МЕЃУНАРОДЕН СЛАВЈАНСКИ УНИВЕРЗИТЕТ
СВЕТИ НИКОЛЕ
РЕПУБЛИКА СЕВЕРНА МАКЕДОНИЈА



ФАКУЛТЕТ ЗА ИНФОРМАТИКА
СТУДИСКА ПРОГРАМА: КОМЈУТЕРСКИ СИСТЕМИ И МРЕЖИ

МАГИСТЕРСКА РАБОТА

РАБОТА СО БАЗИ НА ПОДАТОЦИ И РАЗВОЈ НА SQL SERVER АПЛИКАЦИИ ВО VISUAL STUDIO

СТУДЕНТ: (КРИСТИЈАН ВАСИЛЕСКИ)
БРОЈ НА ИНДЕКС: M208008
МЕНТОР: (ПРОФ.Д-Р. МАРИЈАН СТЕВАНОВСКИ)

Битола, 2025

Содржина

1. ВОВЕД ..... - 3 -

2. ОСНОВНИ КОНЦЕПТИ И ТЕРМИНИ ..... - 5 -

2.1.Табели, Релации и записи ..... - 18 -

2.2. SQL јазик за манипулација со податоци ..... - 21 -

3. КРЕИРАЊЕ И МАНИПУЛИРАЊЕ СО БАЗИ НА ПОДАТОЦИ ..... - 23 -

3.1.Креирање на табели и дефинирање на релации ..... - 23 -

3.2. Внесување, промена и бришење на податоците ..... - 27 -

3.3. Извлекување на податоци со SQL прашалник ..... - 30 -

3.4. Процедури и функции во SQL Server ..... - 43 -

4. РАЗВОЈ НА SQL SERVER АПЛИКАЦИИ ВО VISUAL STUDIO ..... - 45 -

4.1. Конекција на апликација со база на податоци ..... - 45 -

4.2. Извршување на SQL прашалници во кодот ..... - 47 -

5. ДИЗАЈ И РАЗВОЈ НА КОРИСНИЧКИ ИНТЕРФЕЈС ..... - 52 -

5.1. Развој на кориснички интефејс ..... - 52 -

5.2. Дизајн на кориснички интефејс ..... - 59 -

6. ЗАКЛУЧОК ..... - 71 -

7. КОРИСТЕНА ЛИТЕРАТИРА ..... - 72 -

1. ВОВЕД

Во последните години во образовниот систем, како секаде во светот, така и кај нас, се сретнуваат множество на иновации и нивно користење. Самиот тренд да се користи нешто ново од пазарот влијае на сите дејности во општеството. Примената на новите иновации во информатичкиот сектор влијае и го менува начинот на живот и целокупната организација на самата компанија.

Во која и да е дејност во општеството со користење на напреден софтвер и достапен интерфејс за ресурот кој е вработен во компанијата, се олеснува целокупниот процес на работа. Користењето бази на податоци дополнително врши пристап до сите внесени потребни податоци во базата како и нивна обработка и измени.

Во овој Магистерски труд се врши обработка на значењето на базата на податоци, креирање и манипулирање со податоците од базата и развој на апликација каде ќе се прикаже конекцијата на апликацијата со базата на податоци. При креирање на интефејсот ќе се врши конекција од секоја форма до базата на податоци, како и додавање на податоци во табела, измена на податоци од табела, листање на податоци од табела и бришење на податоци од табела.

Стандардниот јазик SQL комуницира со MySQL кој е систем за менаџирање на базите на податоци. Останати системи за менаџирање со базата на податоци се MS Access, Oracle, Sybase, Informix, Postgres и SQL Server.

Во користење на SQL Server (како сервер за базата на податоци) и Visual Studio (како апликација за креирање на форми и конекција до серверот) успешно е креирана апликација за Хотел во која се презентирани сите можности и напредни технологии на користење за овие две апликации.

Самиот поим SQL Server е место каде се чуваат податоците од корисниците и целокупните потребни податоци за работа на претходно дефинирана апликација, без разлика дали се работи за Web апликација или Windows апликација.

Visual Studio е (IDE) развојна околина развиена од Microsoft . Се користи за развој на следните компјутерски програми:

- веб-локации,
- веб-апликации
- веб-услуги,
- мобилни апликации.

Visual Studio-то може да произведе и мајчин и управуван код при што дополнително користи платформи за развој на софтвер на Microsoft, вклучувајќи ги и :

- Windows API ,
- Windows Forms ,
- Windows Presentation Foundation (WPF),
- Windows Store,
- Microsoft Silverlight .

2. ОСНОВНИ КОНЦЕПТИ И ТЕРМИНИ

Главните перфомански на БП се поделени на повеќе елементи и тоа:

- времето на одзив
- минималната цена
- минималната редунданција
- способност за пребарување
- запазување на интегритетот итн.

Самата трансакција е дел од апликативна програма со која се обавуваат операции врз БП и може да биде во еден од следните облици:

- Ажурирање на БП
- Пребарување на БП
- Редукција на голем број податоци во форма прифатлива за корисникот.

Трансакција е програма која има ограничена големина и опис. Транскакцијата е наменета да пристапува до податоци кои се поделени помеѓу повеќе корисници. Таа е низа на операции кои се вршат во базата на податоци со која се изведува определена логичка функција врз реалниот систем.

Важни својства на системите за управување со бази на податоци при работа со трансакции се:

- Атомичност (Atomicy) – има потреба сите операции над базата на податоци успешно да се завршат или ниедна да не се изврши, изразена преку две специфични операции над БП:
  - COMMIT - успешен крај на трансакцијата која ги “потврдува“ сите промени во базата на податоци што биле предизвикани со трансакцијата
  - ROLLBACK - се поништуваат сите претходни операции над базата на податоци во една трансакција, доколку заради некои предвидени или непредвидени грешки БП може да се доведе до неконзистентна состојба.

- Конзистентност (Consistency) - на логичко ниво, обезбедува програмерот при пишување на апликацијата. Конзистентноста е единствен проблем што треба да ја разреши БП и, истовременото овозможува пристап на повеќе корисници притоа вршејќи промени на податоците.
- Изолација (Isolation) – при истовремено извршување на две или повеќе трансакции, нивните ефекти мора да бидат меѓусебно изолирани, ефектите предизвикани од трансакциите кои се извршуваат, мора да бидат истовремено еднакви со ефектите на нивното сериско извршување.
- Трајност (Durability) – по завршувањето на трансакцијата, нејзините ефекти не смеат да бидат изгубени, па дури и при откажување на системот.

Секоја трансакција мора да ја остави базата на податоци во конзистентна состојба, што е основа за сигурна работа на системот. Процесот на трансакција претпоставува дека пресметките во рамките на трансакцијата се временски и просторно мали. Долгите трансакции можат да ги оневозможат сите други трансакции во системот.

Најчесто трансакциите се состојат од:

- повици до системот за управување со бази на податоци или до системот на датотеки за читање и запишување,
- ограничени логички и нумерички пресметки и финансиски извештаи на корисничките терминали.

Процесорот на трансакции употребува различни техники на заштита при работа со трансакции, но секој бара т.н. дневник (log) на дупли информации. Постапката на логирање и обнова на податоците ја обезбедува СУБП. Постојат два начини на работа на програмите:

- Пакетен (BATCH) начин – во овај начин на работа барањата за некоја операција врз базата на податоци се собираат во пакет, а потоа се обработуваат заедно во погодно време.
    - Batch начинот ја користи базата на податоци многу ефикасно, бидејќи индивидуалните барања се подредуваат по оптимален редослед.

- Во овај начин на работа на програмите може чекањето за одредена информација да трае долго време.

- Непосреден (ON-LINE) начин, тука секое барање се обработува веднаш по внесувањето во системот и тоа:
  - Има потреба од користење на терминали за барањата и резултатот да се прикаже веднаш.
  - Секое непосредно барање се изведува преку трансакција.

Додека една трансакција ажурира одредени податоци, друга трансакција не смее да пристапи до истите податоци, бидејќи може да се добијат неконзистентни резултати. Податоците кои се ажурирани се "заклучуваат" и е оневозможен симултан пристап до нив од други трансакции. Времето на изведување на трансакцијата е ограничено за да не се попречи изведувањето на други трансакции.

Користењето на BATCH-програмите мора да е внимателно бидејќи има различни својства:

- Може да ја запрат работата на ON-LINE-трансакциите,
- Може да доведат до погрешни резултати, бидејќи во текот на нивната работа може да се случи некоја друга трансакција да изведе ажурирање на базата на податоци.
- Едно решение е да се има копија на податоците надвор од базата на податоци од каде BATCH-програмата ги добива потребните податоци.
  - имајќи две расположиви копии за читање, системот обезбедува високи перформанси за операциите на пребарување и поставување прашања
  - обезбедува просечно време на одзив на трансакција од 1,5 s, со што 90% од трансакциите завршуваат за помалку од 3 s
  - Системот е конфигуриран со повеќе илјади терминали и секој терминал може да прати по една порака во минута.

Постојат повеќе видови модели на податоци и тоа :

- Модел ентитет-релација (ER модел)
- Хиерархиски модел
- Мрежен модел
- Релационен модел
- Објектно-ориентиран модел
- Објектно-релационен модел
- XML модел

Моделот ентитет-релација (Entity-Relationship) или ER модел е основа за унифицирање на различните погледи врз податоците. Како семантички модел на податоци - го претставува реалниот свет преку ентитетни множества и релации, вклучувајќи важни семантички информации и е основа на т.н. концепциско проектирање. Моделот ентитет-релација е најпопуларен и најчесто користен модел за проектирање на БП како јасен модел со кој се прави прецизен опис на природата на податоците и начинот на кој тие се поврзани.

ER моделот се користи за проектирање на БП во следните чекори:

- Идентификување на ентитетните множества и асоцијациите
- Идентификување на семантичките информации преку одредување на типот на асоцијациите
- Дефинирање на атрибутите и множествата на вредности
- Организирање на податоците во ER-дијаграми и одредување примарни клучеви.

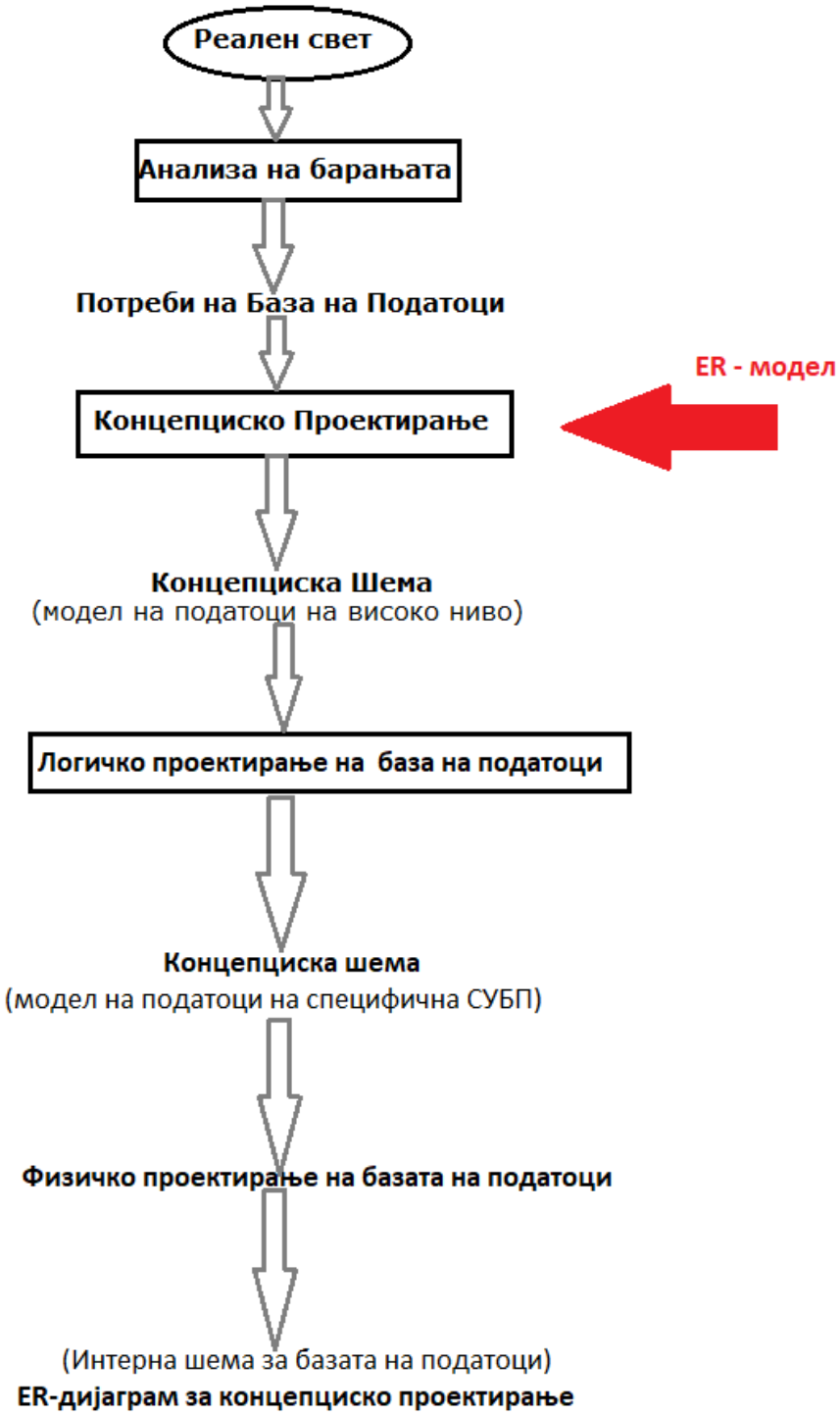
ER-дијаграм е графичка презентација на ентитетите и нивните атрибути, релациите помеѓу ентитетите и ограничувањата на интегритетот на ентитетите. Со ER-дијаграмот се дава графички приказ на структурата на БП што претставува добра основа за проценување на точноста, правилноста и нејзиниот понатамошен развој во процесот на нејзино дизајнирање.

ЕР модел се состои од следните елементи:

- множество ентитети
- дефинирани атрибути на секој од ентитетите
- релации помеѓу ентитетите
- ограничувања на интегритетот на ентитетите
- дефинирани клучеви на ентитетите
- ЕР-дијаграм
- дизајн на глобална шема
- трансформација на ЕР моделот во модел на БП.

Секој релационен модел на база на податоци се состои од три компоненти и тоа:

- Структура на податоци, која врши претставување на податоците (релации атрибути, домеини и.т.н.)
- Интегритет на податоци, кој врши заштита на податоците и нивно ограничување
- Манипулација на податоци е трета компонента која врши манипулирање на податоците, додавање, бришење и измена на податоците во базата на податоци.



Слика 1. Концепциско проектирање

“ Во дизајнот на релационите бази со податоци (RDBMS), **нормализација на бази со податоци** е процесот на организирање на податоците за да се намали оптоварувањето. Целта на нормализацијата на базите со податоци е да се прекинат врските со аномалии со цел да се направат помали, добро структуирани релации. Нормализацијата обично вклучува делење на големите табели во помали табели и дефинирање на врските помеѓу нив. Задачата е да се изолираат податоците со цел собирањата, бришењата и измените на полето да можат да бидат направени само во една табела и како такви да функционираат низ целата база со податоци, дефинирана со релации.

Едгар Ф. Код (Edgar F. Codd), пронаоѓачот на релациониот модел, го прикажал концептот на нормализација што нам ни е познат како Прва Нормална Форма (1NF) во 1970 година. Код работел на тоа да ја дефинира Втората Нормална Форма (2NF) и Третата Нормална Форма (3NF) во 1971г. Код и Рејмонд Ф.Бојс ја дефинираат Бојс-Кодовата Нормална Форма (BCNF) во 1974 година. Повисоките нормални форми биле дефинирани од други теоретичари во следните години. Најчеста била Шестата Нормална Форма (6NF), претставена од Крис Дејт, Хју Дарвен и Никос Лоренцос во 2002 година.

Неформално, табелата на релационата база со податоци (компјутерско остварување на релациите) често е опишано како “нормализирано” ако тоа е во Третата Нормална Форма. Повеќето 3NF табели немаат потреба од внесување, надградување и бришење на аномалии. На пример, во поголемиот дел од случаите 3NF табелите додаваат BCNF, 4NF и 5NF, но не и 6NF.

Стандардната база на податоци е направена така што творецот би можел да направи целосно нормализиран дизајн; селективна денормализација која подоцна може да се искористи за да се прават измени на табелата. Како и да е, некои видови на моделирање како што е на пример димензионалното моделирање, пристапуваат до складиштето на податоци кое е препорачано за ненормализирани дизајни. Тие во голема мерка не се залепени со 3NF.”[1]

[1] https://mk.wikipedia.org/wiki/Нормализација_на_бази_на_податоци

“Примарната цел на Првата Нормална Форма дефинирана од Код во 1970 г., е да овозможи податоците да бидат складирани и манипулирани користејќи “универзален под-јазик за податоци” втемелен во првобитната логика. (SQL е пример за таков под-јазик за податоци). Објекти на нормализација над 1NF според Код, се класифицирани на следниот начин:

- Да го ослободи збирот на релации од несакано внесување, ажурирање, како и бришење на зависности;
- За да се намали потребата за реструктурирање на релации како што се нови видови на податоци, а со тоа зголемување на работниот век при примената на програмите;
- Да се направи релационен модел повеќе информативен за корисниците;
- Да се направи колекција на релациите неутрални за пребарувањето и статистиката, каде што овие статистики се должни да се променат како што времето одминува.

-E.F. Codd "Понатамошна нормализација на База На Податоци (Релационен модел)"

Деловите подолу прикажуваат детали за секоја од овие цели.

Кога се прави обид да се измени (ажурирање, внесување во, или да се избрише од) табела, можна е појава на непожелни ефекти. Не сите табели може да страдаат од овие несакани ефекти, туку несакани ефекти само може да се појават во табели кои не се доволно нормализирани. Недоволно Нормална табела може да има една или повеќе од следните одлики: “[2]

[2] https://mk.wikipedia.org/wiki/Нормализација_на_бази_на_податоци

- “Истите информации може да се изразат во повеќе редови, па затоа ажурирање на табелата може да резултира со логички недоследности. На пример, секој запис во "Employees Skills" табелата може да содржи Employee ID, Employee Address и Skill, така што промената на адресата за одреден вработен потенцијално ќе треба да се промени на повеќе записи (по еден запис за секоја од неговите вештини). “[3]

  “Ако ажурирањето не се изврши успешно, вредностите ќе се изменат само на некои записи. Поточно, табелата дава спротивставени одговори. Овој феномен е познат како ажурирање на аномалија.

- Постојат околности во кои одредени факти не можат да бидат запишани на сите полиња. На пример, секој запис во "Faculty and Their Courses" табелата може да содржи FacultyID, Faculty Name, Faculty Hire Date и Course code. На тој начин можеме да ги снимиме деталите за кој било член на факултетот кој предава на најмалку еден курс, но не може да ги снимаме деталите на нововработените членови на факултетот кои сè уште не се назначени да ги предаваат сите предмети. Овој феномен е познат како вметнување на аномалија.

- Постојат околности во кои бришење на податоци претставува поставување на сосема различни факти. На "Faculty and Their Courses" табелата како што е опишано во претходниот пример страда од овој вид на аномалија, бидејќи ако факултетскиот член привремено престане да предава на сите курсеви, ние мораме да ја избришеме последната евиденција за која факултетскиот член се појавува. Овој феномен е познат како бришење на аномалија.

Кога целосно Нормалната основа на податоци се проширува, потребно е приспособување на новите типови на податоци. Постоечките аспекти на базата на податоци можат во голема мера или целосно да останат непроменети.”[4]

[3] https://mk.wikipedia.org/wiki/Нормализација_на_бази_на_податоци
[4] https://mk.wikipedia.org/wiki/Нормализација_на_бази_на_податоци

Нормализираните табели се погодни за општа намена. Ова значи дека какви било замерки за овие табели, вклучувајќи идни променливи, чии детали не можат да се предвидат, се поддржани. Спротивно на тоа, табели кои не се нормализирани се приспособуваат на некои видови, на некои и не.

Како Лорд Бајрон застанал против неговите современи поети? Одговорите на овие прашања мораат да дојдат од посебени адаптивни алатки, сосема различни од базата на податоци. Една од алатките може да биде софтвер напишан за да се справи со таквите пребарувања. Овој специјален адаптивен софтвер има само една единствена цел: ефектно нормализирање на не-нормализирано поле. Непредвидливи прашања може да се одговорат тривијално, во базата на податоци, со Нормална табела.

Пребарување и манипулирање со податоци во рамките на ненормална податочна структура, како што се следниве ненормализирани форми за застапености на клиентите, трансакции со кредитни картички, вклучува повеќе комплексност отколку што е навистина потребно: Секој клиент одговара на група на повторување на трансакции. За автоматско оценување во врска со трансакциите на клиентите настапуваат 2 фази:

- Отпакување на една или повеќе групи на клиенти на трансакции со што се овозможува поединечни трансакции во една група да бидат испитани.
- Изведување на пребарувачки резултат врз основа на резултатите од првата фаза.

Еден од важните увиди на Код беше дека оваа структурна комплексност секогаш може да се отстрани целосно, што доведува до многу поголема моќ и флексибилност во начинот на табелирање и како би можеле да бидат формулирани (од страна на корисниците и апликациите) и вреднувани (од страна на DBMS).

