Aplikacija za desktop računala za mjerenje masnoće u hrani

Šangut, Filip

Undergraduate thesis / Završni rad

2020

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **Josip Juraj Strossmayer University of Osijek, Faculty of Electrical Engineering, Computer Science and Information Technology Osijek / Sveučilište Josipa Jurja Strossmayera u Osijeku, Fakultet elektrotehnike, računarstva i informacijskih tehnologija Osijek**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:200:068452

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2024-04-23**

Repository / Repozitorij:

Faculty of Electrical Engineering, Computer Science and Information Technology Osijek

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Sveučilišni stručni studij Elektrotehnike, smjer Informatika

Aplikacija za desktop računala za mjerenje masnoće u hrani

Završni rad

Filip Šangut

Osijek, 2020.

Sadržaj:

1. UVOD 1
1.1. Nedostatak VLCAD-a 1
1.2. Zadatak završnog rada 2
2. IZRAČUN BJELANČEVINA, KALORIJA, MASTI, ŠEĆERA I UGLJIKOHIDRATA 3
3. TEORIJSKE PODLOGE 4
3.1. C# i .NET platforma 4
3.2. Baza podataka 6
4. C# APLIKACIJA ZA GENERIRANJE DOKUMENTA 9
4.1. Dizajn i korisničko sučelje 9
4.2. Osnovne funkcionalnosti aplikacije 14
4.3. Baza podataka 23
5. ZAKLJUČAK 26
LITERATURA 28
SAŽETAK 29
ABSTRACT 30
ŽIVOTOPIS 31

1. UVOD

Izračunavanje udjela kalorija, masti, bjelančevina, ugljikohidrata i šećera u svakodnevnim namirnicama ima velike koristi, ukoliko želimo pratiti razinu unosa navedenih tvari u organizam. Na taj način možemo preciznije paziti i pratiti razvoj tijela te njegovo oblikovanje, što je posebno značajno kod nekih oboljenja, kao primjerice kod osoba s VLCAD (eng. *Very long chain acyln-CoA dehydrogenase deficiency*). Oboljeli od nedostatka VLCAD, imaju poteškoće pri pravilnom razgrađivanju određenih masti, osobito u razdoblju bez hrane te moraju paziti na kvantitativni unos određenih tvari kako bi održali homeostazu i normalno funkcionirali [1].

Zadatak ovog završnog rada je izrada aplikacije koja će omogućiti korisnicima praćenje dnevnog unosa bjelančevina, masnoća, šećera i ugljikohidrata na dnevnoj razini, te će im sugerirati je li to u skladu s njihovom preporučenim vrijednostima. Aplikacija se temelji na tome da korisnik odabire namirnice i unosi njihovu masu, a kao rezultat dobije zbroj svih hranidbenih vrijednosti u vidu bjelančevina, kalorija, masnoća, šećera i ugljikohidrata. Korisniku je u svakome trenutku omogućen pregled svih prethodno spremljenih obroka, kao i mjesečna evidencija istih putem e-mail.

U radu je ponuđen pregled svih korištenih tehnologija tijekom realizacije aplikacije, a to su C# i lokalna baza podataka. Baza podataka sadrži sve namirnice i u nju se upisuju parametri svake od namirnica preko kojih se radi proračun svakoga obroka.

1.1. Nedostatak VLCAD-a

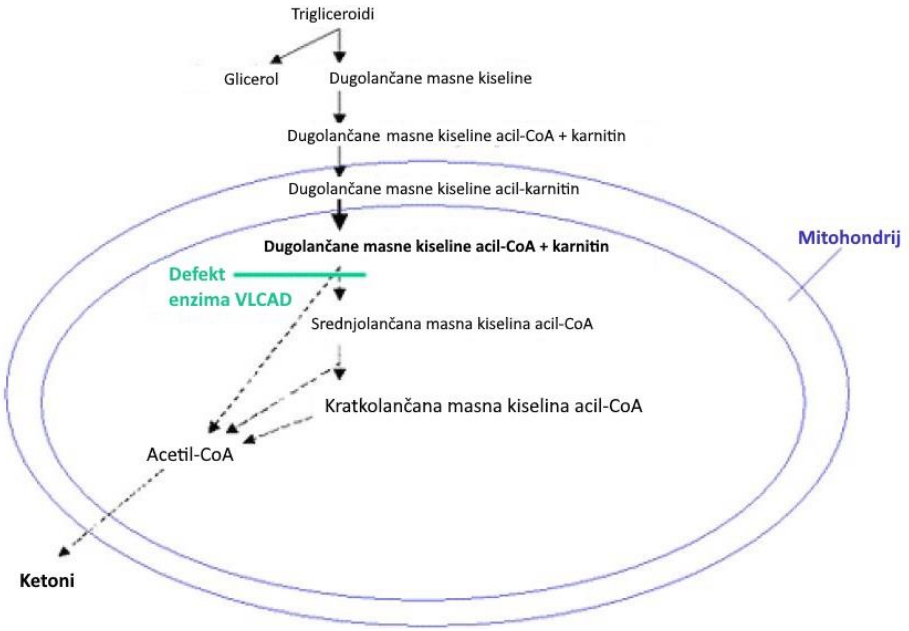
Vrlo dugi lanac acil-COA dehidrogenaze (eng. *very long-chain acyl-CoA dehydrogenase*) je enzim koji katalizira prvi korak u procesu mitohondrijske β-oksidacije dugolančanih masnih kiselina koje sadrže od 14 do 20 ugljikovih atoma. Nedostatak ovog enzima nasljeđuje se autosomno-recesivo što implicira da obje kopije gena, od majke i oca, u svakoj stanici nose mutacije u *VLACAD* genu [2].

