форма за евиденцију књига). Ова форма изгледа као на Слици 2.8.



*Слика 2.8. Форма за евиденцију аутора*

### Форма за статистику

Ова форма пружа увид у статистику књига које су се највише пута изнајмљивале. Отварањем форме добија се списак изнајмљених књига и њихов број изнајмљивања поређаних тако да књига која се највише пута изнајмљивала буде прва. Форма има претрагу која функционише на исти начин као и у осталим формама, а изнајмљене књиге се могу претраживати по удк броју, наслову, аутору, издавачу или категорији. У овој форми има смао једно дугме „Restartuj“ и кликом на њега бришу нам се сви подаци о изнајмљеним књигама, затим наредна изнајмњивања крећу да се броје од почетка.



*Слика 2.9. Форма за статистику*

Рећићемо нешто више о функцији btnRestart\_Click која се покреће кликом на дугме „Restartuj“. Код функције изгледа овако:

1. **private** **void** btnRestart\_Click(**object** sender, EventArgs e)
2. {
3. sql.Connection();
4. **string** result = sql.EmptyAllRows("statistika");
5. sql.Close();
6. **if**(result != "")
7. {
8. MessageBox.Show("Došlo je do greške pri restartovanju!", "Greška!", MessageBoxButtons.OK, MessageBoxIcon.Error);
9. }
10. **else**
11. {
12. dataGridView1.DataSource = **null**;
13. **string** query = "select statistika.UDKBroj, knjige.Naziv, autori.ImePrezime as Autor, izdavaci.Naziv as Izdavač, kategorije.Naziv as Kategorije, jezik.Naziv as Jezik, statistika.BrojIznajmljivanja from statistika inner join knjige on statistika.UDKBroj = knjige.UDKBroj inner join autori on knjige.SifraAutora = autori.Sifra inner join izdavaci on knjige.SifraIzdavaca = izdavaci.Sifra inner join kategorije on knjige.SifraKategorije = kategorije.Sifra inner join jezik on knjige.SifraJezika = jezik.Sifra ORDER BY statistika.BrojIznajmljivanja desc";
14. SearchBooks(query);
15. }
16. }

У 3. линији кода позива се функција Connection која омогућава конекцију са базом, затим у 4. линији кода позива се функција EmptyAllRows која се користи како би обрисали све податке везане за статистику која се води. Ако није дошло до грешке приликом брисања функција ће вратити празан стринг (О овој функцији биће више речи у секцији 2.7.3.). У 5. линији кода затвара се конекција позивом функције Close, док у 6. линији кода у if-у проверавамо да ли је финкција вратила вредност која није празан стринг. Ако вредност коју је функција вратила није празан стринг улази се у if и позива функција Show која приказује поруку на екрану „Došlo je do greške pri restartovanju!“. Ако је функција вратила празан стринг улази се у else у 13. линији кода правимо упит који у 14. линији кода прослеђујемо функцији SearchBooks kaо њен параметар.

### Функција EmptyAllRows()

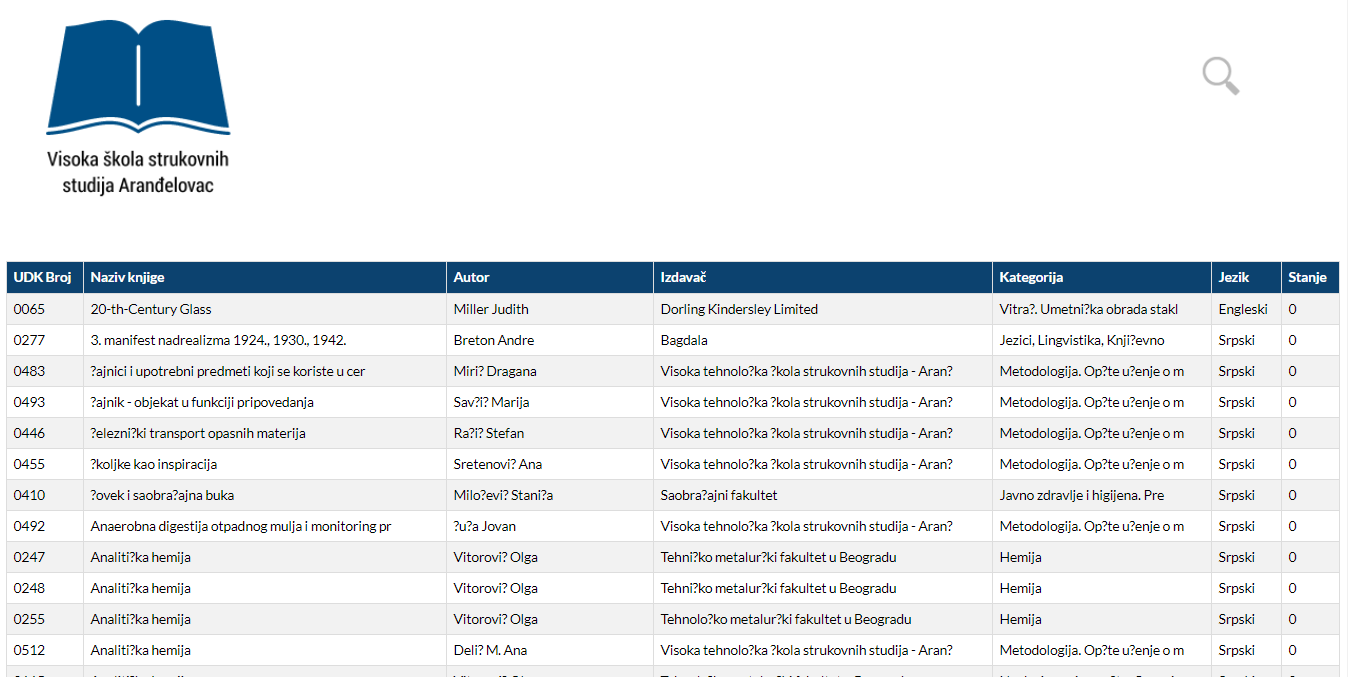
Ова функција се налази у MySQL класи. Она брише све податке из табеле чији је назив прослеђен преко параметара функције. Код ове функције изгледа овако:

1. **public** **string** EmptyAllRows(**string** table)
2. {
3. **try**
4. {
5. ReadID(table);
6. **for** (**int** i = 0; i < id.Count; i++)
7. {
8. cmd.CommandText = "delete from " + table + " where ID = " + id[i];
9. conn.Open();
10. MySqlDataReader reader = cmd.ExecuteReader();
11. conn.Close();
12. }
13. **return** "";
14. }
15. **catch** (Exception ex)
16. {
17. **return** ex.Message;
18. }
19. }

У 3. линији кода дефинисана је try-catch конструкција, у 5. линији кода позвана је ReadID функција којој се прослеђује један параметар. Параметар који се прослеђује функкцији је назив табеле, затим функција враћа листу свих id-јева из те табеле. У 6. линији кода налази се for петља која извршава код у њој онолико пута колико id-јева има у листи. Сваким проласком кроз for петљу брише се по једна колона у табели, тако да for престаје са радом оног тренутка када табела буде празна. Ако не дође ни до какве грешке приликом брисања финкција ће вратити празан стринг.

# Веб страница школске библиотеке ВШАР

Ова страница је намењена свима, а највише студентима ВШАР. Она пружа брз и једноставан увид у све књиге које се налазе у библиотеци. Самим отварањем странице излази табела са свим књигама, ту студенти могу видети податке као и тренутно стање књига у библиотеци. У десном углу странице се налази претраживач који пружа брзу претрагу њига по удк броју, наслову књиге, аутору, издавачу, категорији или језику. Веб страница изгледа као на Слици 3.1.



*Слика 3.1. Веб страница школске библиотеке ВШАР*

Ова Веб страница је одрађена у JetBrains-овом окружењу у PhpStorm-у. Главни код је писан у Php-у и један део кода изгледа овако:

1. <?php
3. $connect = mysqli\_connect("localhost", "root", "", "vts\_biblioteka");
5. **if** (isset($\_POST['search']) && $\_POST['search'] != "Vrednost za pretragu") {
6. $valueToSearch = $\_POST['valueToSearch'];
7. // search in all table columns
8. // using concat mysql function
10. $query = "select knjige.UDKBroj, knjige.Naziv, autori.ImePrezime as Autor, izdavaci.Naziv as Izdavac, kategorije.Naziv as Kategorija, jezik.Naziv as Jezik, knjige.Stanje from knjige inner join autori on knjige.SifraAutora = autori.Sifra inner join izdavaci on knjige.SifraIzdavaca = izdavaci.Sifra inner join kategorije on knjige.SifraKategorije = kategorije.Sifra inner join jezik on knjige.SifraJezika = jezik.Sifra WHERE CONCAT(LOWER(knjige.UDKBroj), LOWER(knjige.Naziv), LOWER(autori.ImePrezime), LOWER(izdavaci.Naziv), LOWER(kategorije.Naziv), LOWER(jezik.Naziv)) LIKE '%" . mysqli\_escape\_string($connect, strtolower($valueToSearch)) . "%' ORDER BY knjige.Naziv asc";
12. $search\_result = filterTable($query, $connect);
14. } **else** {
15. $query = "select knjige.UDKBroj, knjige.Naziv, autori.ImePrezime as Autor, izdavaci.Naziv as Izdavac, kategorije.Naziv as Kategorija, jezik.Naziv as Jezik, knjige.Stanje from knjige inner join autori on knjige.SifraAutora = autori.Sifra inner join izdavaci on knjige.SifraIzdavaca = izdavaci.Sifra inner join kategorije on knjige.SifraKategorije = kategorije.Sifra inner join jezik on knjige.SifraJezika = jezik.Sifra ORDER BY knjige.Naziv asc";
17. $search\_result = filterTable($query, $connect);
18. }