Функционалната зависност е дефинирана како:

Атрибутот X на релацијата е функционално зависен од атрибутот Y (клуч) на таа релација, ако атрибутот X има само една вредност поврзана со одредена вредност на атрибутот Y. Со

други зборови, ако вредноста на атрибутот Y е позната, поврзаната вредност на атрибутот X може да се утврди.

Тривијалната функционална зависност е функционална зависност на атрибут која повторно покажува сама на себе.

Целосната функционална зависност на Атрибутот е целосно функционално зависен од множество на атрибути X ако тоа е:

- функционално зависно од X, и
- не е функционално зависно од кое и да е соодветно подмножество на X. {Employee Adress} има функционална зависност од {Employee ID, Skill}, но не и целосно функционално зависно, поради тоа што е, исто така, зависи од {Employee ID}.

Преодната зависност е индиректна функционална зависност, во која x → Z, само врз основа на X → Y и Y → Z.

Зависноста со повеќе вредности е пречка според која присуството на одредени редови во табелата подразбираат присуство на некои други редови.

Табелата T е предмет на приклучна зависност ако T секогаш може да се создава повторно од спојување на повеќе табели, а секоја од нив има подмножество со особини на T.

Нормални форми (abbrev. NF) на релациона основа на податоци е теорија која треба да обезбеди критериуми за утврдување на степенот на ранливоста на табелата како и логички недоследности и аномалии. Колку повисоко се наоѓа Нормална форма на табелата, толку помалку е ранлива и поседува помалку недоследности и аномалии. Секоја табела има "највисок нормален облик" (HNF).

Нормалните форми се применуваат за поединечни табели. За да се каже дека целата база на податоци е во Нормална форма n треба сите нејзини табели да се во Нормална форма Н.

Бази на податоци наменети за онлајн трансакција (OLTP) се обично повеќе нормализирани од базите на податоци наменети за онлајн аналитичка обработка (OLAP). OLTP апликации се одликуваат со голем обем на мали трансакции, како што се ажурирање на продажбата во супермаркет. Во очекување е дека секоја трансакција ќе ја напушти базата на податоци во константна состојба. Спротивно на тоа, базите на податоци наменети за OLAP операции се "претежно за читање" бази на податоци. OLAP апликациите имаат тенденција да извлечат историски податоци што ги имаат акумулирано во текот на долг временски период.

Денормализација се користи за да се подобри ефикасноста на помали компјутери, како во електронски кеш регистри и мобилни уреди, бидејќи тие можат да ги користат податоците само за гледање. Денормализација, исто така, може да се користи кога нема RDBMS платформа (како Palm PC).[5]

[5] https://mk.wikipedia.org/wiki/Нормализација_на_бази_на_податоци

Денормализација може да биде наменска и корисна, облик кој не е во прва нормална форма е дефиниција за база на податоци со дизајни кој не се во согласност со првата Нормална форма, со тоа што дозволува да има атрибути со домени" (Schek 1982). Јазиците се користат за пребарување и манипулирање со податоци, моделот мора да се прошири во согласност со таквите вредности. Еден од начините на гледање на овој принцип е да се разгледа како структурирана вредност, како и специјализирани типови на вредности (домени), со свој домен-специфичен јазик. Сепак, она што обично се подразбира под не-1NF модел, е пристапот во кој релациониот модел и јазиците кои се користат за барањето, е да се прошири со општ механизам за таква структура; На пример, вгнезден релационен модел поддржува употреба на односи како што се домен-, вредности, со додавање на два дополнителни оператори (nest и unnest) на релационата алгебра што може да создаде и израмни вгнездени односи, соодветно."[6]

[6] https://mk.wikipedia.org/wiki/Нормализација_на_бази_на_податоци

Како во секоја област така и во базите на податоци постојат формални и неформални поими кои се користат за дадени компоненти.

- ✓ ДОМЕИН – се користи како множество дозволени вредности за податоците во колоните
- ✓ АТРИБУТ – се користи како име на колона во табелата
- ✓ ТОРКА, ИНСТАНЦА - се користи како редица во табелата
- ✓ РЕЛАЦИЈА - се користи како табела
- ✓ СТЕПЕН НА РЕЛАЦИЈА - се користи како број на колони во табелата
- ✓ КАРДИНАЛНОСТ НА РЕЛАЦИЈА - се користи како број на редици во табелата
- ✓ РЕЛАЦИОНА БАЗА НА ПОДАТОЦИ - се користи како множество табели.

2.1.Табели, Релации и записи

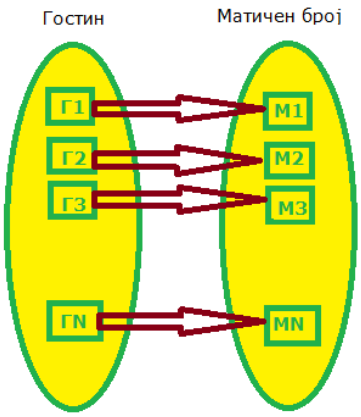
Табелите во SQL уште се познати како ентитети во базата на податоци. Ентитетот е елемент кој може да се појаснува со дополнителни елементи. Дополнителните елементи кои го појаснуваат ентитетот се викаат атрибути. Значи секоја табела е составена од атрибути кои се однесуваат на истата табела. Предноста на SQL серверот е можноста овие табели да се поврзат и да комуницираат меѓусебе. Постојат повеќе видови на релации и тоа :

- ➢ Релација 1:1
- ➢ Релација 1:M
- ➢ Релација M:1
- ➢ Релација M:M

Релација 1:1

За да се појасни релацијата 1:1 најдобро би било преку примери на користење. Значи самата релација 1:1 се однесува на два ентитети.

Пример 1 : Доколку првиот ентитет е Гостин и вториот ентитет е ЕМБГ односно матичен број. Во овај случај јасно е видливо дека секој вработен има по еден единствен матичен број и затоа релацијата помеѓу овие два ентитети е 1:1.



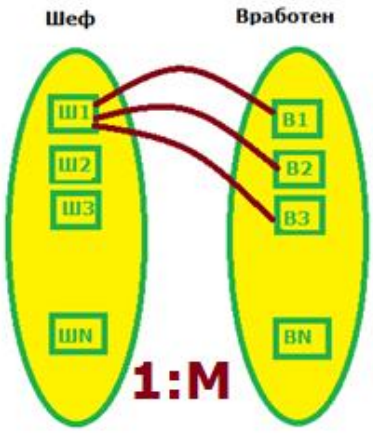
Слика 2. Приказ на релација 1:1

Пример 2: Доколку првиот ентитет е Вработен и вториот ентитет е матичен број на вработениот од хотелот. Имено јасно е дека секој вработен во хотелот има еден единствен матичен број на вработување во хотелот и затоа оваа релација е 1:1.

Релација 1:М

За да се појасни релацијата 1:М најдобро би било преку примери на користење. Значи самата релација 1:М се однесува на два ентитети.

Пример 1: Доколку првиот ентитет е Шеф а вториот ентитет е Вработен тогаш релацијата помеѓу овие два ентитети ќе биде 1:М тоа значи дека 1 Шеф е во релација со повеќе Вработени.

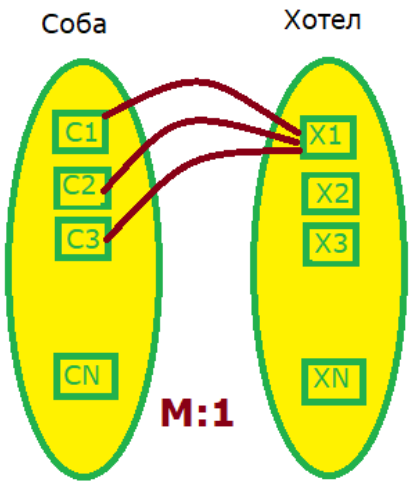


Слика 3. Приказ на релација 1:М

Пример 2: Доколку првиот ентитет е Шанкер а вториот ентитет е Келнер тогаш релацијата помеѓу овие два ентитети ќе биде 1:М тоа значи дека 1 Шанкер е во релација со повеќе Келнери.

Релација М:1

Пример 1: Доколку првиот ентитет е Соба, а вториот ентитет е Хотел тогаш релацијата помеѓу нив ќе биде М:1, тоа значи дека многу Соби се наоѓаат во еден Хотел.



Слика 4. Приказ на релација М:1

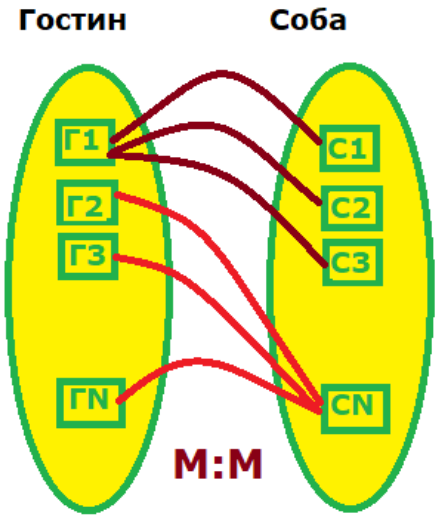
Пример 2: Доколку првиот ентитет е Гости, а вториот ентитет е Базен релацијата помеѓу нив ќе биде М:1, тоа значи дека многу гости ќе користат еден базен.

Релација М:М

Во релацијата М:М како битен елемент е дополнителното појаснување и настанокот на нов ентитет (односно нова табела), каде ќе се поврзат овие два ентитети и ќе се дополнат со дополнителен ентитет по потреба, во однос на барањата и потребите на самиот систем на податоци.

Пример 1: Доколку првиот ентитет е Гости а вториот ентитет е Соби тогаш релацијата помеѓу нив ќе биде М:М, тоа значи дека повеќе гости се сместени во повеќе соби. Новата табела каде ќе бидат споени овие два ентитети може како пример да биде резервација.

Таа табела има свој примарен клуч, еден клуч од едната табела и еден клуч од другата табела (тие два клуча се викаат секундарни клучеви) и плус дополнителни атрибути според барањата и организацијата на базата на податоци (пр. Дата резервација, тип соба, број денови, цена и.т.н.)



Слика 5. Приказ на релација M:M

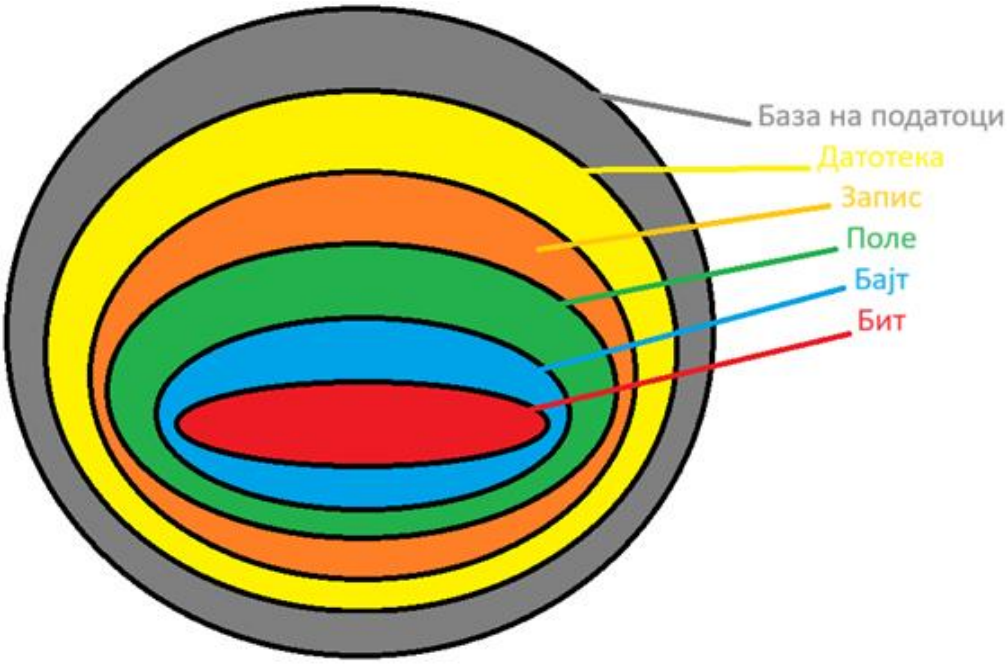
2.2. SQL јазик за манипулација со податоци

На следната табела е прикажано нивото за обработка на податоци и нивна манипулација.

1	Концепти и модел на податоци
2	Технологија на систем за претставување и управување со датотеки
3	Алгоритми и структури на податоци
4	Хардвер за обработка на информации

Базата на податоци е множество од датотеки. Секоја датотека е составена од записи. Записот е множество полиња за секој објект. Полињата претставуваат множество од еден

или повеќе последователни бајти. Секој бајт е најмалата организациска единица која содржи 8 бита. Секој бит е најмала мерка за количество податоци која можи да има веројатни состојби 0 или 1.



Слика 6. Мерки за количество на податоци

Предности на работењето преку БП:

- Полесен, побрз, поевтин и пофлексибилен развој на апликациите
- Промените во БП не бараат промени и во програмите
- Јасност во користењето на податоците
- Лесен пристап до податоците
- Можност за ad-hoc прашања
- Се намалува цената на користење и ажурирање на податоците
- Се обезбедува конзистентност на податоците
- Се постигнува поголема расположливост на податоците
- Се намалува на минимум редундантноста на податоците
- Се овозможува заштита и контрола на интегритетот на податоците
- Се овозможува обнова на податоците и нагодување на перформансите

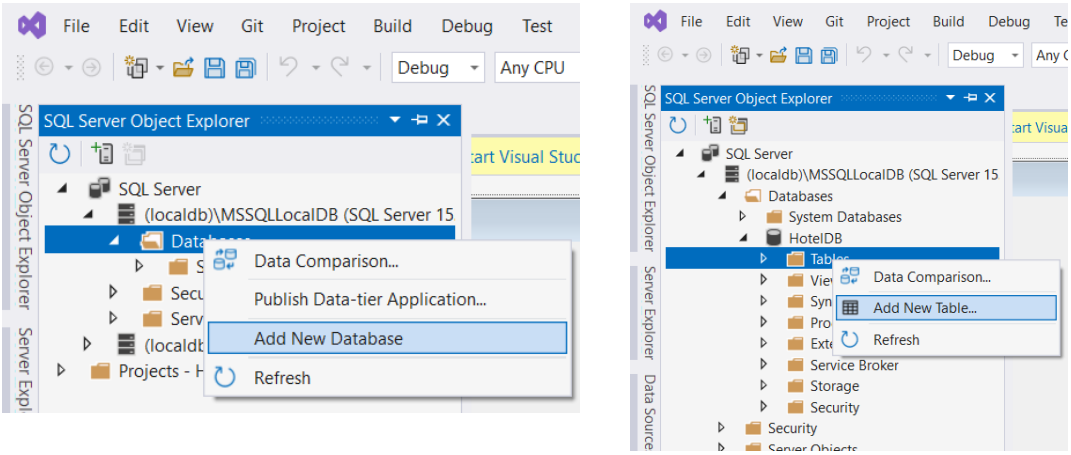
- Се овозможува перспективно минување на дистрибуирана обработка.

3. КРЕИРАЊЕ И МАНИПУЛИРАЊЕ СО БАЗИ НА ПОДАТОЦИ

За презентација на магистерската работа изработена е база на податоци на тема Хотел, со име кој се наоѓа во различни градови со различни сектори. Секој хотел има можност за резервација на соба, но и други гранки кои ги нуди самиот хотел. Во хотелот дополнително има сала за состаноци, кино сала, спа центар и ресторан кој може да се користат со резервации за секој клиент кој ќе има потреба од истото.

3.1.Креирање на табели и дефинирање на релации

При креирање на база на податоци на SQL серверот кој се користи во Visual Studio се внесува името на новата база која е потребна да се креира. По внесување на името на базата на податоци се внесуваат табелите соодветно со своите атрибути.



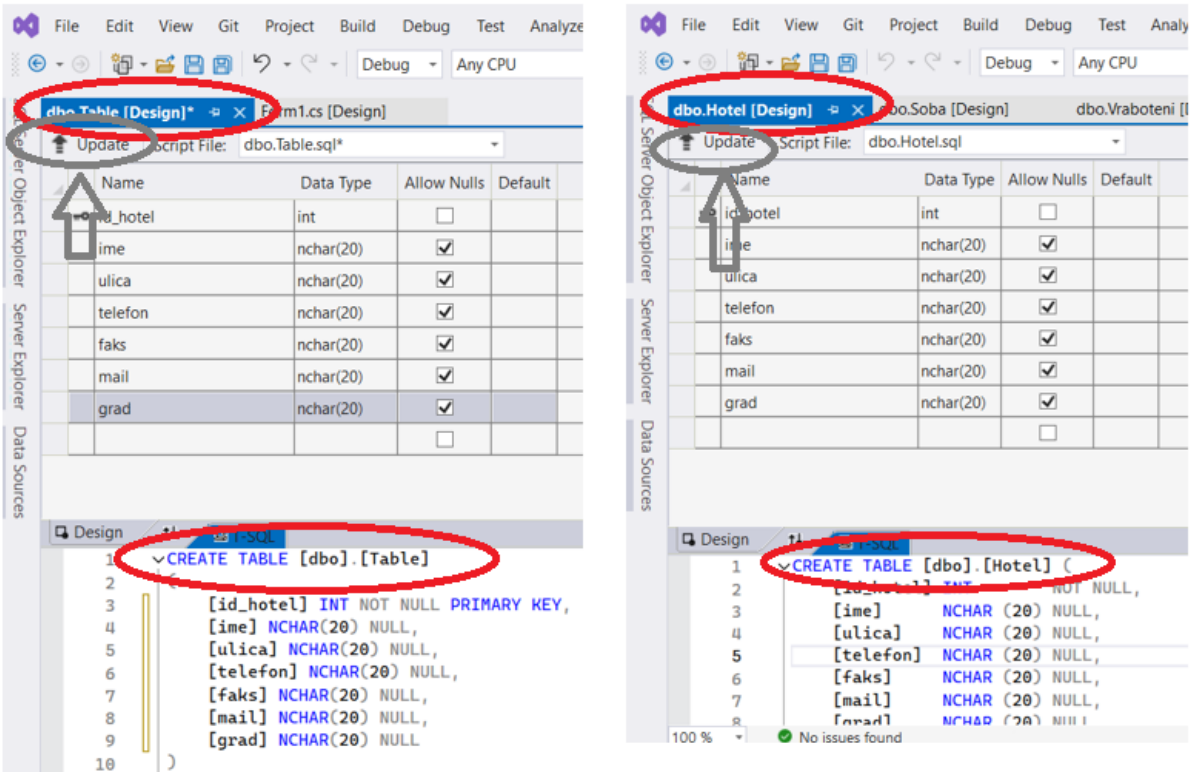
Слика 7. Креирање на база на податоци

Во базата на податоци се креирани ентитети со соодветни атрибути и тоа:

1. Хотел(id_hotel, ime, prezime, ulica, telefon, faks, mail, grad)
2. Сектор(id_sektor, id_hotel, ime, cena)
3. Гостин(id_gostin, id_soba ime, prezime, ulica, telefon, embg, mail)
4. Соба(id_soba, id_hotel, tip)
5. ГостинСектор(id_gostin_sektor, id_gostin, id_sektor, den)

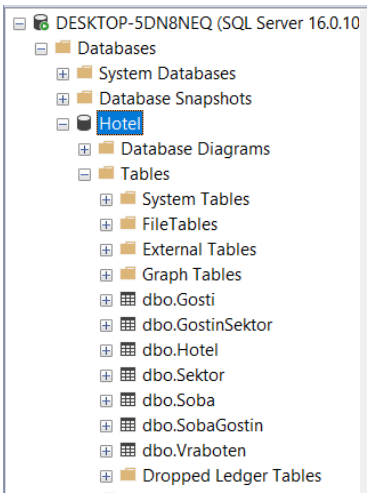
6. Вработен(id_vraboten, ime, prezime, KorisnickoIme, Lozinka, ulica, telefon, embg, mail)
7. СобаГостин(id_soba_gostin, id_soba, id_gostin, data_od, data_do v_cena)

На следната слика е прикажана постапката на креирање и зачувување на табелите со нивните атрибути.