Kao rezultat mutacije, osobe s deficitom ovog enzima imaju poteškoća u razgradnji masnih kiselina iz hrane, a one predstavljaju glavni izvor hrane za srce i mišiće, osobito u razdoblju gladovanja kad su izvor energije i za ostale organe. Masne kiseline se nakupljaju ili ostaju djelomično razgrađene, a to ima toksično djelovanje.

Karakteristični simptomi ovog poremećaja pojavljuju se još u djetinjstvu, a uključuju letargiju (manjak energije) i hipoglikemiju (nisku razinu šećera u krvi). Oboljele osobe izložene su povećanom riziku abnormalnosti jetre, srčanim problemima i rabdomiolizi (razgradnji

mišićnog tkiva). Prevalencija bolesti je otprilike 1 na 40 000 do 120 000 ljudi [3].

Posljedično, ovakvo stanje iziskuje dugoročno liječenje i kontroliranu prehranu koja obiluje ugljikohidratima, a siromašna je mašću te nadoknadu triglicerida srednjeg lanca, uz izbjegavanje dugotrajnijeg gladovanja i teži fizički napor [4].



Slika 1.1. Shematski prikaz procesa razgradnje dugolančanih masnih kiselina u mitohondriju i ključnog početnog enzima VLCAD u razgradnji istih, prilagođeno prema [5].

1.2. Zadatak završnog rada

Cilj završnog rada je realizirati C# desktop aplikaciju za oboljele od VLCAD deficita, koja bi im omogućila izračun bjelančevina, kalorija, masti, šećera i ugljikohidrata pojedinih namirnica te korisnikov unos personaliziranih propisanih dnevnih vrijednosti koje smije unijeti u organizam. Prilikom odabira namirnica, potrebno je omogućiti računanje svih unesenih tvari i njihovu pohranu. Također, namjera je osigurati korisnicima mjesečno (svakog prvog u mjesecu) dospijeće svih pohranjenih obroka putem e-maila, uz mogućnost proizvoljnog vremena slanja tj. kada god to korisnik želi, kao i pregled povijesti upisanih obroka, neovisno o tome koji je dan u mjesecu. Uz navedeno, korisniku je omogućen prikaza neiskorištenih nutritivnih parametara kako bi mu sljedećim obrocima osigurao optimalne dnevne vrijednosti. Korisnik ima mogućnost izmjene parametara namirnica, unos novih namirnica te njihovo brisanje. Korisnik na dnevnoj bazi ima limit koji bi trebao poštovati, a u slučaju da korisnik unese vrijednosti iznad limita, tada mu u sklopu aplikacije dolazi obavijest o kojim se to parametrima radi.

2. IZRAČUN BJELANČEVINA, KALORIJA, MASTI, ŠEĆERA I UGLJIKOHIDRATA

Korisniku je od krucijalne važnosti imati mogućnost izračuna bjelančevina, kalorija, masti, šećera i ugljikohidrata u gramima kako bi izbjegao neželjene simptome bolesti.

2.1. Izračun vrijednosti putem aplikacije

Izrada tablica sa sveukupnim informacijama o vrijednosnom sastavu namirnica u *Microsoft Excel* programu jedan je od načina kako možemo izračunati potrebne vrijednosti za svaki obrok. Putem te aplikacije na vrlo jednostavan i učinkovit način možemo izračunati potrebne podatke o sadržaju namirnica. Ipak, ovakva primjena na svakodnevnoj razini može biti dosta nepraktična i može rezultirati greškama u izračunu i na taj način utjecati na zdravlje korisnika, jer bilo kakve nenamjerne promjene u podacima mogu utjecati na zdravlje korisnika.