У 3. линији кода прави се конекција са базом тако што се позива функција mysqli\_connect и прослеђују јој се 4 параметра а то су: назив сервера, корисничко име, лозинка и назив базе. У 5. линији кода проверава се да ли је постављена вредност search која је послата POST методом и да ли та вредност није једнака стрингу „Vrednost za pretragu“, ако су оба услова испуњена улази се у if. У 6. линији кода креира се променљива valueToSearch којој се додељује вредност претраге послата POST методом. У 10. линији кода пише се упит, а у 12. линији кода тај упит као и конекција се прослеђују функцији filterTable о којој ће више речи бити у секцији 3.1. Ако услови из if-a нису испуњени улази се у else у 14. линији кода и такође се прави упит који се прослеђује са конекцијом функцији filterTable.

Део кода у ком се креира табела која се налази на Веб страници изгледа овако:

1. <table align="center" ;>
2. <tr>
3. <th>UDK Broj</th>
4. <th>Naslov knjige</th>
5. <th>Autor</th>
6. <th>Izdavač</th>
7. <th>Kategorija</th>
8. <th>Jezik</th>
9. <th>Stanje</th>
10. </tr>
12. <!-- populate table from mysql database -->
13. <?php **while** ($row = mysqli\_fetch\_array($search\_result)): ?>
14. <tr>
15. <td><?php echo $row['UDKBroj']; ?></td>
16. <td><?php echo $row['Naziv']; ?></td>
17. <td><?php echo $row['Autor']; ?></td>
18. <td><?php echo $row['Izdavac']; ?></td>
19. <td><?php echo $row['Kategorija']; ?></td>
20. <td><?php echo $row['Jezik']; ?></td>
21. <td><?php echo $row['Stanje']; ?></td>
22. </tr>
23. <?php **endwhile**; ?>
24. </table>

У 1. линији кода креирамо табелу отварањем тага table који дефинише границе табеле. Унутар овог тага користимо најосновније тагове за прављење табеле а то су: tr, th и td. Tr таг дефинише ред табеле. Насловне ћелије табела се дефинишу th тагом, док се садржај ставља унутар td тагова. У th тагове стављени су „UDK Broj“, „Naslov knjige“, „Autor“, „Izdavač“, „Kategorija“, „Jezik“ и „Stanje“, док су td тагови попуњени вредностима из MySQL базе података. У 13. линији кода отвара се php таг и прави while петља која ишчитава податке из базе, затим ти подаци се исписују у td тагове и приступа им се преко кључа.

## Функција filterTable()

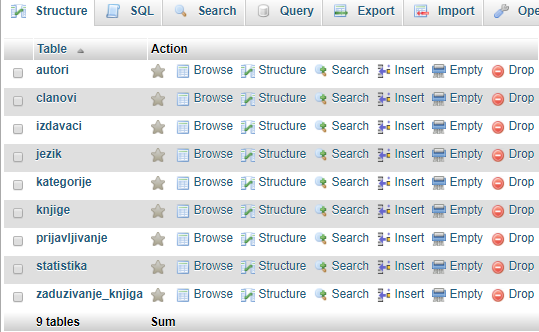
Ова функција прихвата 2 параметра, конекцију са базом и упит који треба да се изврши. Код функције изгледа овако:

1. // function to connect and execute the query
2. **function** filterTable($query, $conn)
3. {
4. $filter\_Result = mysqli\_query($conn, $query);
5. **return** $filter\_Result;
6. }
7. ?>

У 4. линији кода извршава се упит тако што се позива функција mysqli\_query и прослеђују јој се два параметра, конекција и упит. У 6. линији кода функција враћа резултат извршавања упита.