Слика 8. Креирање на табела и нивните атрибути

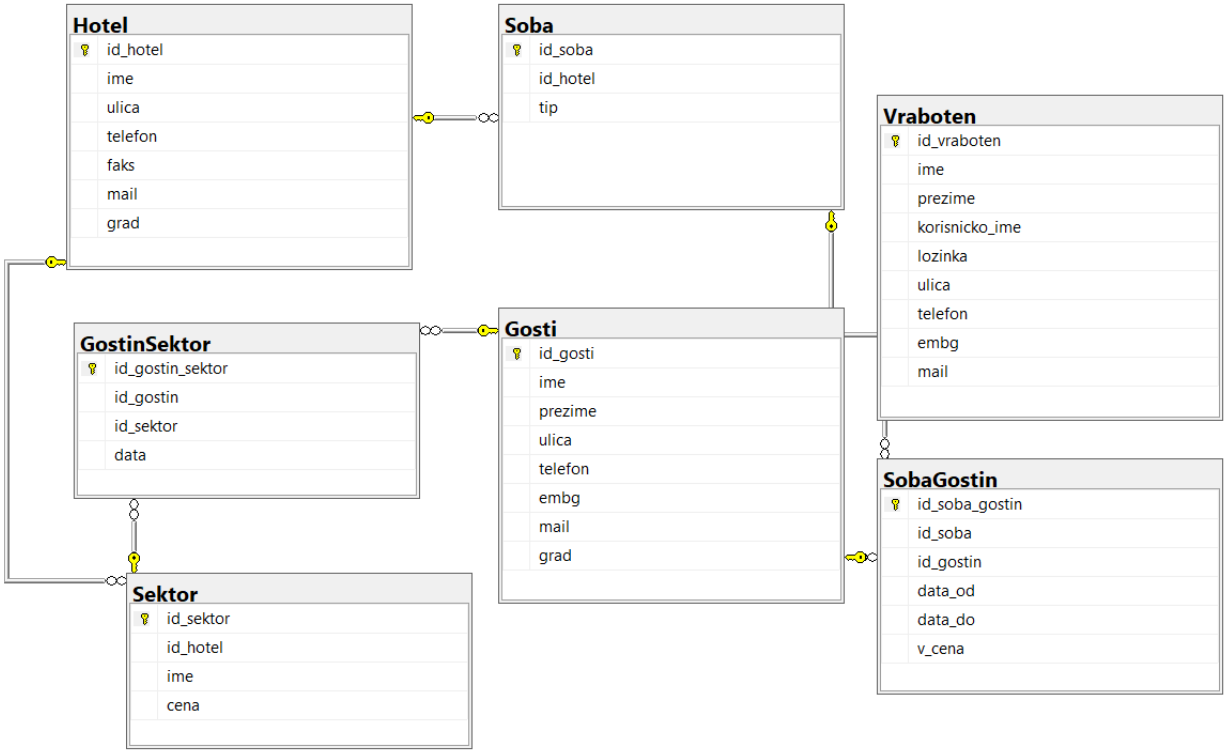
По креирање на сите податоци кои се потребни за обработката на оваа база на податоци, сликата со сите ентитети во базата на податоци е прикажана на следната слика.



Слика 9. Приказ на креирани табели во базата

За подетален преглед на базата на податоци е креиран дијаграм за приказ на сите ентитети и сите атрибути заедно со примарни и секундарни клучеви како и релациите помеѓу ентитетите. Само да направиме разлика помеѓу примарен и секундарен клуч. Примарен клуч е клучот кој се наоѓа во својата табела, додека секундарен клуч е клучот кој се користи од надворешна табела.

На следната слика е прикажан дијаграмот креиран во SQL Server Management Studio.



Слика 10. Приказ на database диаграм

Во табелата Хотел примарен клуч е id_hotel, додека дополнителни атрибути кои се чуваат во табелата се име, презиме, телефон, улица, факс, маил и град.

Во табелата Соба примарен клуч е id_soba, додека секундарен клуч е id_hotel и тип е дополнителен атрибут.

Во табелата Гостин примарен клуч е id_gostin, додека дополнителни атрибути кои се чуваат во табелата се име, презиме, улица, телефон, матичен број и маил .

Во табелата Сектор примарен клуч е id_sektor, додека секундарен клуч е id_hotel и дополнителни атрибути се име и цена.

Во табелата ГостинСектор примарен клуч е id_gostin_sektor, додека секундарни клучевисе id_gostin и id_sektor и дата е дополнителен атрибут.

Во табелата СобаГостин примарен клуч е id_gostin_soba, додека секундарни клучеви се id_gostin и id_soba и дополнителен атрибут дата од , дата до и вкупна цена.

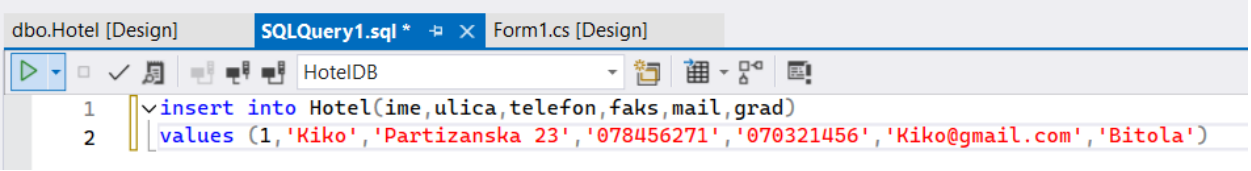
Во табелата Вработен примарен клуч е id_vraboten и дополнителни атрибути се име, презиме, корисничко име, лозинка, улица, телефон, матичен број и маил.

3.2. Внесување, промена и бришење на податоците

Во SQL сервер постојат четири видови на прашалници и тоа :

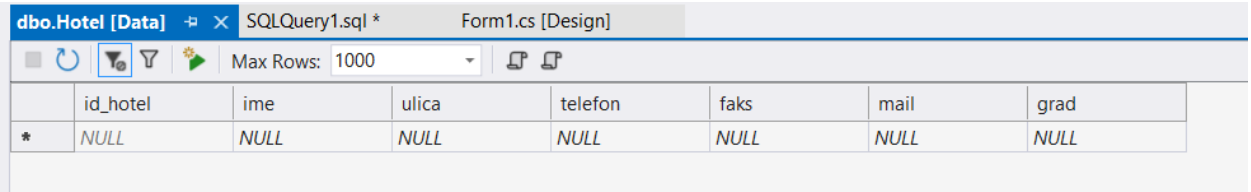
- Прашалник за прикажување на податоци
- Прашалник за внесување на податоци
- Прашалник за измена на податоци
- Прашалник за бришење на податоци.

Синтаксата за внесување на нов податок во еден ентитет е прикажана на следната слика:



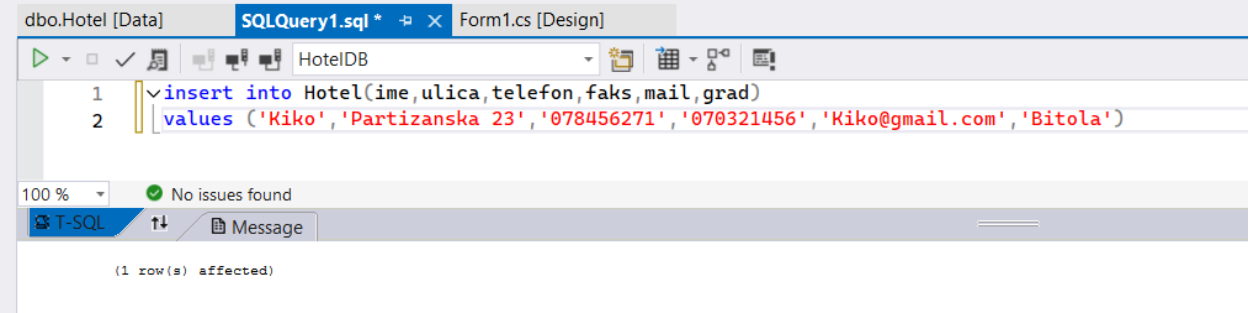
Слика 11. Приказ на прашалник за внесување податоци

Како пример на следната слика е прикажана празна табела без податоци.



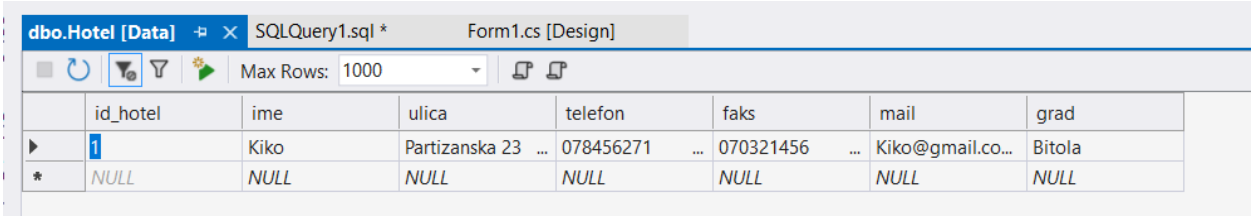
Слика 12. Приказ на празна табела

По извршувањето на прашалникот за внесување на податоци SQL испраќа порака за внесена една редица во табелата.



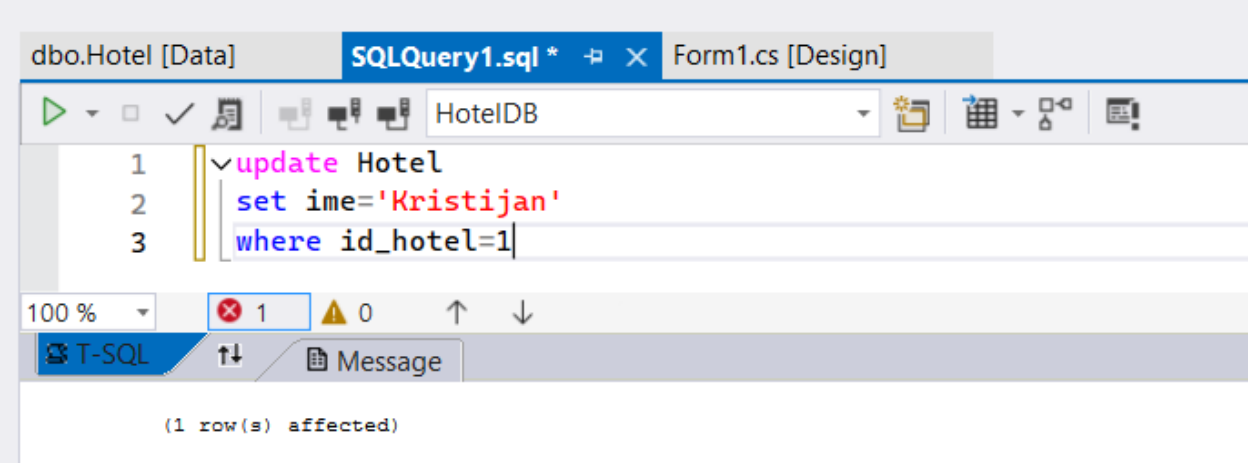
Слика 13. Приказ на извршен прашалник за внесување на податоци

За контрола на точно извршен прашалник на екран се прикажуваат податоците кои се внесени во прашалникот и истите се запишани во табелата, односно во базата на податоци.



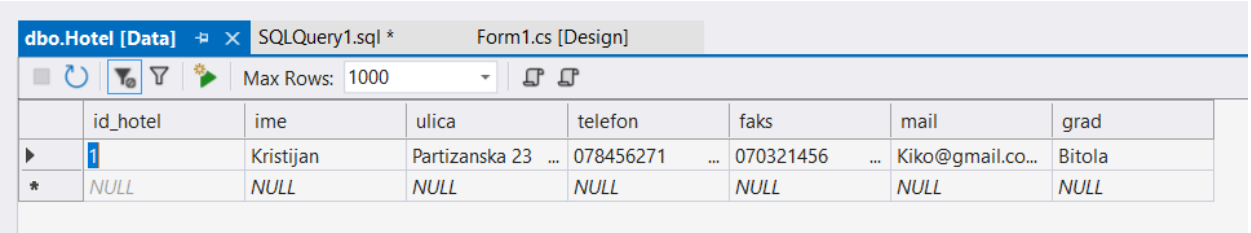
Слика 14. Приказ на табела после извршување на прашалник за внесување на податоци

Прашалник за измена на податоците во база на податоци односно во барана табела е прикажан на следната слика:



Слика 15. Приказ на прашалник за измена на податоци

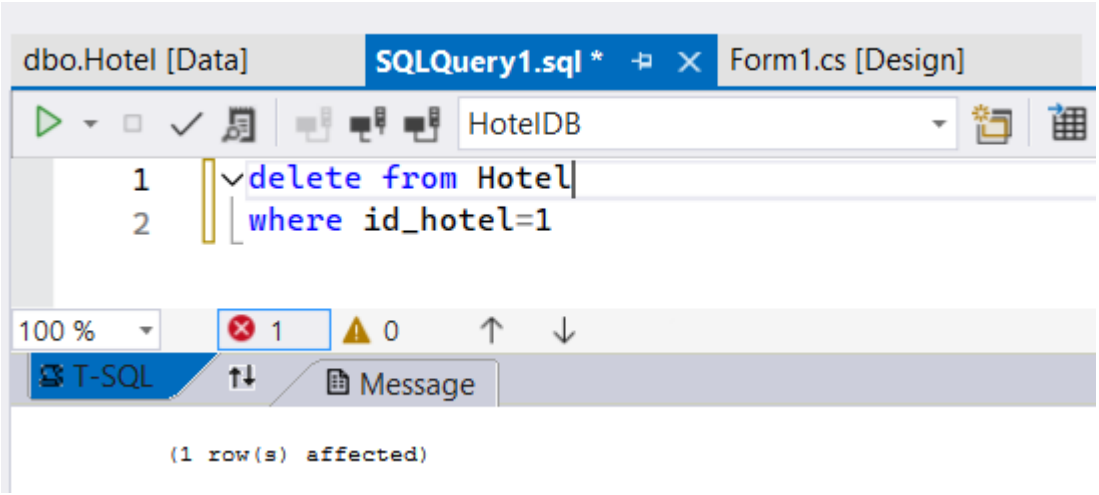
На следната слика е прикажан резултатот од извршениот прашалник за измена на име за хотел со реден број 1.



id_hotel	ime	ulica	telefon	faks	mail	grad
1	Kristijan	Partizanska 23 ...	078456271 ...	070321456 ...	Kiko@gmail.co...	Bitola
NULL	NULL	NULL	NULL	NULL	NULL	NULL

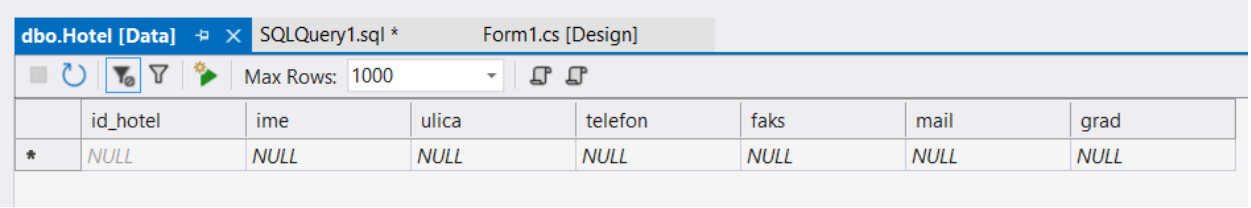
Слика 16. Приказ на табела после извршување на прашаник за измена на податоци

На следната слика е прикажан прашалник за бришење на податоци во табелата Хотел за хотелот со реден број 1.



Слика 17. Приказ на прашалник за бришење на податоци

На следната слика е прикажан резултатот од извршениот прашалник. Бидејќи во табелата имаше само еден податок за хотел со реден број 1 и кога тој податок ќе се избрише таа табела останува празна:

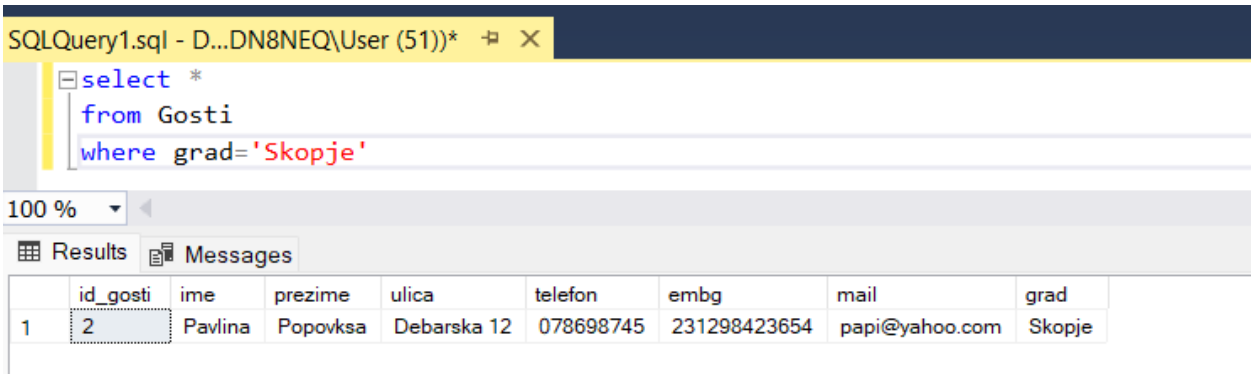


id_hotel	ime	ulica	telefon	faks	mail	grad
NULL	NULL	NULL	NULL	NULL	NULL	NULL

Слика 18. Приказ на табела со податоци после извршување на прашалникот за бришење

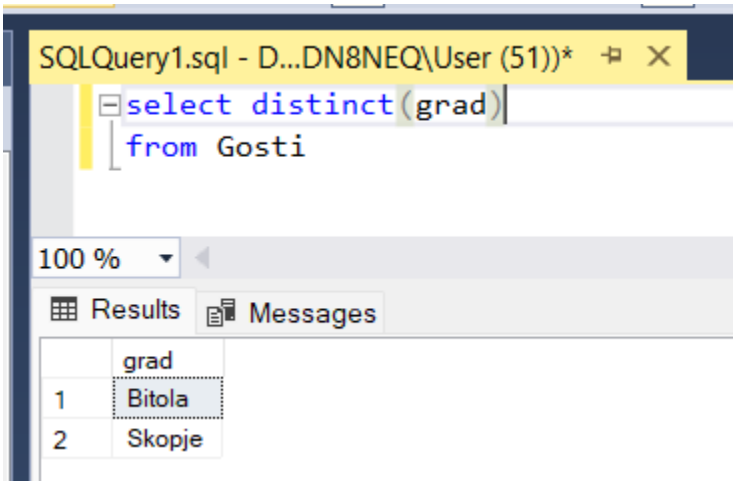
3.3. Извлекување на податоци со SQL прашалник

Извлекување на податоци од база е приказ на податоците кои се запишани во базата според различни услови и потреби од страна на клиентот. На следната слика е прикажан пример за прикажување на сите податоци од табелата Гостин, за гостите кои се од град Скопје.



Слика 19. Приказ на прашаник за извлекување на податоци од табела

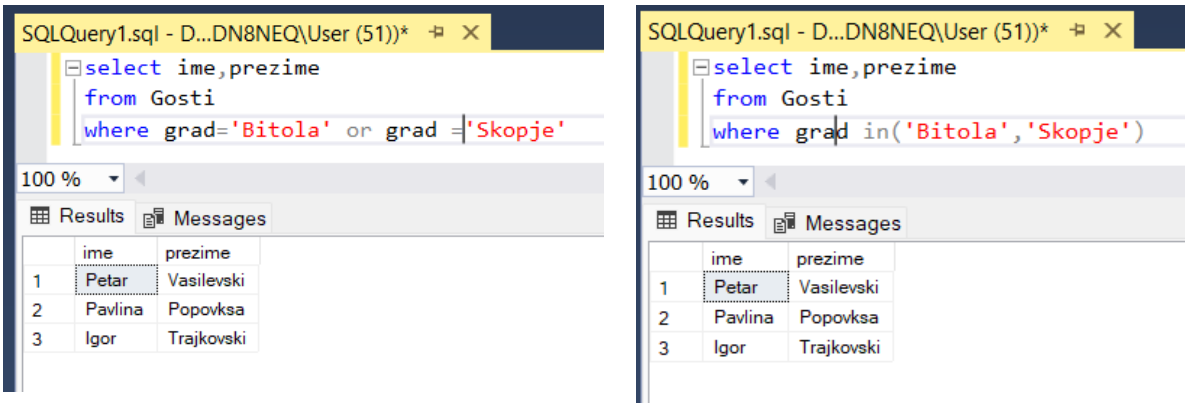
На следната слика се прикажани различните градови од кои припаѓаат гостите.



Слика 20. Приказ на прашаник за извлекување на податоци од табела

На следната слика се прикажани два прашалници кои вршат споредба на две решенија кои нудат еден одговор. На левата слика се врши приказ на податоците име и презиме за табелата Гостин кои се од Битола и Скопје со користење на логички оператор или (or).

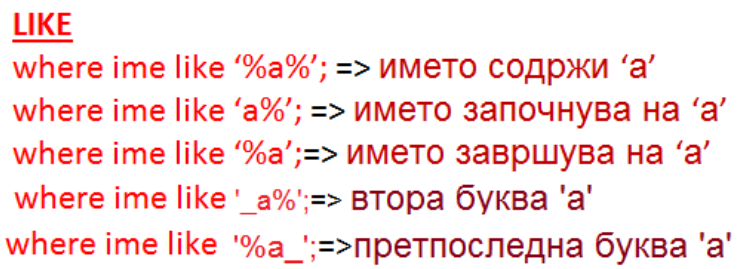
На десната слика се врши приказ на име и презиме за гостите од Битола и Скопје со користење на наредбата (in), односно во било кој град од понудените во листата.



Слика 21. Приказ на прашалник за извлекување на податоци од табела

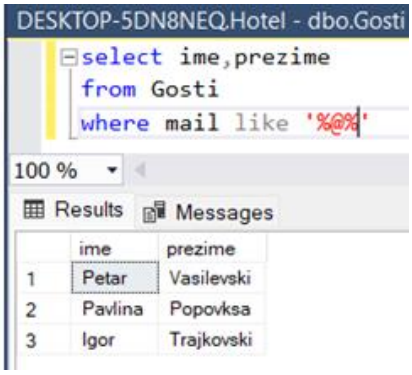
Наредбата Like нуди повеќе можности за пребарување на податоци и тоа содржи даден податок, започнува на даден податок, завршува на даден податок и пребарување на конкретен карактер.

Можностите за користење на наредбата “like” се прикажани на следната слика.



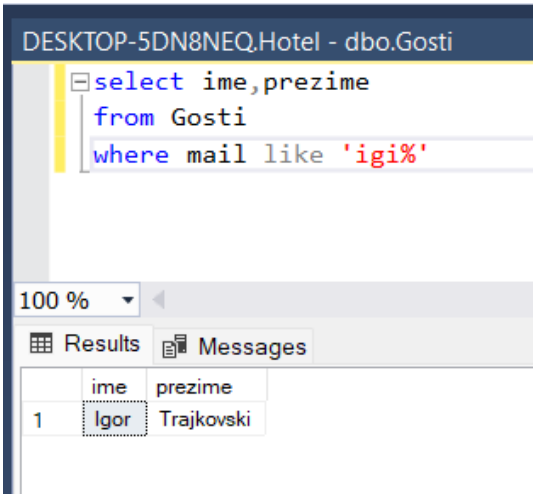
Слика 22. Приказ на синтакса за наредбата LIKE

На следната слика е искористена наредбата Like односно пребарување по даден атрибут дали содржи даден карактер. Прашалникот врши приказ на име и презиме за гостите чии маил содржи ‘@’.



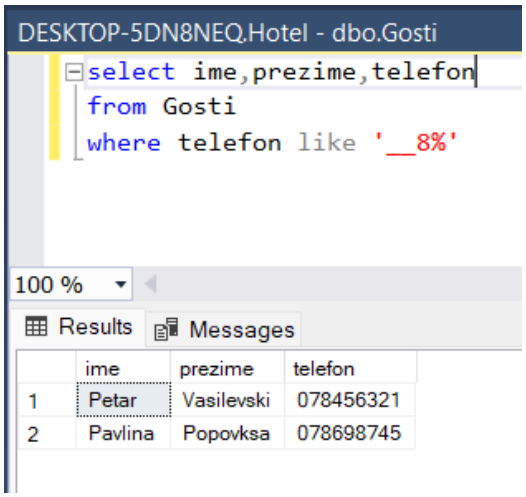
Слика 23. Приказ на прашалник за извлекување на податоци од табела

На следната слика е прикажан прашалникот за приказ на име и презиме за гостите чии маил започнува на ‘igi’.



Слика 24. Приказ на прашалник за извлекување на податоци од табела

На следната слика е прикажан приказ на име, презиме и телефон за гостите чии телефон има трета цифра ‘8’.



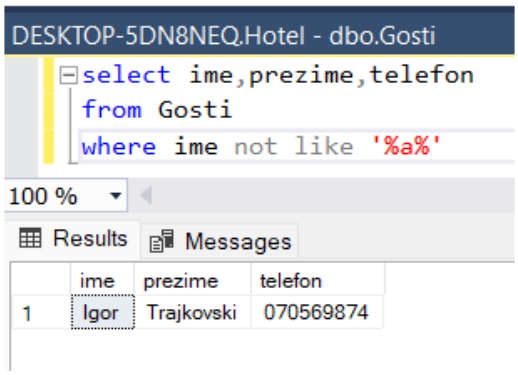
Слика 25. Приказ на прашалник за извлекување на податоци од табела

Наредбата Not Like нуди повеќе можности за пребарување на податоци и тоа не содржи даден податок, не започнува на даден податок, не завршува на даден податок. Можностите за користење на наредбата “Not like” се прикажани на следната слика.

NOT LIKE
where ime not like ‘%a%’; => името не содржи ‘a’
where ime not like ‘a%’; => името не започнува на ‘a’
where ime not like ‘%a’;=> името не завршува на ‘a’

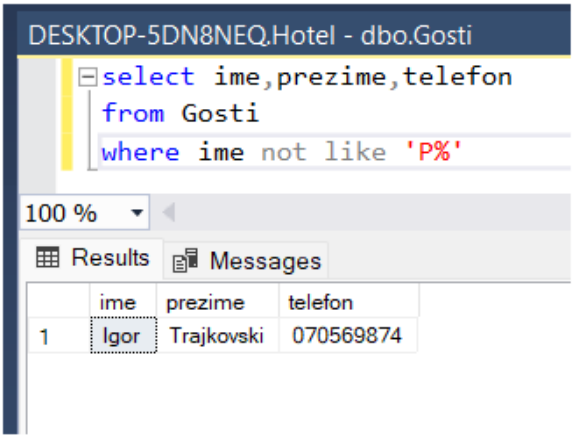
Слика 26. Приказ на синтакса за наредбата NOT LIKE

На следната слика се врши приказ на име, презиме и телефон за гостите чие име не содржи а. Значи Not like е наредба за не започнува, не содржи, не завршува итн.



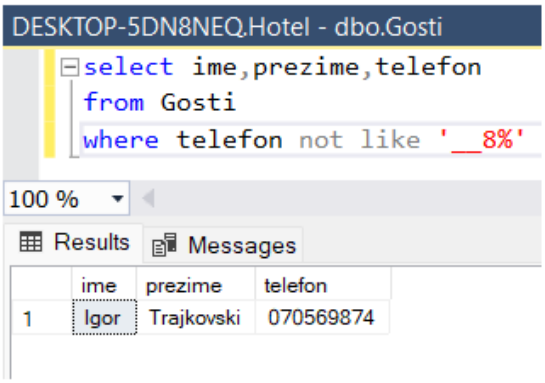
Слика 27. Приказ на прашалник за извлекување на податоци од табела

Да се прикажи име, презиме и телефон за гостите чие име не започнува на ‘P’ е прашалникот кој е прикажан на следната слика.



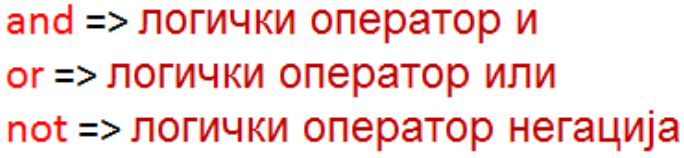
Слика 28. Приказ на прашалник за извлекување на податоци од табела

На следната слика е прикажано: име, презиме и телефон за гостите кои немаат трета цифра ‘8’.



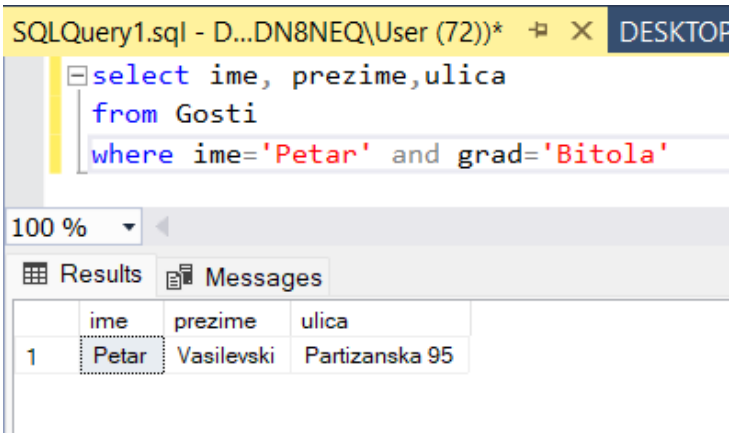
Слика 29. Приказ на прашалник за извлекување на податоци од табела

Логички оператори кои се користат во SQL се and(и), or (или) и not(негација).



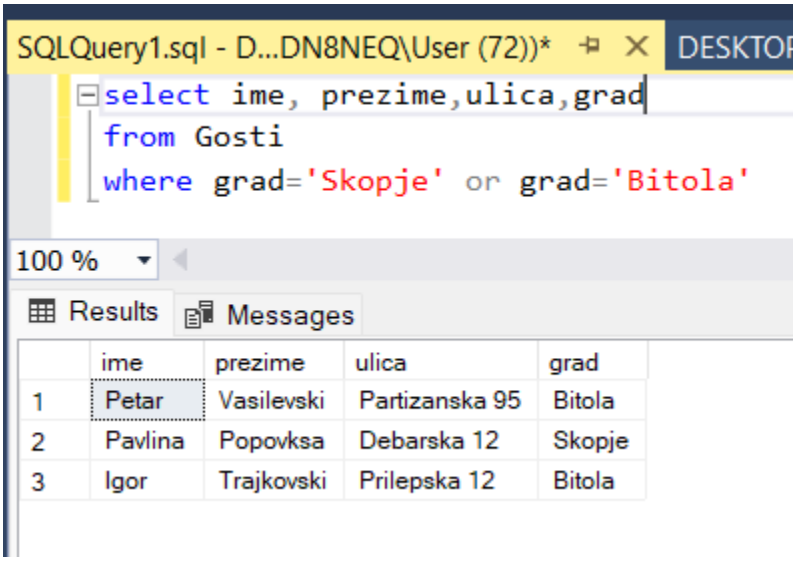
Слика 30. Логички оператори

На следната слика е прикажан пример за прикажување на име, презиме и улица за сите гости кои се со име Петар и се од Битола, каде се врши презентација на логичкиот оператор и (and).



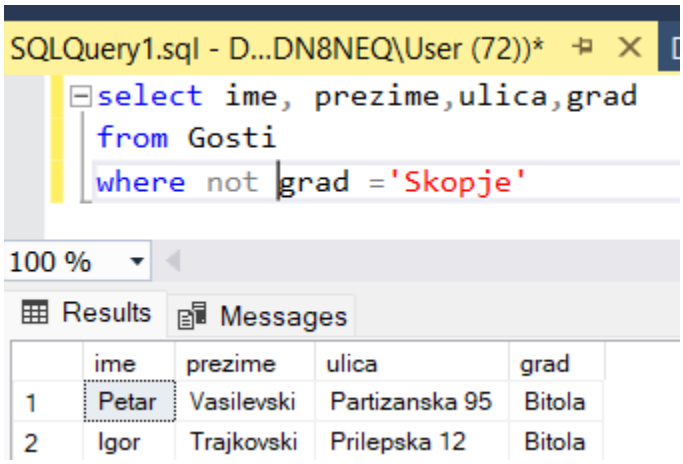
Слика 31. Приказ на прашалник за извлекување на податоци од табела

На следната слика е прикажан пример за прикажување на име, презиме, улица и град за сите гости кои се од Скопје или од Битола, каде се врши презентација на логичкиот оператор или (or).



Слика 32. Приказ на прашалник за извлекување на податоци од табела

На следната слика е прикажан пример за прикажување на име, презиме, улица и град за сите гости кои не се од град Скопје, каде се врши презентација на логичкиот оператор негација(not).



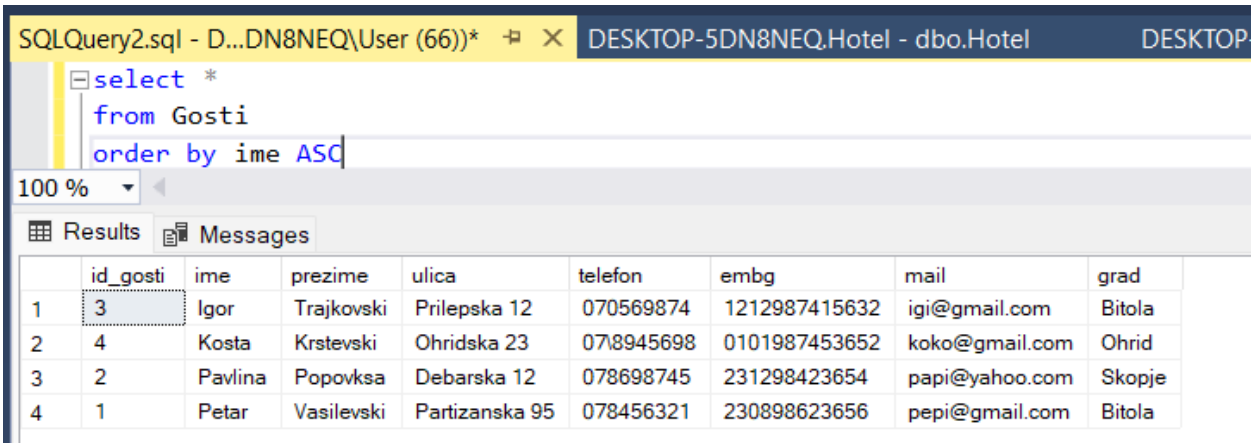
Слика 33. Приказ на прашалник за извлекување на податоци од табела

Во самата работа со бази на податоци потребно е сортирање на податоците за кои се користат и посебни наредби кои нудат можност за сортирање на податоците според даден атрибут во растечки и опаѓачки редослед. ASC е наредба за сортирање на податоците во растечки редослед, додека DESC е наредба за сортирање на податоци во опаѓачки редослед. Order by е главната наредба за сортирање на податоците. Синтаксата за работа е прикажана на следната слика.

```
ORDER BY ime ASC;
ORDER BY ime DESC;
```

Слика 34. Синтакса за видови на сортирање растечки/опаѓачки

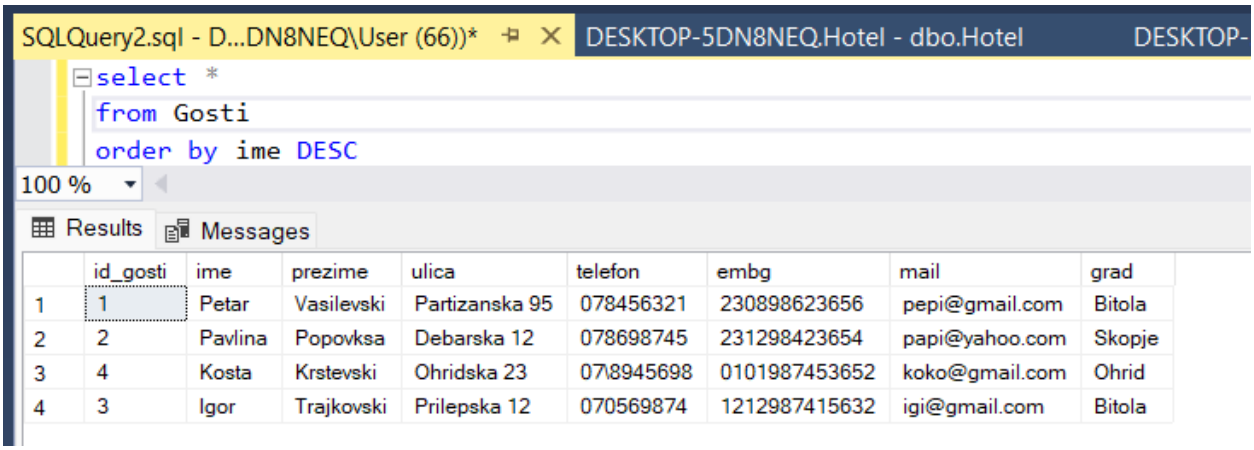
На следната слика се прикажани сите податоци од табелата Гостин кои се сортирани во растечки редослед по име.



	id_gosti	ime	prezime	ulica	telefon	embg	mail	grad
1	3	Igor	Trajkovski	Prilepska 12	070569874	1212987415632	igi@gmail.com	Bitola
2	4	Kosta	Krstevski	Ohridska 23	07\8945698	0101987453652	koko@gmail.com	Ohrid
3	2	Pavlina	Popovksa	Debarska 12	078698745	231298423654	papi@yahoo.com	Skopje
4	1	Petar	Vasilevski	Partizanska 95	078456321	230898623656	pepi@gmail.com	Bitola

Слика 35. Приказ на прашалник за извлекување на податоци од табела

На следната слика се прикажани сите податоци од табелата Гостин кои се сортирани во опаѓачки редослед по име.

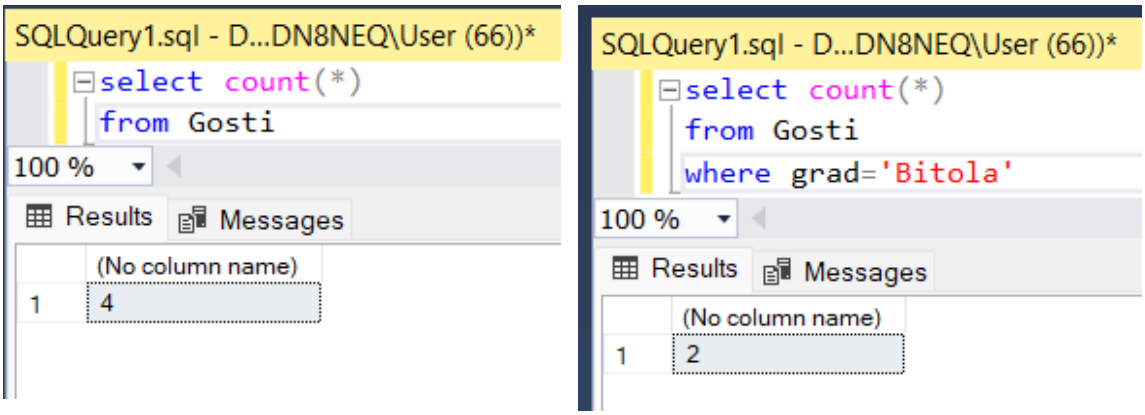


	id_gosti	ime	prezime	ulica	telefon	embg	mail	grad
1	1	Petar	Vasilevski	Partizanska 95	078456321	230898623656	pepi@gmail.com	Bitola
2	2	Pavlina	Popovksa	Debarska 12	078698745	231298423654	papi@yahoo.com	Skopje
3	4	Kosta	Krstevski	Ohridska 23	07\8945698	0101987453652	koko@gmail.com	Ohrid
4	3	Igor	Trajkovski	Prilepska 12	070569874	1212987415632	igi@gmail.com	Bitola

Слика 36. Приказ на прашалник за извлекување на податоци од табела

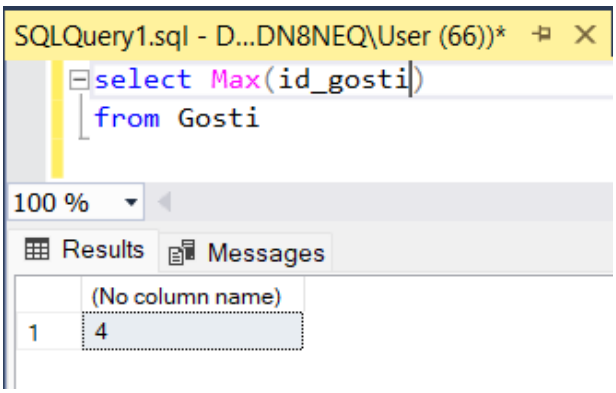
Самиот SQL нуди можност за пресметка на одредени математички функции.

- Count() е функција за одредување на бројот на елементи во дадена табела. На левиот дел од следната слика е прикажан пример за наредбата без дефиниран услов, додека на десниот дел од сликата е прикажана примена на наредбата со услов.
  - Да се прикажи бројот на сите гости;
  - Да се прикаже бројот на сите гости од Битола.



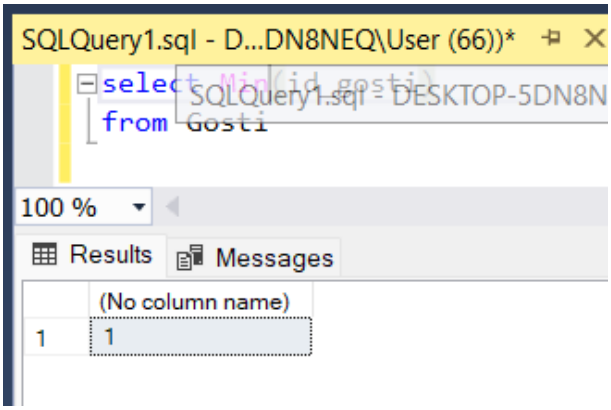
Слика 37. Приказ на прашалник за пресметка

- Max() -е функција за одредување на најголемата вредност од барана колона.



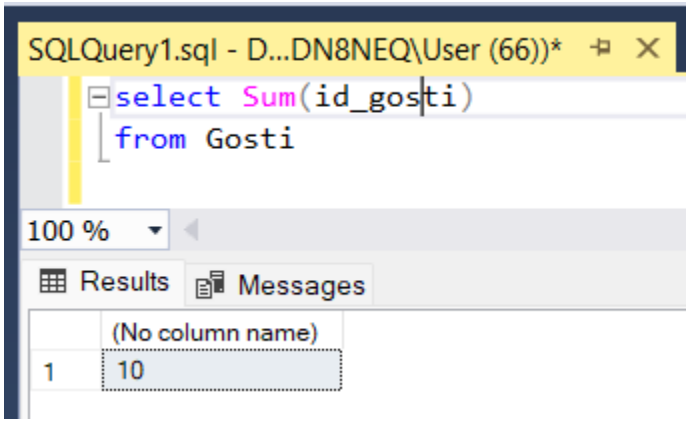
Слика 38. Приказ на прашалник за пресметка

- Min() -е функција за одредување на најмалата вредност од барана колона.