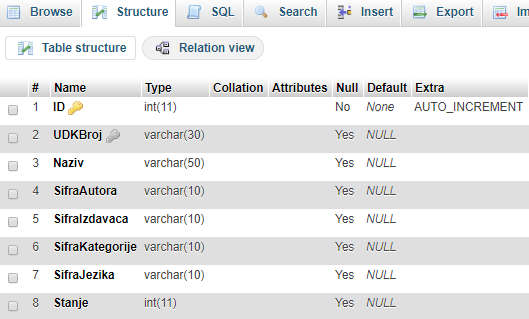
# База података vts\_biblioteka

Ова база је рађена у MySQL-у. Садржи 9 табела које су међусобно повезане. Називи табела могу се видети на Слици 4.1.



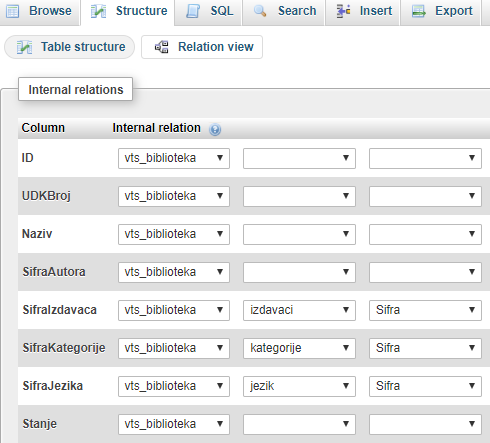
*Слика 4.1. Називи табела у бази података vts\_biblioteka*

Биће више речено о табели „knjige“ и табели „zaduzivanje\_knjiga“. Структуру табеле „knjige“ можете видети на Слици 4.2.



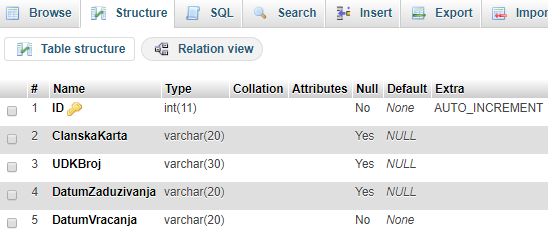
*Слика 4.2. Структура табеле „knjige“*

Из структуре табеле „knjige“ види се да табела има 8 колона, а то су: „ID“, „UDKBroj“, „Naziv“, „SifraAutora“, „SifraIzdavaca“, „SifraKategorije“, „SifraJezika“ и „Stanje“. „ID“ је примарни кључ, док су „SifraAutora“, „SifraIzdavaca“, „SifraKategorije“ и „SifraJezika“ спољни кључеви и они су повезани у релацији која може да се види на Слици 4.3. „UDKBroj“ је јединствен, што значи да не може да се понавља.



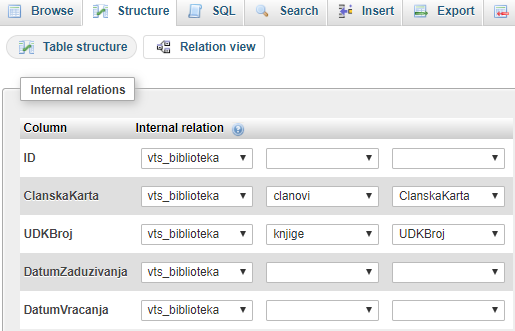
*Слика 4.3. Релација табеле „knjige“*

Структуру табеле „zaduzivanje\_knjiga“ можете видети на Слици 4.4.



*Слика 4.4. Структура табеле „zaduzivanje\_knjiga“*

Табела има 5 колона, „ID“, „ClanskaKarta“, „UDKBroj“, „DatumZaduzivanja“ и „DatumVracanja“. „ID“ је примаран кључ, док су „ClanskaKarta“ и „UDKBroj“ спољни кључеви чија релација може да се види на Слици 4.5.



*Слика 4.5. Релација табеле „zaduzivanje\_knjiga“*

# Закључак

У раду смо представили Windows апликацију која олакшава рад библиотекару и Веб страницу која пружа потребне информације студентима.

За будуће унапређење Windows апликације може се додати бар код читач на чланским картицама. Док се Веб страна може одрадити у frimeworku Laravel и интегрисати са већ постојећим пројектом за online пријаву испита. Такође се база за библиотеку може повезати са већ постојећом базом података у ВШАР.

Целокупно унапређење библиотеке могуће је и креирањем апликације за мобилне телефоне, која ће студентима пружити увид у тренутно стање књига.

# Референцe

[1] C# programming,

<https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/>

[2] Microsoft Visual Studio,

<https://www.visualstudio.com/>

[3] Client-server architecture,

<https://www.britannica.com/technology/client-server-architecture>

[4] JetBrains PhpStorm,

<https://www.jetbrains.com/>

[5] MySQL database,

https://www.mysql.com/