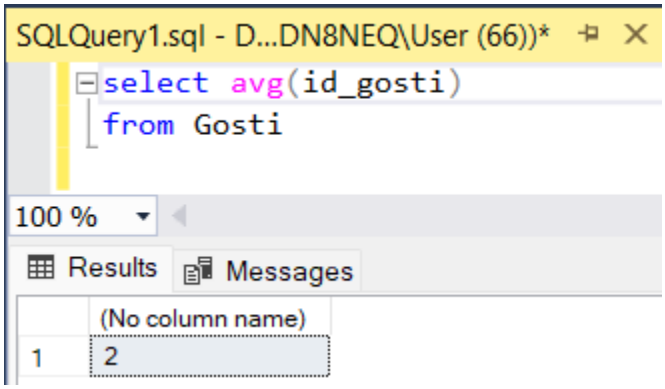
Слика 39. Приказ на прашалник за пресметка

- Sum() – е функција за определување на сумата на податоците од бараната колона.



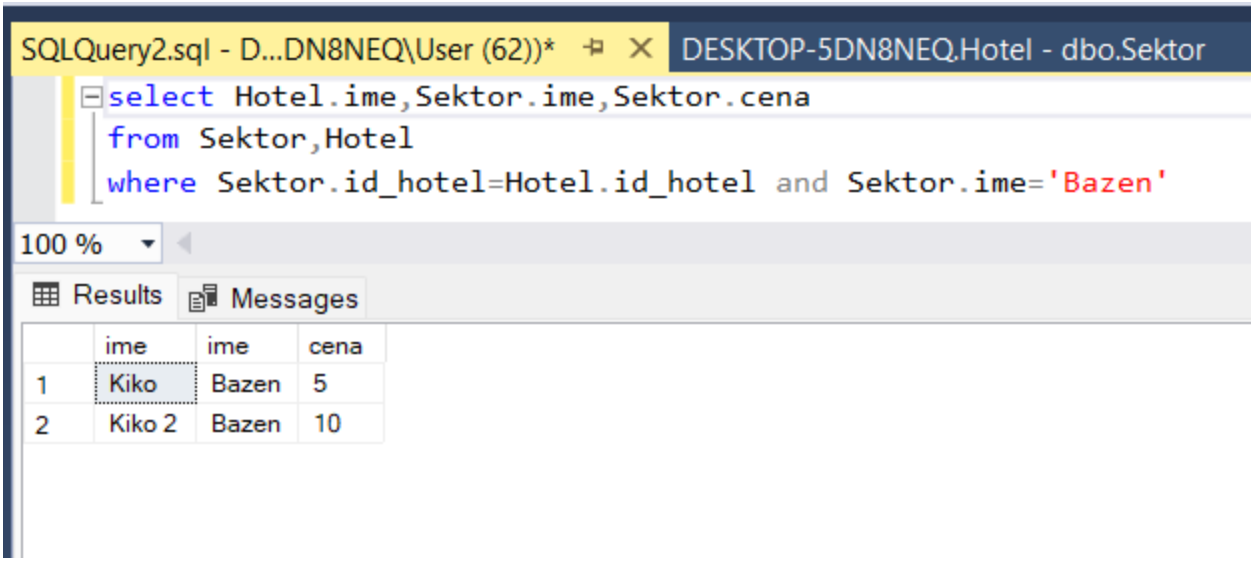
Слика 40. Приказ на прашалник за пресметка

- Avg() – е функција за определување на просек, средна вредност или аритметичка средина на податоци од барана колона.



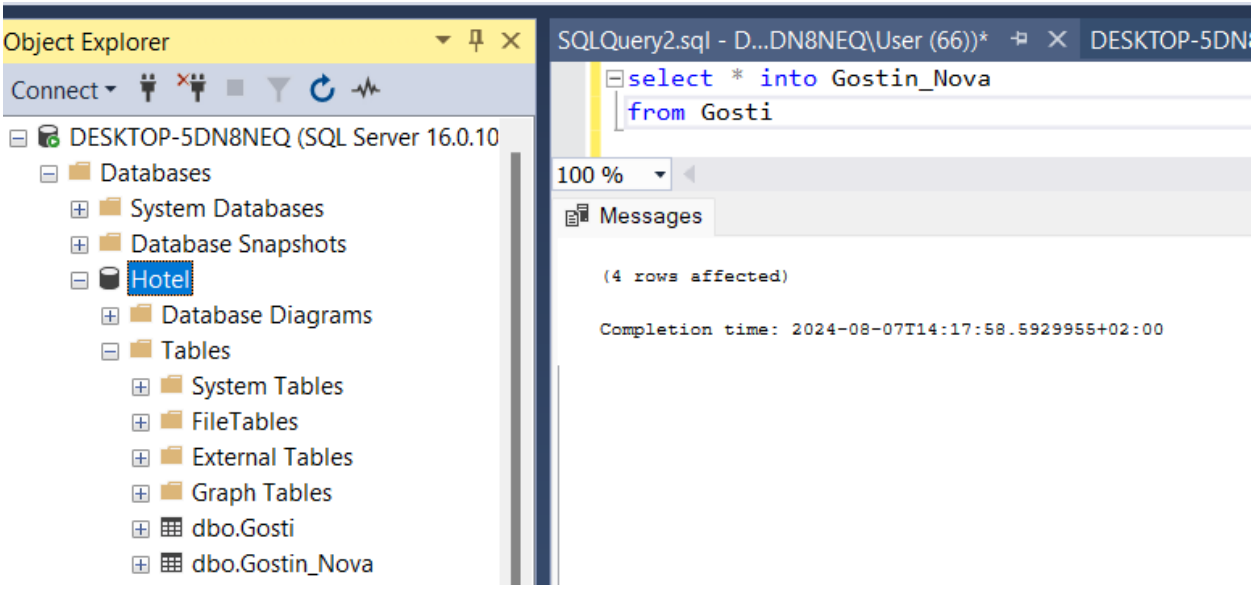
Слика 41. Приказ на прашалник за пресметка

На следната слика е презентиран пример со користење на повеќе од една табела. Пример, се прикажуваат име на хотел, име на сектор, и цена за тој сектор, за секторот 'Bazen'. Како најбитен процес во работа со повеќе од една табела е поврзувањето на табелите и пристапот до секој атрибут. Пристапот до секој атрибут се врши со името на ентитетот (табелата) точка пристапот до потребен атрибут. Додека поврзувањето на табелите е преку нивните примарни и секундарни клучеви.



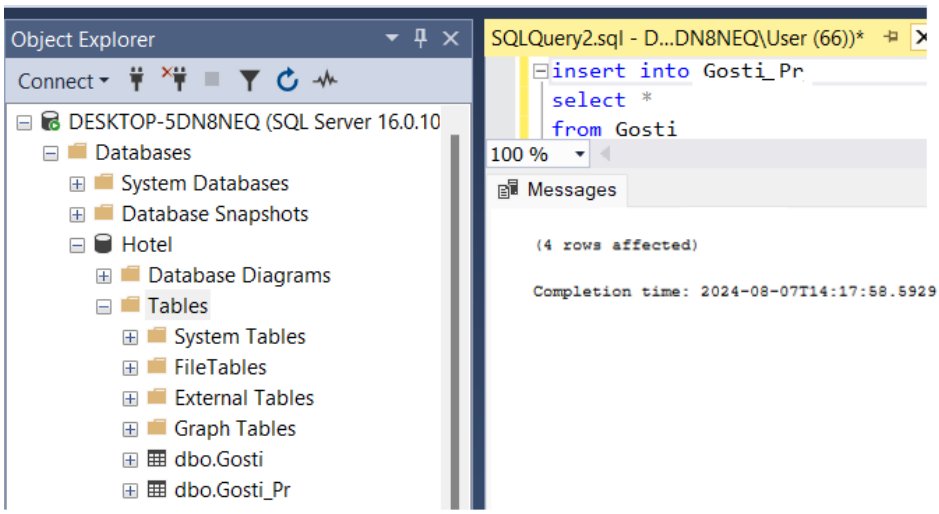
Слика 42. Приказ на прашалник за читање на податоци од повеќе табели

На следната слика е прикажан прашалник за копирање на сите податоци од табелата Гостин како и нејзино креирање.



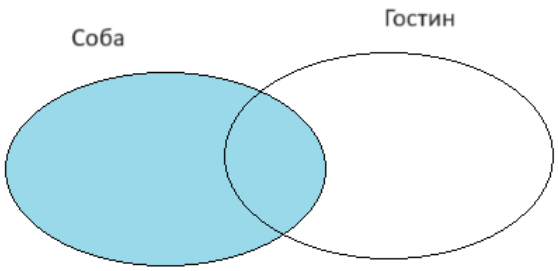
Слика 43. Приказ на прашалник за копирање на податоци и креирање нова табела

Ако табелата е веќе креирана и е празна, потребно е само копирање на податоци од постоечка табела, тогаш прашалникот е различен и истиот е прикажан на следната слика.



Слика 44. Приказ на прашалник за копирање на податоци во друга табела

Постојат повеќе видови на спојување на табели според заедничко поле на дизајн на организација на самата табела. Тоа претставува приказ на сите податоци од табелата лево и нивниот заеднички пресек.



Слика 45. Приказ на лево поврзување на податоци

SQLQuery2.sql - D...DN8NEQ\User (66))*

```
select *
from Soba as S left join Hotel as H on s.id_hotel=h.id_hotel
```

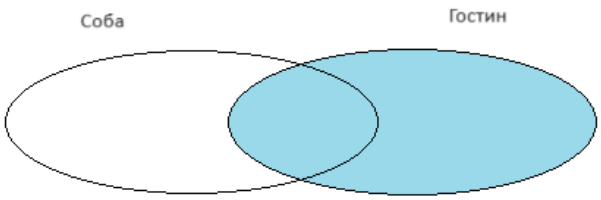
100 %

Results | Messages

	id_soba	id_hotel	tip	id_hotel	ime	ulica	telefon	faks	mail	grad
1	1	1	Vip	1	Kiko	Prikepska 67	070698452	078456213	kiko@gmail.com	Bitola
2	2	1	Dvosobna	1	Kiko	Prikepska 67	070698452	078456213	kiko@gmail.com	Bitola
3	3	1	Trosobna	1	Kiko	Prikepska 67	070698452	078456213	kiko@gmail.com	Bitola
4	4	2	Vip	2	Kiko 2	Partizanska 12	078459876	070564789	kiko@gmail.com	Skopje
5	5	1	Studio	1	Kiko	Prikepska 67	070698452	078456213	kiko@gmail.com	Bitola
6	6	2	Dvosobna	2	Kiko 2	Partizanska 12	078459876	070564789	kiko@gmail.com	Skopje
7	8	1	Bazen	1	Kiko	Prikepska 67	070698452	078456213	kiko@gmail.com	Bitola

Слика 46. Приказ на решение од лево поврзување на податоци

На следната слика е прикажано десно спојување на две табели со заеднички атрибут.



Слика 47. Приказ на десно поврзување на податоци

SQLQuery2.sql - D...DN8NEQ\User (66))*

```
select *
from Soba as S right join Hotel as H on s.id_hotel=h.id_hotel
```

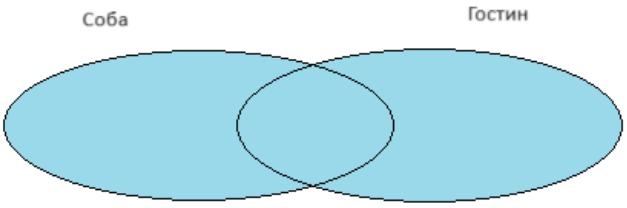
100 %

Results | Messages

	id_soba	id_hotel	tip	id_hotel	ime	ulica	telefon	faks	mail	grad
1	1	1	Vip	1	Kiko	Prikepska 67	070698452	078456213	kiko@gmail.com	Bitola
2	2	1	Dvosobna	1	Kiko	Prikepska 67	070698452	078456213	kiko@gmail.com	Bitola
3	3	1	Trosobna	1	Kiko	Prikepska 67	070698452	078456213	kiko@gmail.com	Bitola
4	5	1	Studio	1	Kiko	Prikepska 67	070698452	078456213	kiko@gmail.com	Bitola
5	8	1	Bazen	1	Kiko	Prikepska 67	070698452	078456213	kiko@gmail.com	Bitola
6	4	2	Vip	2	Kiko 2	Partizanska 12	078459876	070564789	kiko@gmail.com	Skopje
7	6	2	Dvosobna	2	Kiko 2	Partizanska 12	078459876	070564789	kiko@gmail.com	Skopje

Слика 48. Приказ на решение од поврзување на податоци

На следната слика е прикажано внатрешно спојување на две табели, тоа се податоците од двете табели споени.



SQLQuery2.sql - D...DN8NEQ\User (66))*

```
select *
from Soba as S inner join Hotel as H on s.id_hotel=h.id_hotel
```

100 %

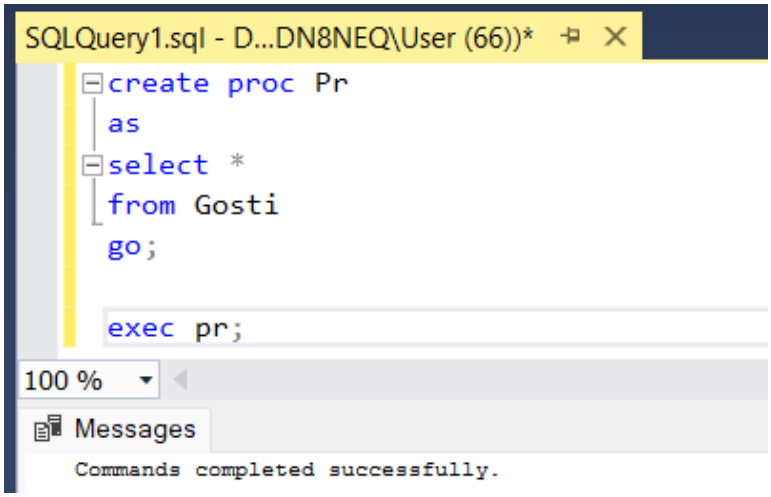
Results | Messages

	id_soba	id_hotel	tip	id_hotel	ime	ulica	telefon	faks	mail	grad
1	1	1	Vip	1	Kiko	Prikepska 67	070698452	078456213	kiko@gmail.com	Bitola
2	2	1	Dvosobna	1	Kiko	Prikepska 67	070698452	078456213	kiko@gmail.com	Bitola
3	3	1	Trosobna	1	Kiko	Prikepska 67	070698452	078456213	kiko@gmail.com	Bitola
4	4	2	Vip	2	Kiko 2	Partizanska 12	078459876	070564789	kiko@gmail.com	Skopje
5	5	1	Studio	1	Kiko	Prikepska 67	070698452	078456213	kiko@gmail.com	Bitola
6	6	2	Dvosobna	2	Kiko 2	Partizanska 12	078459876	070564789	kiko@gmail.com	Skopje
7	8	1	Bazen	1	Kiko	Prikepska 67	070698452	078456213	kiko@gmail.com	Bitola

Слика 49. Приказ на внатрешно поврзување на податоци и решение од прашалникот

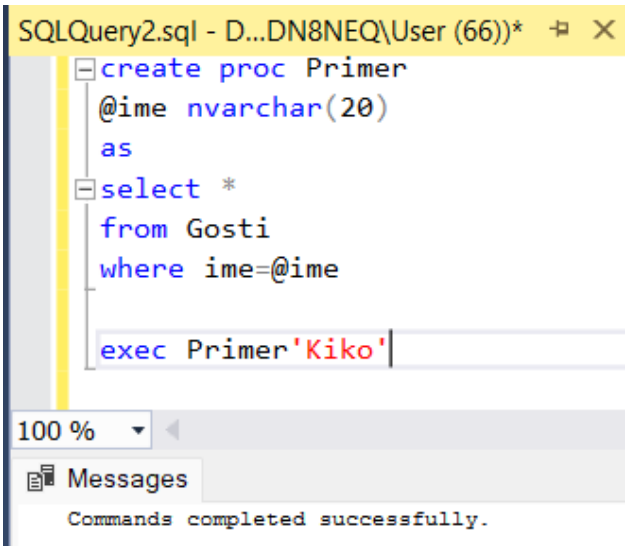
3.4. Процедури и функции во SQL Server

Сторираните процедури се посебни функции кои се користат и тоа еднаш се дефинираат, а се повикуваат онолку пати колку што има потреба. Секоја процедура се креира со клучните зборови ‘create proc’, потоа се запишува името на процедурата. На следната слика е креирана процедура која не прима влезен аргумент и врши приказ на сите податоци од табелата Gostin. Клучниот збор за повик на процедурата во SQL е exec и името на процедурата.



Слика 50. Приказ на прашалник за процеура

На следната слика е прикажан пример за процедура која прима влезен аргумент за измена на пребарување по име. Со помош на знакот ‘@’ се декларира влезниот аргумент на процедурата и потоа истиот се користи за потребите на самата процедура.

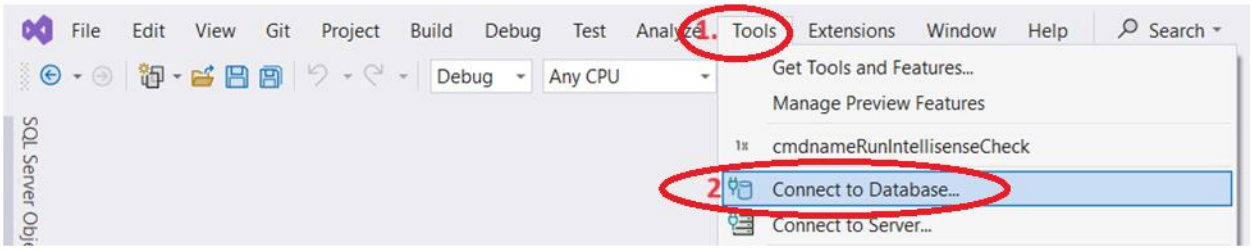


Слика 51. Приказ на прашалник за процедура со влезен параметар

4. РАЗВОЈ НА SQL SERVER АПЛИКАЦИИ ВО VISUAL STUDIO

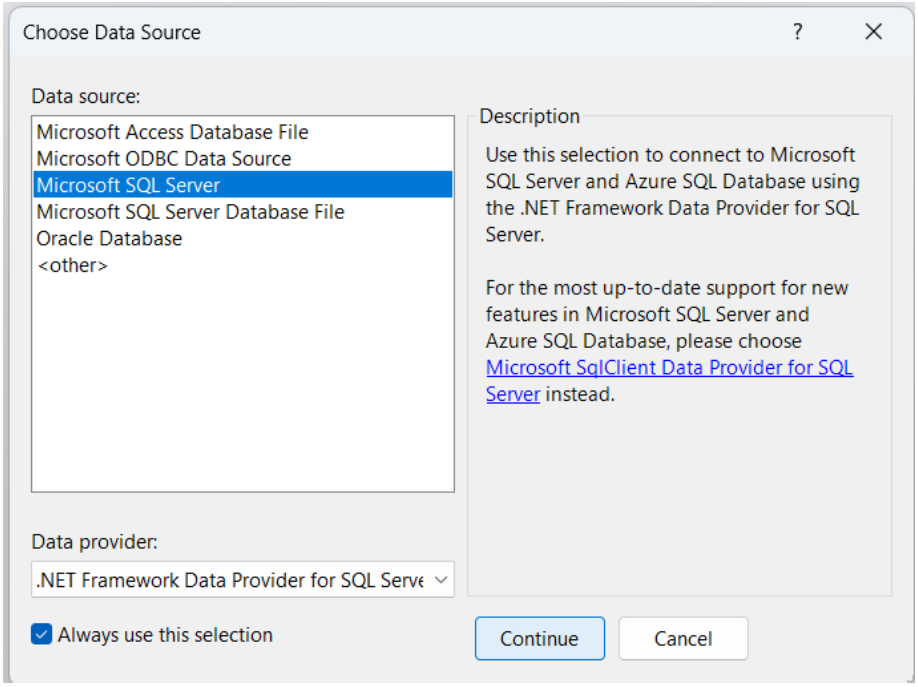
4.1. Конекција на апликација со база на податоци

При креирање на апликацијата потребна е конекција до SQL серверот за целосна функционалност на апликацијата и работа со податоци од базата на податоци. На следната слика е прикажана конекцијата на апликацијата со серверот и базата на податоци. Од менито обележано со реден број 1 се избира Tools и со реден број 2 обележано на сликата се избира Connect to Database.



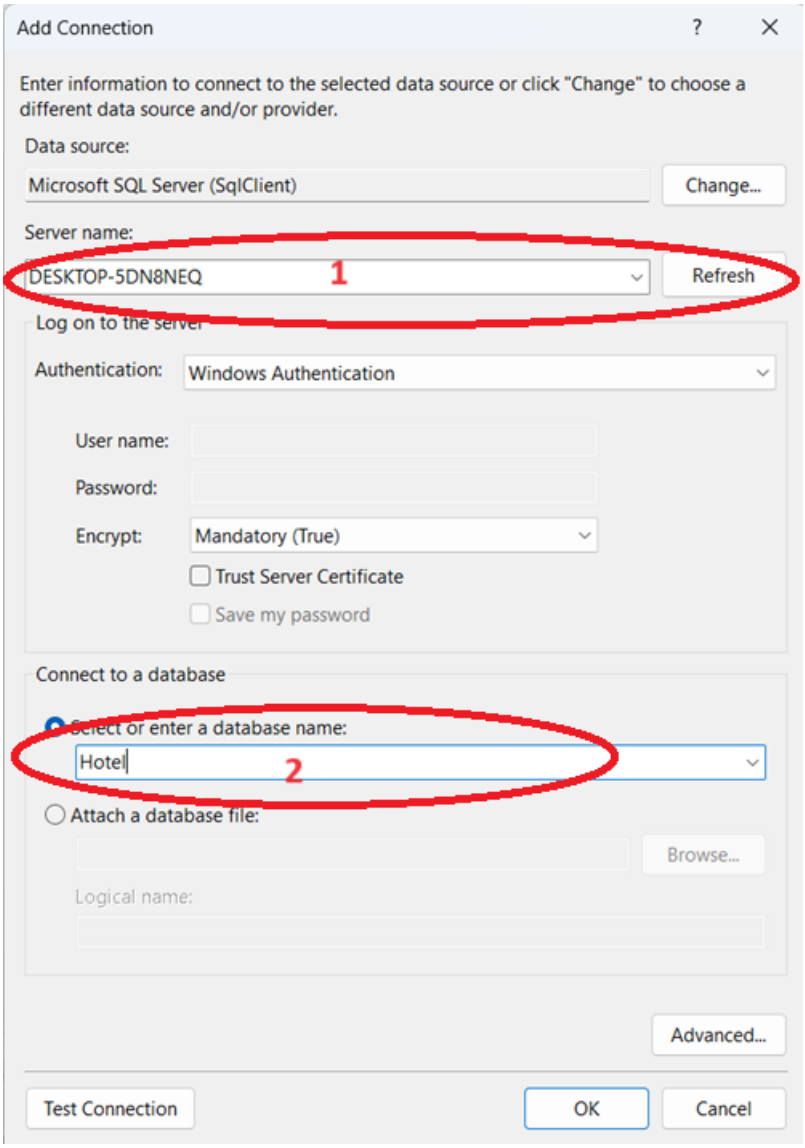
Слика 52. Приказ за конекција на базата на податоци

На следната слика е прикажан избор на типот на серверот и неговото користење.



Слика 53. Приказ за избор на сервер

По изборот на типот на сервер на екран се прикажува прозорец во кој е потребно да се избере името на серверот и базата на податоци која ќе се користи во апликацијата. Со реден број 1 е обележано името на серверот и со реден број 2 е обележана базата на податоци.

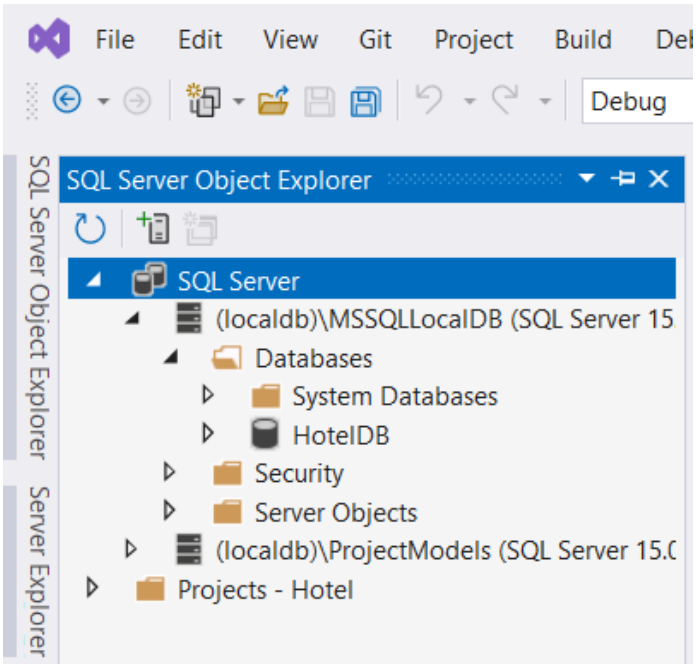


Слика 54. Приказ за избор на база

Откако ќе се изврши оваа постапка со конекција на апликацијата до базата се добива технички веќе конектирана апликација со база.

```
("Data Source=DESKTOP-5DN8NEQ;Initial Catalog=Hotel;Integrated Security=True")
```

На следната слика е прикажана базата и серверот кои се веќе видливи од Visual Studio-то каде се гради апликацијата.



Слика 55 Приказ за извршената конекција

4.2. Извршување на SQL прашалници во кодот

На следната слика е прикажан прашалник кој се извршува во апликацијата за логирање на вработен во Хотелот.

- Sqlc – е објект од класата SqlConnection кој се користи за конекцијата на апликацијата со базата на податоци која се наоѓа на Sql Server-от.
- Query – е променлива од тип string, во која се наоѓа како вредност, прашалникот за контрола на корисничко име и лозинка.
- sData – е објект од класата SqlDataAdapter, каде се поврзуваат конекцијата со базата и прашалникот кој е потребно да се изврши.
- dTab – објект од класата DataTable. Со помош на објектот се повикува функцијата Fill за извршување на прашалникот и чување на податоците од решението.

- Со конторлата if се врши проверка дали постои корисник со податоци кои се внесени во формата од апликацијата.

```
public void fLogin(Object sender,EventArgs e)
{
    SqlConnection sqlc = new SqlConnection("Data Source=DESKTOP-5DN8NEQ;Initial Catalog=Hotel;Integrated Security=True");
    string query = "select * from Vraboten where  korisnicko_ime = '"+TBKoriscnickoIme.Text +"' and lozinka='"+TBLozinka.Text+"' ";
    SqlDataAdapter sdata = new SqlDataAdapter(query, sqlc);
    DataTable dtab = new DataTable();
    sdata.Fill(dtab);
    if (dtab.Rows.Count > 0)
    {
        this.Hide();
        MeniIzbor mn = new MeniIzbor();
        mn.Show();
    }

}
```

Слика 56. Приказ на извршен код за логирање

На следната слика се врши приказ на прашалник кој е извршен во апликацијата за внесување на податоци за Вработен. Табелата на сервер за Вработен се состои од следните атрибути: име, презиме, корисничко име, лозинка, улица, телефон, матичен број и маил за вработениот. За сите атрибути во формата се креирани текст полиња, каде корисникот има можност да изврши внесување на податоци. Податоците од формата се преземаат преку објектот sql, кој е креиран од класата SqlCommand и параметарот Parametars, за секој атрибут одделно според типот на податок. Со помош на функцијата ExecuteNonQuery, која е повикана за објектот sql, се врши извршување на прашалникот и автоматски внесување на податоците во табелата Вработен во базата на податоци.

```
SqlConnection konekcija = new SqlConnection("Data Source=DESKTOP-5DN8NEQ;Initial Catalog=Hotel;Integrated Security=True");
konekcija.Open();
SqlCommand sql = new SqlCommand("insert into Vraboten(ime,prezime,korisnicko_ime,lozinka,ulica,telefon,embg,mail) " +
    "             values(@TBIme,@TBPrezime,@TBKorisnickoIme,@TBLozinka,@TBUlica,@TBTelefon,@TBEmbg,@TBMail)",konekcija);
sql.Parameters.AddWithValue("@TBIme", SqlDbType.NChar).Value = TBIme.Text;
sql.Parameters.AddWithValue("@TBPrezime", SqlDbType.NChar).Value = TBPrezime.Text;
sql.Parameters.AddWithValue("@TBKorisnickoIme", SqlDbType.NChar).Value = TBKoriscnickoIme.Text;
sql.Parameters.AddWithValue("@TBLozinka", SqlDbType.NChar).Value = TBLozinka.Text;
sql.Parameters.AddWithValue("@TBUlica", SqlDbType.NChar).Value = TBUlica.Text;
sql.Parameters.AddWithValue("@TBTelefon", SqlDbType.NChar).Value = TBTelefon.Text;
sql.Parameters.AddWithValue("@TBEmbg", SqlDbType.NChar).Value = TBEmbg.Text;
sql.Parameters.AddWithValue("@TBMail", SqlDbType.NChar).Value = TBMail.Text;

sql.ExecuteNonQuery();
```

Слика 57. Приказ на извршен код за внесување на податоци

Во апикацијата се искористени и прашалници за внесување на податоци со претходни решенија од веќе извршени прашалници во зависност од изборот на корисникот. За да се добие вкупната цена за бројот на денови кои ги внесува корисникот,

потребно е да се добие разликата помеѓу дата за почеток на резервација и датата за крај на резервација. Во апликацијата се внесуваат: дата за почеток на резервацијата и дата за крај на резервацијата. Со помош на прашалникот се добива разликата од тие две дати и се креира втор прашалник кој ја дава цената за избраната соба. Производот помеѓу овие две решенија ја дава вкупната сума, која е потребна да се плати за бараната резервација. Тоа решение се запишува во текст поле во кое корисникот не може да внесе, но самата апликација го пополнува тоа поле автоматски со пресметка. Со помош на функцијата ExecuteScalar() се добива решение од прашалник кој се извршува.

```
private void fPresmetaj(object sender, EventArgs e)
{
    konekcija.Open();
    SqlCommand sql = new SqlCommand("SELECT DATEDIFF(DAY,@TBOD, @TBDO)  from SobaGostin", konekcija);

    sql.Parameters.AddWithValue("@TBOD", SqlDbType.NChar).Value = tbDataOd.Text;
    sql.Parameters.AddWithValue("@TBDO", SqlDbType.NChar).Value = tbDataDo.Text;


    //se opredeluva brojot da denovi za rezervacija
    int x = (int)sql.ExecuteScalar();

    SqlCommand sql1 = new SqlCommand("SELECT cena from Soba where id_soba=@TBSoba", konekcija);
    //sql1 za koristenje na vtoriot prasalnik
    sql1.Parameters.AddWithValue("@TBSoba", SqlDbType.NChar).Value = tbSoba.Text;
    //se opredleuva cenata za eden den vo taa soba
    int y = (int)sql1.ExecuteScalar();

    int vcena = x * y;
    tbVcena.Text = vcena.ToString();
    konekcija.Close();
}
```

Слика 58. Приказ на извршен код за пресметка на вкупна цена

На следната слика е прикажан прашалник за бришење на податоци според избрана вредност од табелата, за приказ и повторно приказ на податоците со избришана редица од табелата.

```
konekcija.Open();
SqlCommand sql = new SqlCommand("delete from Sektor where id_sektor=@TBRedenBroj", konekcija);
sql.Parameters.AddWithValue("@TBRedenBroj", SqlDbType.NChar).Value = tbRedenBroj.Text;

sql.ExecuteNonQuery();



dataAdapter = new SqlDataAdapter("select Sektor.id_sektor as 'Реден број ',Hotel.ime as " +
    "'Име хотел ', Hotel.grad as 'Град Хотел', Sektor.ime as 'Име сектор', Sektor.cena as " +
    "'Цена ' \r\nfrom Sektor, Hotel\r\nwhere Sektor.id_hotel=Hotel.id_hotel", konekcija);
SqlCommandBuilder comm1 = new SqlCommandBuilder(dataAdapter);
DataTable table1 = new DataTable();
table1.Locale = System.Globalization.CultureInfo.InvariantCulture;
dataAdapter.Fill(table1);
tabela.ReadOnly = true;
sourse.DataSource = table1;
konekcija.Close();
Controls.Add(tabela);
```

Слика 59. Приказ на извршен код за бришење на податоци табела

На следната слика е прикажан прашалник за измена на податоци според избрана вредност од табелата за приказ и повторно приказ на табелата со изменета вредност според критериумите од дефинираниот избор за измена.

```
konekcija.Open();
SqlCommand sql = new SqlCommand("update soba set tip=@TBNovo where id_soba=@TBRedenBroj", konekcija);
sql.Parameters.AddWithValue("@TBRedenBroj", SqlDbType.NChar).Value = tbRedenBroj.Text;
sql.Parameters.AddWithValue("@TBNovo", SqlDbType.NChar).Value = tbNovo.Text;

sql.ExecuteNonQuery();

dataAdapter = new SqlDataAdapter("select Soba.id_soba as 'Реден број' ,Hotel.ime as 'Име' ," +
"Hotel.grad as 'Град', Soba.tip as 'Тип ',Soba.cena as 'Цена ' from Soba,Hotel where " +
"Soba.id_hotel=Hotel.id_hotel", konekcija);
SqlCommandBuilder comm1 = new SqlCommandBuilder(dataAdapter);
DataTable table1 = new DataTable();
table1.Locale = System.Globalization.CultureInfo.InvariantCulture;
dataAdapter.Fill(table1);
tabela.ReadOnly = true;
sourse.DataSource = table1;
konekcija.Close();
Controls.Add(tabela);
```

Слика 60. Приказ на извршен код за измена на податоци и приказ на измена

5. ДИЗАЈН И РАЗВОЈ НА КОРИСНИЧКИ ИНТЕРФЕЈС

5.1. Развој на кориснички интефејс

При креирање и развој на апликацијата се искористени класи кои се готови во Visual Studio-то и е извршена примена на својствата за секоја од класите.

Пристапот до дизајнот за формата која се креира се врши со клучниот збор this. Искористени се следните својства:

- Позадинска боја (this.BackColor = Color.Gray)
- Исклучување на Minimize операцијата (this.MaximizeBox = false)
- Исклучување на иконата од формата (this.ShowIcon = false)
- Дефинирање на текст на формата како наслов
- Дефинирање на ширина на формата (800)
- Дефинирање на висина на формата (400)

```
this.BackColor = Color.Gray;
this.ShowIcon = false;
this.Text = "К И К О  -  Х О Т Е Л ";
this.MinimizeBox = false;
this.Width = 800;
this.Height = 400;
```

Слика 61. Приказ на извршен код, форма

Label - Креиран е објект од класата лабела со име LNaslov. За лабелата се искористени својствата: текст, должина, ширина, фонт, локација, каде е позицијата на лабелата на самата форма и додавање на лабелата на формата.

```
Label lNAslov = new Label();
lNAslov.Text = "     К И К О  -  Х О Т Е Л     ";
lNAslov.Width = 400;
lNAslov.Height = 50;
lNAslov.Font = new System.Drawing.Font(lNAslov.Name, 25F);
lNAslov.Location = new Point(180, 30);
this.Controls.Add(lNAslov);
```

Слика 62. Приказ на извршен код за лабела

На следната слика е прикажано користење на објект од класата TextBox со име TBKorisnikoIme за кое се користат својствата кои се прикажани на сликата.

```
TextBox TBKoriscnickoIme = new TextBox();
TBKoriscnickoIme.Width = 200;
TBKoriscnickoIme.Height = 60;
TBKoriscnickoIme.Font = new System.Drawing.Font(TBKoriscnickoIme.Name, 15F);
TBKoriscnickoIme.Location = new Point(410, 100);
this.Controls.Add(TBKoriscnickoIme);
```

Слика 63. Приказ на извршен код за текст поле

На следната слика е прикажано користењето на објектот од класата Button со име tbLogin. За овој објект има дополнително својство за клик на копчето. Со клик на копчето се врши повик на функцијата со име fLogin која е претходно дефинирана.

```
Button btLogin = new Button();
btLogin.Width = 100;
btLogin.Height = 50;
btLogin.Text = " Влез ";
btLogin.Font = new System.Drawing.Font(btLogin.Name, 15F);
btLogin.Location = new Point(410, 200);
btLogin.Click += new EventHandler(this.fLogin);
this.Controls.Add(btLogin);
```

Слика 64. Приказ на извршен код за копче

На следната слика е прикажан дел од креирање на мени за избор, со свои компоненти, кои ги нуди самата апликација за корисникот.

```
MainMenu meni = new MainMenu();

MenuItem Hotel = new MenuItem();
Hotel.Text = "Хотел";
meni.MenuItems.Add(Hotel);

    MenuItem vnesiH = new MenuItem();
    vnesiH.Text = "Внеси";
    vnesiH.Click += new EventHandler(this.fVnesiHotel);
    Hotel.MenuItems.Add(vnesiH);

    MenuItem izbrisiH = new MenuItem();
    izbrisiH.Text = "Избриши";
    izbrisiH.Click += new EventHandler(this.fIzbrisiHotel);
    Hotel.MenuItems.Add(izbrisiH);

    MenuItem izmeniH = new MenuItem();
    izmeniH.Text = "Измени";
    izmeniH.Click += new EventHandler(this.fIzmeniHotel);
    Hotel.MenuItems.Add(izmeniH);


        MenuItem Soba = new MenuItem();
        Soba.Text = "Соба";
        meni.MenuItems.Add(Soba);

            MenuItem vnesiS = new MenuItem();
            vnesiS.Text = "Внеси";
            vnesiS.Click += new EventHandler(this.fVnesiSoba);
            Soba.MenuItems.Add(vnesiS);

            MenuItem izbrisiS = new MenuItem();
            izbrisiS.Text = "Избриши";
            izbrisiS.Click += new EventHandler(this.fIzbrisiSoba);
            Soba.MenuItems.Add(izbrisiS);

            MenuItem izmeniS = new MenuItem();
            izmeniS.Text = "Измени";
            izmeniS.Click += new EventHandler(this.fIzmeniSoba);
            Soba.MenuItems.Add(izmeniS);
```

Слика 65. Приказ на извршен код за мени со повеќе избори и подмени

За секој избор од менито се креирани функции кои се повикуваат со нивни избор, односно клик на самата можност за избор од менито.

```
public void fIzmeniHotel(Object sender, EventArgs e)
{
    IzmeniHotel ih = new IzmeniHotel();
    ih.Show();
    //this.Close();
}
1 reference
public void fIzbrisiHotel(Object sender, EventArgs e)
{
    IzbrisiHotel ih = new IzbrisiHotel();
    ih.Show();
    //this.Close();
}
```

Слика 66. Приказ на извршен код за функции со креирање на објект

На следната слика е прикажано користење на објект од класата PictureBox кој се користи за приказ на слика која се чита директно од локација.

```
PictureBox primer = new PictureBox();
primer.Width = 760;
primer.Height = 320;
primer.Location = new Point(10, 10);
primer.BorderStyle = System.Windows.Forms.BorderStyle.Fixed3D;
primer.SizeMode = PictureBoxSizeMode.StretchImage;
primer.ImageLocation = "C://Users//User//source//repos//Hotel//Hotel//primer.jpg";
Controls.Add(primer);
```

Слика 67. Приказ на извршен код за слика

На следната слика е прикажано користење на табеларен запис на податоци со повлекување на податоци од база на податоци и директен дизајн на самата табела за приказ.

```
DataGridView tabela = new DataGridView();
BindingSource sourse = new BindingSource();
SqlDataAdapter dataAdapter = new SqlDataAdapter();

tabela.Width = 730;
tabela.Height = 130;
tabela.Location = new Point(20, 120);
tabela.DataSource = sourse;

tabela.BackgroundColor = Color.LightGray;
tabela.ForeColor = Color.Gray;
tabela.Font = new Font("Ariel", 10.0f, FontStyle.Bold);


konekcija.Open();
dataAdapter = new SqlDataAdapter("select id_hotel AS 'Реден број',ime AS 'Реден број',ulica AS " +
    "'Улица',telefon AS 'Телефон', faks AS 'Факс',mail AS 'Маил',grad AS 'Град' from Hotel", konekcija);
SqlCommandBuilder comm = new SqlCommandBuilder(dataAdapter);
DataTable table = new DataTable();
table.Locale = System.Globalization.CultureInfo.InvariantCulture;
dataAdapter.Fill(table);
tabela.ReadOnly = true;
sourse.DataSource = table;
konekcija.Close();
Controls.Add(tabela);
```

Слика 68. Приказ на извршен код за табеларен приказ на податоци

Објектот gb кој е креиран од класата за групирање има примена за групирање на податоци со можност за избор, но само еден елемент од понудените опции за избор.

```
GroupBox gb = new GroupBox();
gb.Width = 550;
gb.Height = 60;
gb.Location = new System.Drawing.Point(40, 100);
this.Controls.Add(gb);
```

Слика 69. Приказ на извршен код за група од податоци

На следната слика е прикажано користење на објект од класата RadioButton со сите негови својства кои се потребни за дизајн.

```
RadioButton rbIme = new RadioButton();
rbIme.Text = " Име";
rbIme.Width = 100;
rbIme.Height = 40;
rbIme.Font = new System.Drawing.Font(rbIme.Name, 12F);
rbIme.Location = new Point(10, 10);
rbIme.Click += new EventHandler(this.fIme);
gb.Controls.Add(rbIme);
```

Слика 70. Приказ на извршен код за радио копче

На следната слика е прикажано користење на објект од класата ComboBox со дефинирани две можности и тоа ‘Тип соба’,’Цена за соба’. Објектот кој е креиран од оваа класа нуди можност за избор од паѓачка листа.

```
ComboBox lis = new ComboBox();
lis.Items.Add("Тип соба");
lis.Items.Add("Цена за соба");
lis.Width = 150;
lis.Font = new System.Drawing.Font(lNAslov.Name, 15F);
lis.Location = new Point(115, 130);
this.Controls.Add(lis);
```

Слика 71. Приказ на извршен код за паѓачка листа

На следната слика е прикажан изборот од паѓачката листа со две можности како и самата работа на избор – зголемување на големината на формата за работа.

```
private void fIzbor(object sender, EventArgs e)
{

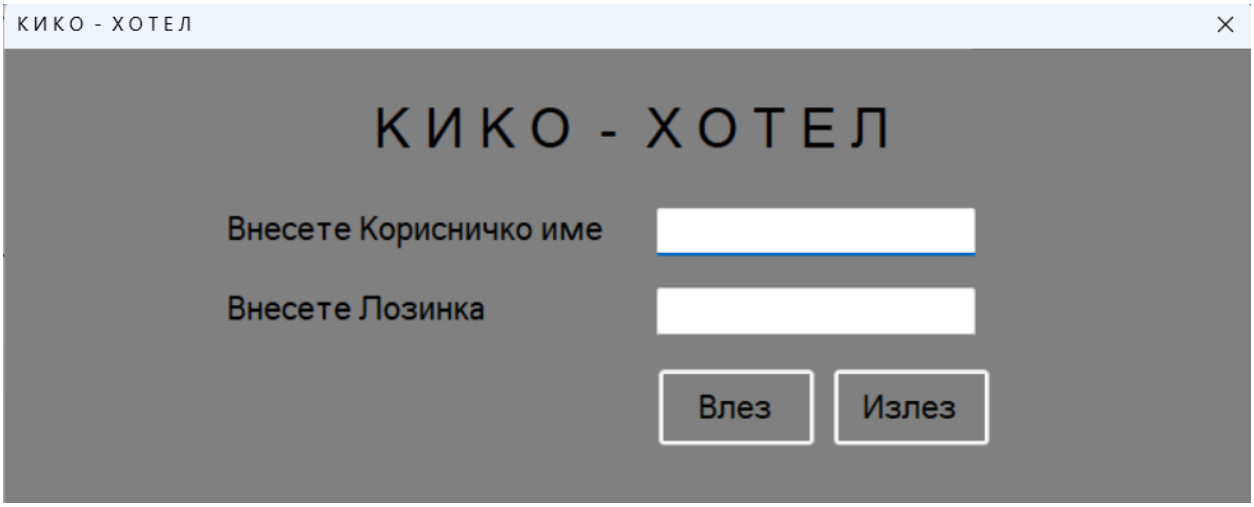
   if(lis.SelectedIndex==0)
    {
        this.Width = 1000;
        this.Height = 440;

    }
   if(lis.SelectedIndex==1)
    {
        this.Width = 1000;
        this.Height = 440;
    }
}
```

Слика 72. Приказ на извршен код за избор од група податоци

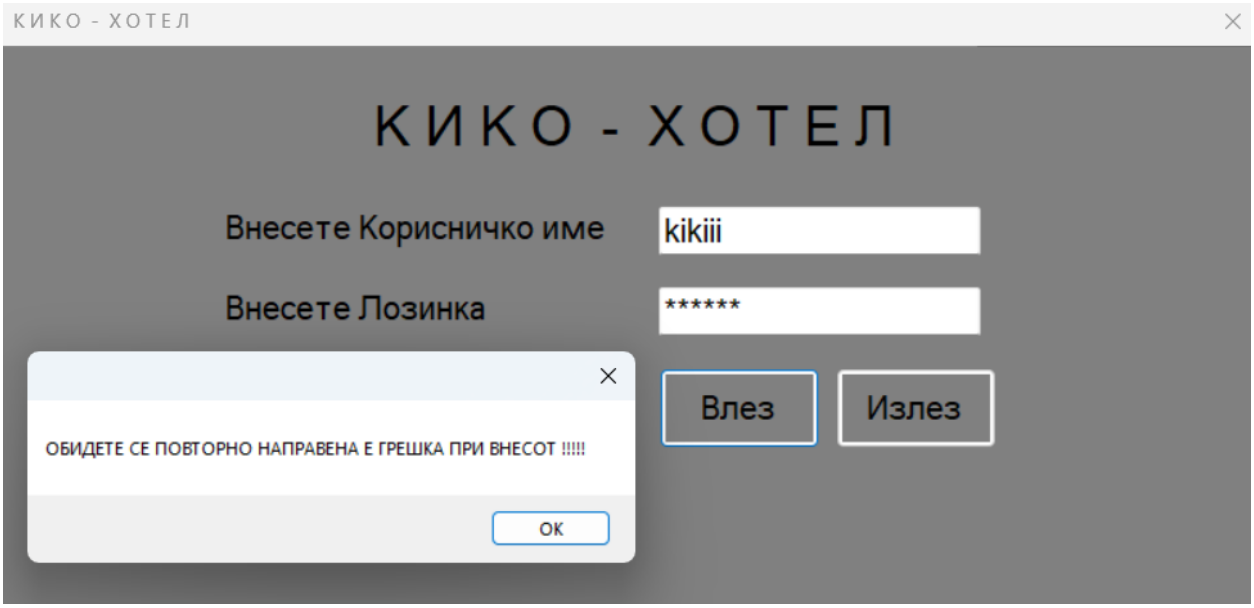
5.2. Дизајн на кориснички интефејс

При старт на програмата на екран се прикажува форма за внесување на корисничко име и лозинка на вработениот во хотелот. При внесување на податоци истите имаат можност да бидат или точни или неточни.



Слика 73. Приказ на почетна форма

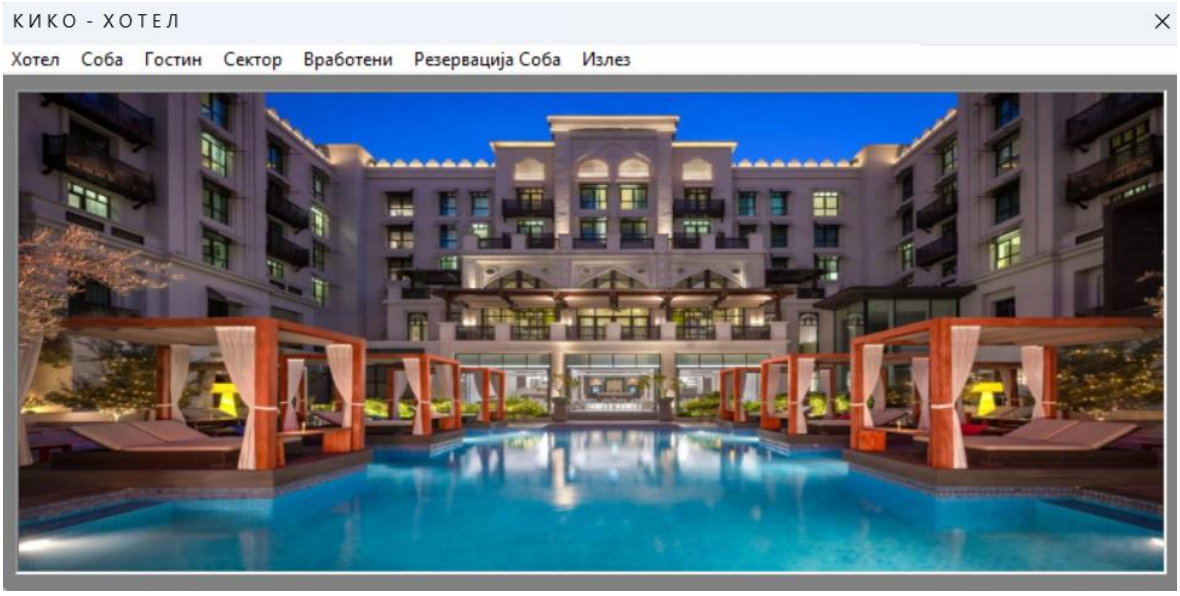
Доколку корисникот направи грешка во внесување на податоците системот врши контрола за направената грешка и истиот прикажува порака на екран за некоректен внес.



Слика 74. Приказ на форма за логирање

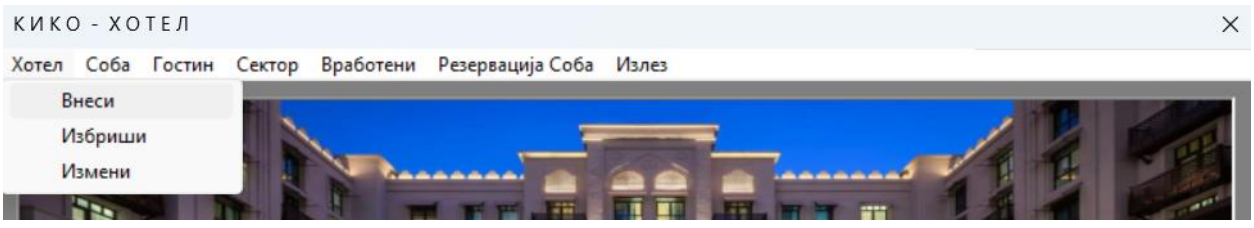
Доколу корисникот внесе точно корисничко име и лозинка на екран се прикажува мени со можност за избор и тоа:

- Хотел
- Соба
- Гостин
- Сектор
- Вработен
- Резервација Соба
- Излез



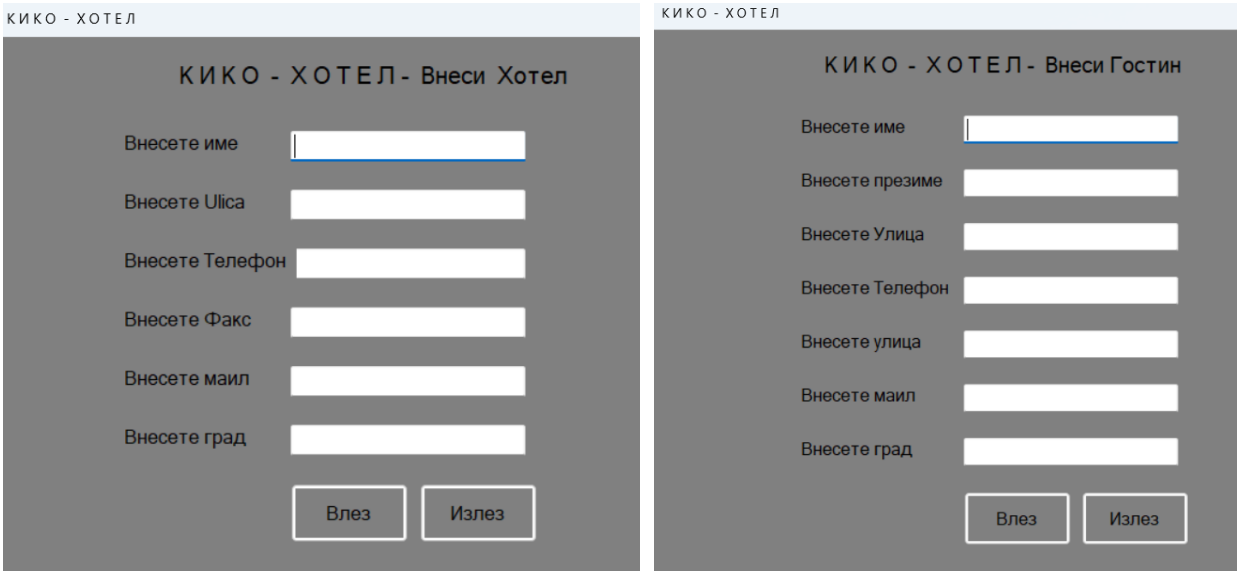
Слика 75. Приказ на форма со мени за избор

За секој елемент од менито креирано е подмени за внесување, бришење и измени на податоците.



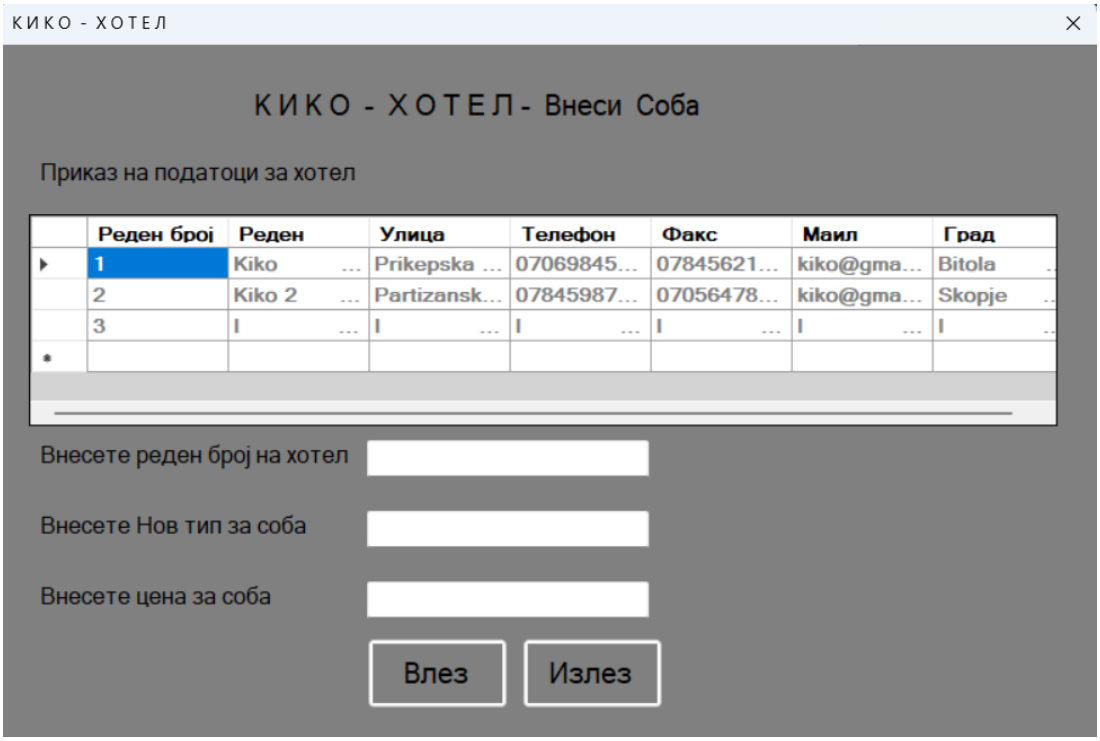
Слика 76. Приказ на подмени од главното мени за избор

На следните две слики се прикажани форми за внесување на вработен и хотел.



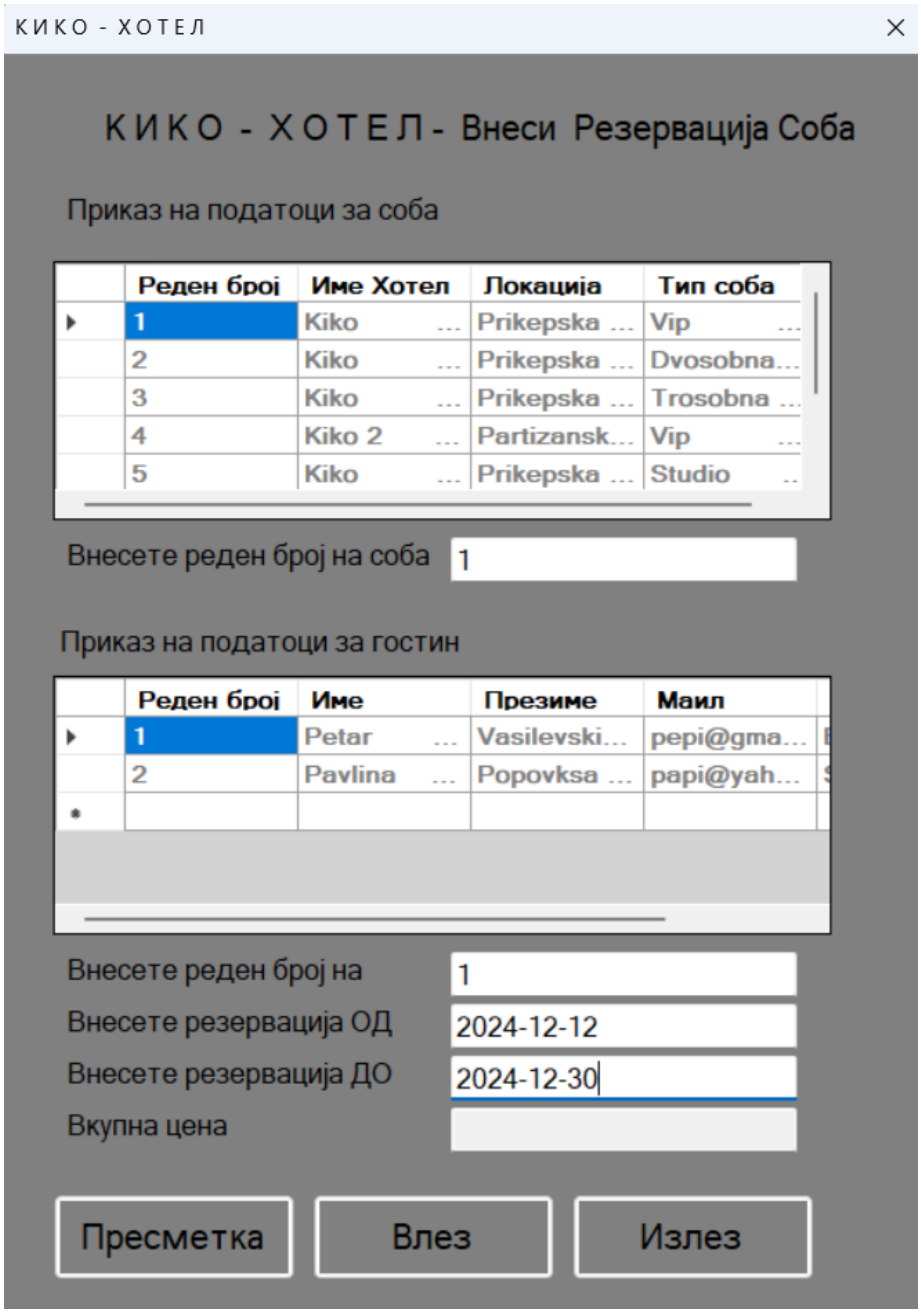
Слика 77. Приказ на форма за внесување на податоци

На следната слика е прикажано внесување на податоци во базата на податоци за ентитетот Соба. На екран се прикажува табела каде се прикажуваат податоците на кој хотел припаѓа собата. Потоа се внесува редниот број на хотелот за новиот тип на соба и цената за таа соба.



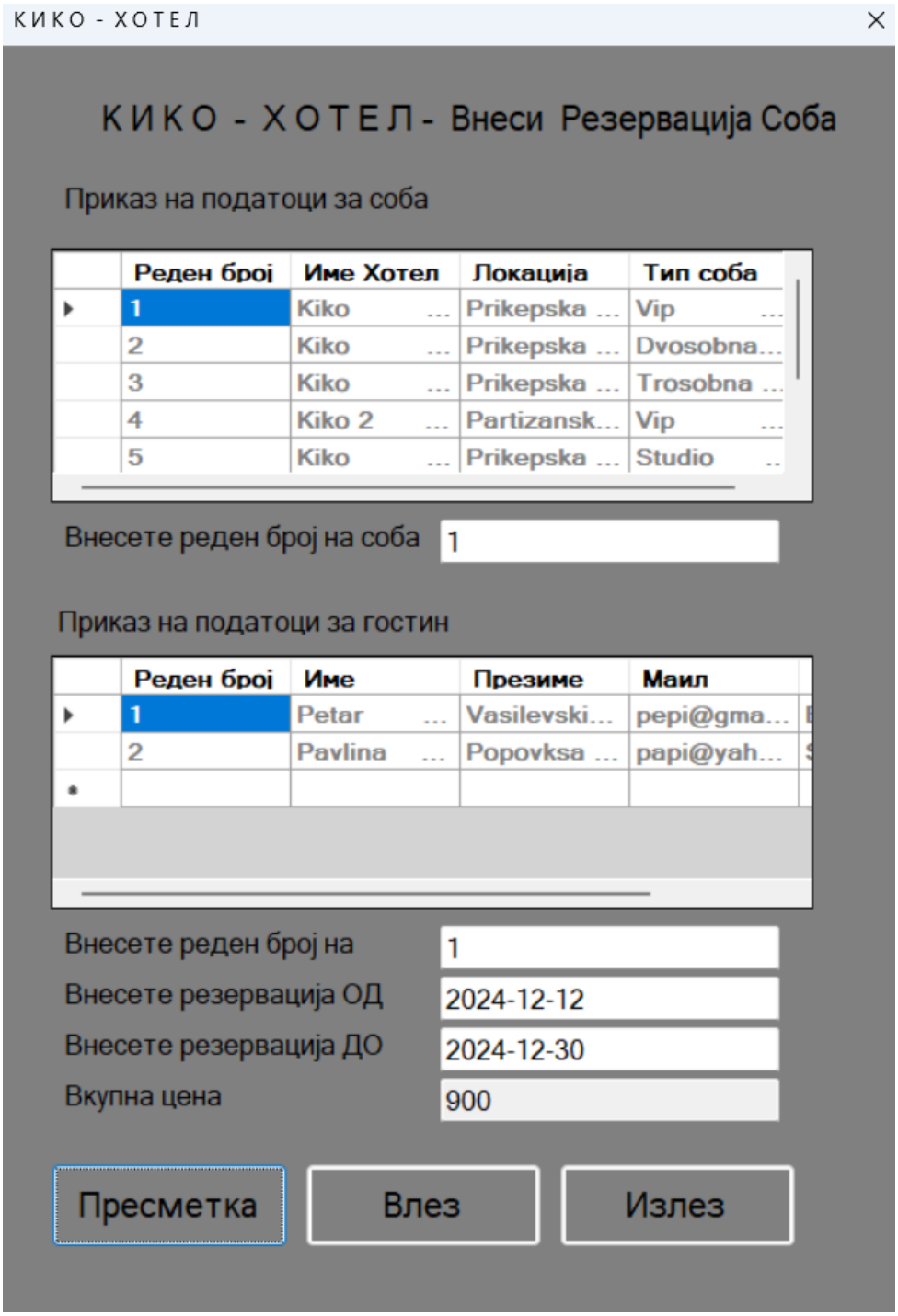
Слика 78. Приказ на форма за внесување на соба

На следната слика е прикажано внесување на резервација на соба за која се потребни повеќе податоци. Како податоци за внесување се потребни редниот број на собата, редниот број на гостинот, дата од кога почнува резервацијата, дата кога завршува резервацијата и посебно текст поле во кое не е дозволено внесување на податоци. За да се изврши резервацијата потребно е прво да се избере копчето “Пресметај” каде се врши пресметката за вкупната цена на целата резервација.



Слика 79. Приказ на форма за резервација

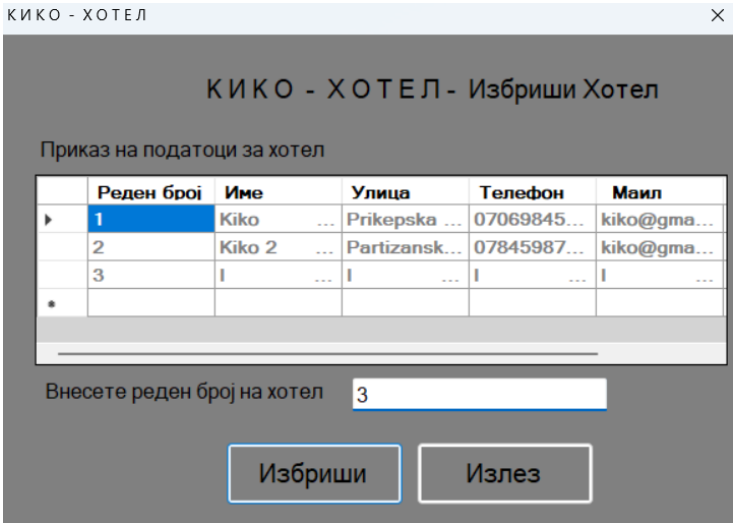
Со изборот на копчето пресметка, текст полето кое останува празно за вкупна цена за резервацијата се пополнува со цена според бројот на денови кои се избрани за престој. Потоа со избор на копчето за внес на податоците во базата на податоци истите податоци се запишуваат во база.



Слика 80. Приказ на форма за резервација со пресметка

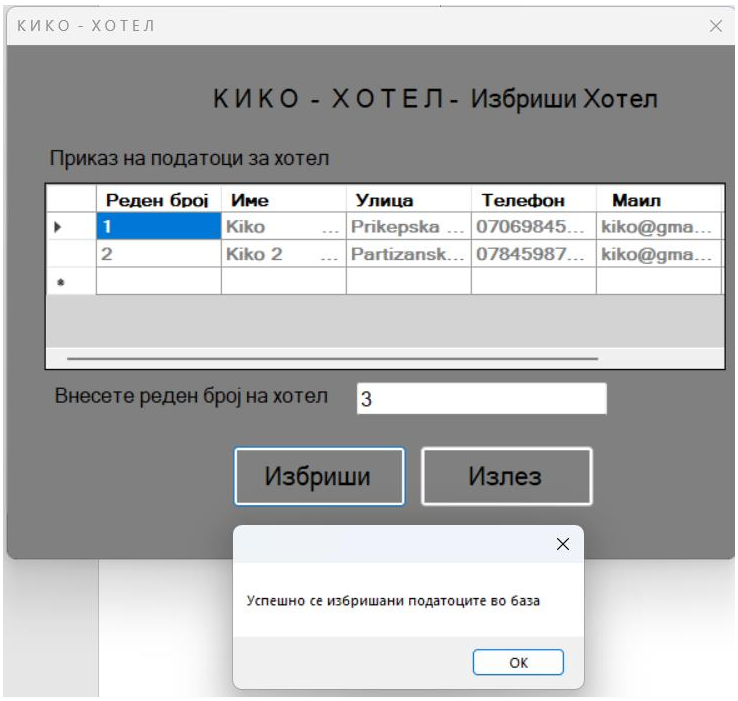
На следната слика е прикажана формата за бришење на податоци од база за ентитетот Хотел. Прикажана е табела со хотелите кои се запишани во базата а потоа

корисникот е потребно да внесе реден број на хотел кој е потребен да се избриши. Со избор на копчето за бришење, се бриши податокот од базата на податоци.



Слика 81. Приказ на форма за бришење на хотел

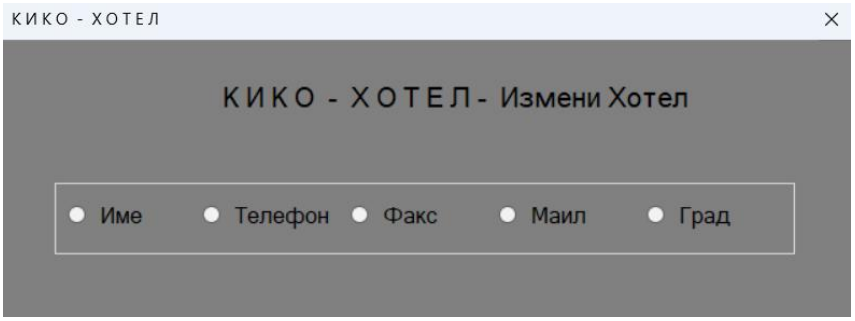
На следната слика е прикажана формата со приказ на податоците каде реден број 3 за хотелот веќе е избришан.



Слика 82. Приказ на форма за бришење на хотел

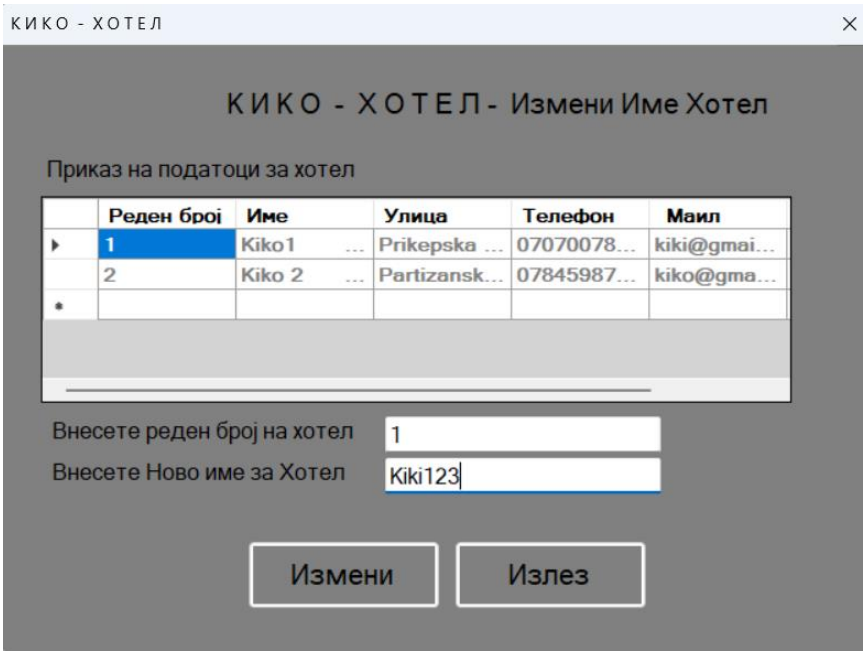
На следната слика е прикажана форма за измена на Хотел со можност за избор и тоа:

- Име
- Телефон
- Факс
- Маил
- Град



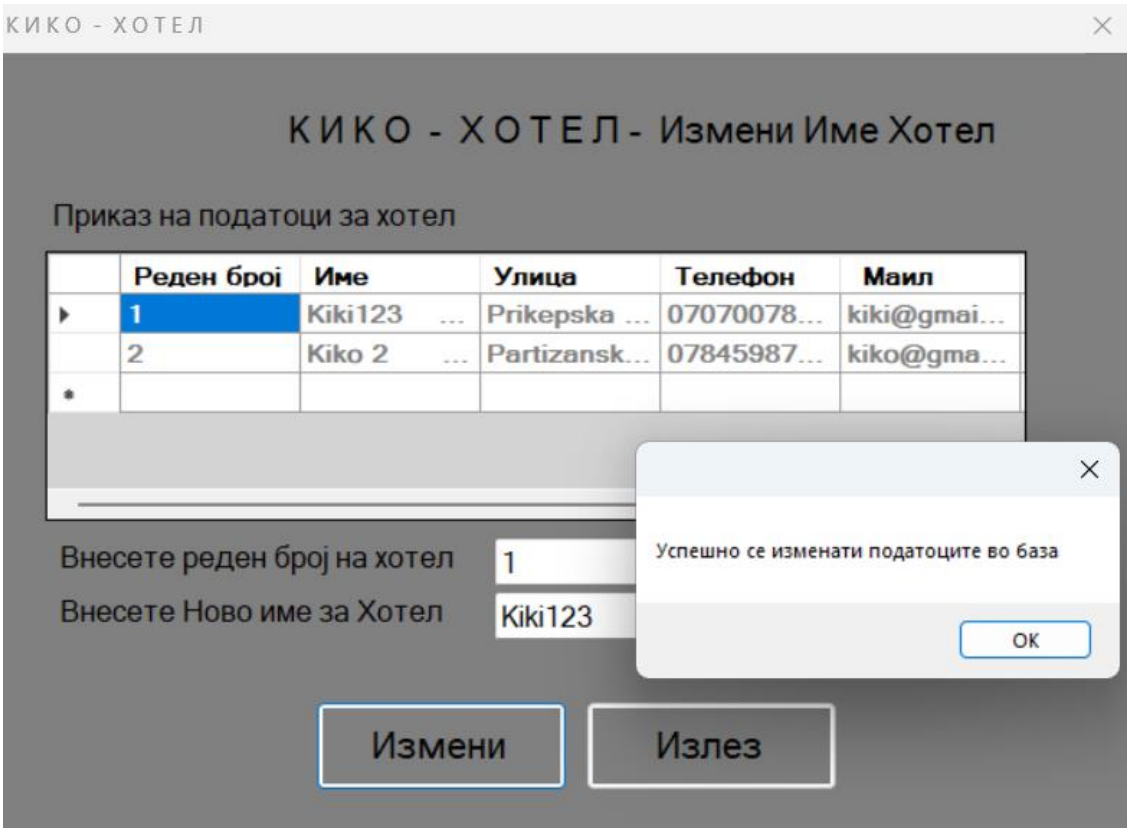
Слика 83. Приказ на форма за избор на податоци за измена

При избор од формата за измена на име на хотелот на екран се прикажува порака за измена на име и табеларен приказ на податоците од табелата за Хотел. На самата форма постојат две текст полиња од кои во првото текст поле потребно е да се внеси редниот број на хотелот за измена и во второто поле, потребно е да се внеси новото име на хотелот.



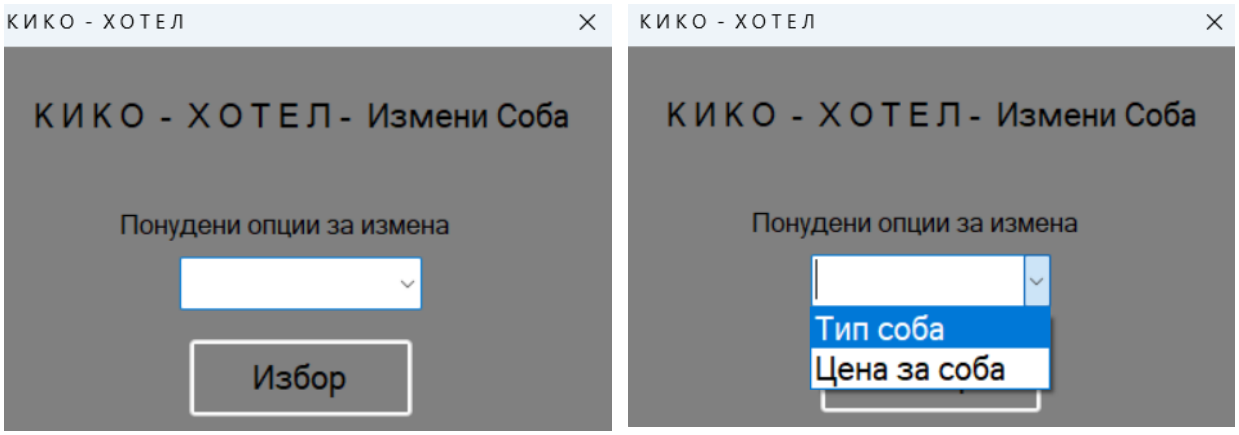
Слика 84. Приказ на форма за измена на име

На следната слика е прикажана измената која е извршена за хотелот, со реден број 1 е извршена промена на името.



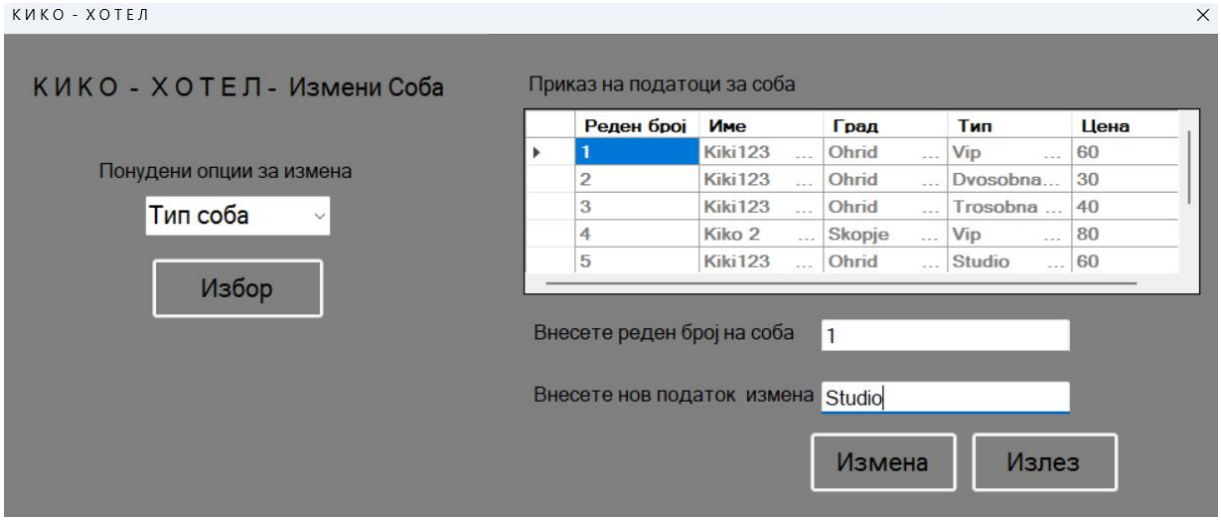
Слика 85. Приказ на форма за измена на име на хотел

На следните две слики е прикажана формата за измена на соба со користење на паѓачка листа за избор.



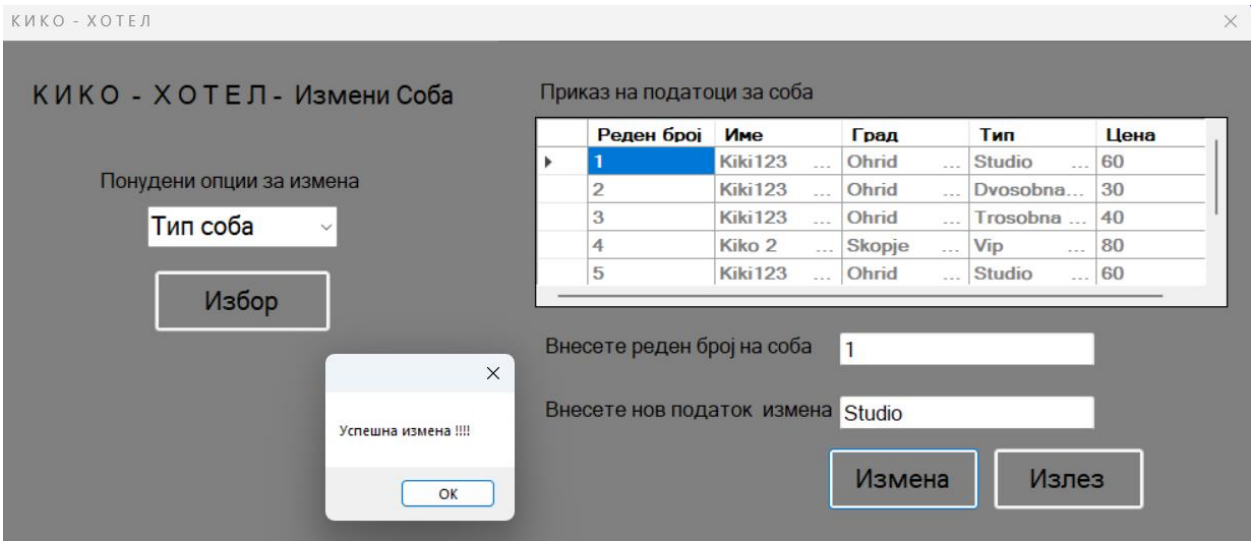
Слика 86. Приказ на форма за избор за измена на соба

При изборот за измена на тип на соба се врши зголемување на формата со дополнителни компоненти и можност за измена на тип на соба. На формата прво се прикажува табеларен запис на податоците од соба, а потоа две текст полиња со можност за внесување на податоци. Во првото поле се внесува реден број на соба, а во второто поле се внесува новиот тип на соба. Со избор на копчето измени се врши измена во базата на податоци и повторно се врши приказ на табелата со изменети податоци.



Слика 87. Приказ на форма за измени на соба

На следната слика е прикажана измената за тип на соба, за соба со реден број 1, и на екран се прикажува порака за успешна измена и табелата со сите податоци дополнително со измената која е направена.



Слика 88. Приказ на форма за измена на податоци за соба

6. ЗАКЛУЧОК

Изборот на темата на самиот Магистерски труд е со цел да се докаже и покаже колку е важна внатрешната организација на секоја компанија без разлика на нивната тема и начин на работа. За да се зголеми функционалноста на секоја компанија потребен е внатрешен локален систем за обработка на потребни податоци кои се како секојдневни задачи на дадени ресурси во компанијата.

Изборот на темата е обработка на систем за Windows апликација со помош на Windows Forms од Visual Studio. Во системот се обработени интерните обврски на еден вработен како менаџер во Хотелот.

Со цел да се зголеми ефикасноста на апликацијата се искористени бази на податоци и SQL сервер со кои е поврзана апликацијата. Значи магистерскиот труд се состои од два дела и тоа:

- Серверски дел
- Апликациски дел

И двата дела кога се добро изменаџирани и добро обработени можат да понудат одличен систем за работа, кој е од голема помош на вработените во дадена компанија (во нашиот случај Хотел.)

7. КОРИСТЕНА ЛИТЕРАТИРА

1. Brad Bulger, Jay Greenspan, David Wall, 2004.. MySQL/PHP Database Applications [2nd ed], Wiley Pub

2. Charles Bell, 2018.. Introducing the MySQL 8 Document Store, Apress

3. Adrian W. West, Steve Prettyman, 2018.. Practical PHP 7, MySQL 8, and MariaDB Website Databases, Apress

4. Abraham Silberschatz, Henry F. Korth, S. Sudarshan, 2010.. Database System Concepts, 6th Edition, McGraw-Hill

5. Shabbir Challawala, Jaydip Lakhatariya, Chintan Mehta, Kandarp Patel, 2017.. MySQL 8 for Big Data, Packt Publishing

6. Eric Vanier, Birju Shah, Tejaswi Malepati, 2019.. Advanced MySQL 8: Discover the full potential of MySQL and ensure high performance of your database, Packt Publishing

7. Robert Manger, 2010.. Osnove projektiranja baza podataka [Mrežno] Link: https://www.srce.unizg.hr/files/srce/docs/edu/osnovni-tecajevi/d310_polaznik.pdf [Pokušaj pristupa 2 Rujan 2020.].

8. Koraljka B. Damir M., n.d., UTJECAJ DENORMALIZACIJE BAZE PODATAKA NA WEB APLIKACIJE [Mrežno] Link: https://bib.irb.hr/datoteka/500408.CASE_10_-_Utjecaj_denormalizacije_baza.pdf [Pokušaj pristupa 2 Rujan 2020.].

9. Robert Manger, 2008.. BAZE PODATAKA Dodaci uz skripta [Mrežno] Link: http://web.studenti.math.pmf.unizg.hr/~manger/bp/dodaci.pdf [Pokušaj pristupa 2 Rujan 2020.].

10. Robert Manger, 2003.. BAZE PODATAKA skripta [Mrežno] Link: http://jadran.izor.hr/~dadic/EKO/baze-podataka.pdf [Pokušaj pristupa 2 Rujan 2020.].

11. Dries Buytaert, 2010.. The history of MySQL AB [Mrežno] Link: https://dri.es/the-history-of-mysql-ab [Pokušaj pristupa 2 Rujan 2020.].

12. Anon., 2020.. MySQL 8.0 reference Manual [Mrežno] Link: https://dev.mysql.com/doc/refman/8.0/en/what-is-mysql.html [Pokušaj pristupa 2 Rujan 2020.].

13. https://learn.microsoft.com/en-us/visualstudio/ide/create-csharp-winform-visual-studio?view=vs-2022

14. https://stackoverflow.com/questions/76057586/i-cant-find-windows-forms-app-net-framework-in-visual-studio-2022-free-do

15. https://mk.wikipedia.org/wiki/Нормализација_на_бази_на_податоци