Izrađena aplikacija „Maskal“ pruža bolju alternativu, gdje uz postojeću bazu podataka korisnik ima mogućnost samostalno nadopunjavati nove namirnice, odnosno informacije o njihovom vrijednosnom sastavu bjelančevina, masti, šećera, ugljikohidrata i ukupnih kalorija. Na ovaj način doprinosimo dodatnoj sigurnosti, koja se posebice odnosi na to da sve što modificiramo bude namjenski, odnosno da se sve izmjene dogode samo kada i kako mi to želimo. Uz navedeno, aplikacija ima proširenu funkcionalnost prikazivanja prethodno upisanih događaja te mogućnost obavještavanja korisnika o graničnim vrijednostima unosa pojedinih tvari u organizam. Također, korisniku je omogućeno mjesečno primanje izvještaja o konzumiranim namirnicama i njihovoj vrijednosti putem e-mail adrese te njihov prosječni dnevni unos, što dodatno pomaže učinkovitom poboljšanju prehrane te tako i samoj kontroli bolesti.

Aplikacija će omogućiti liječnicima da jednostavno i sustavno prate stanje svojih pacijenta, odnosno njihov unos pojedinih namirnica. Na ovaj način moći će lakše usmjeravati pacijenta kroz trajan proces reguliranja stanja. Uz ranije navedeno, korisnici su u mogućnosti vidjeti koje sve namirnice mogu kombinirati, a da upotpune dnevnu normu unosa tvari, s obzirom da imaju točno zadane vrijednosti. Sam izgled aplikacije sa svim njezinim funkcijama prikazan je na slici 4.

3. TEORIJSKE PODLOGE

Za izradu aplikacije koristio sam C# programski jezik i .NET platformu te Service Based Database bazu podataka.

3.1. C# i .NET platforma

C# je potpuno objektno orijentirani programski jezik, što znači da svi elementi koji se nalaze unutar C# aplikacije, predstavljaju objekt, dok objekt predstavlja strukturu sa svojim metodama, interakcijama i podatkovnim elementima koristeći .NET platformu. Kada pokrećemo napisani C# kôd, tada C# kompajler uzima napisani kôd kao ulaz i na njemu vrši obradu te ga prikazuje na izlazu kao IL (eng. *Intermediate language*) kôd koji se sprema u **.exe* ili **.dll* datoteku. No kako i dalje procesor ne razumije taj kôd, potreban nam je CLR (eng. Common Language Runtime) kako bi taj kôd pretvorio u strojni, kojeg procesor razumije. To se sve odvija preko JIT (eng. *just-in-time*) kompajlera, koji pretvara IL kôd u trenutku kada ga želimo pokrenuti (npr. kada dvostruko kliknemo na *.exe* datoteku), te taj kôd procesor može izvršiti. Tijekom ovoga procesa, napisani program ima dvije vrste pogrešaka: pogreške u sintaksi (eng. *compile time errors*) i pogreške izvršavanja (eng. *runtime errors*). Pogreške u sintaksi otkriva C# kompajler i on sprječava da se napisani kod pretvori u .exe. Primjer ovakve pogreške je:" pogrešno napisana funkcija ili izostanak „;“ ". Pogreške izvršavanja su puno ozbiljnije pogreške koje se javljaju pri izvođenju programa, odnosno tijekom JIT prevođenja i kod njih se izvršavanje programa zaustavlja, jer bi nastavak mogao utjecati na druge datoteke ili operacijski sustav. Primjer ovakve pogreške je:" pokušaj dijeljenja s nulom ili pokušaj upisivanja podataka u *read-only* datoteku" [6].

C# se koristi za izradu raznih vrsta aplikacija poput baze podataka, *XML* web servisa, *Windows* aplikacija i drugo. Nastao je s ciljem da bude siguran, pouzdan, jednostavan i moderan objektno-orijentiran jezik visokih performansi za .NET platformu temeljen na C++, Visual Basic-a i Java objektnih jezika. Nije neovisan o operacijskom sustavu, a glavna primjena, zbog čega je i stvoren, je izrada desktop aplikacija i internet aplikacija u .NET okruženju.

Objektno-orijentiranim programiranjem kôd koji programiramo trebao bi biti izuzetno čitljiv i lako izmjenjiv, uz mogućnost jednostavne nadogradnje i ponovnog korištenja.

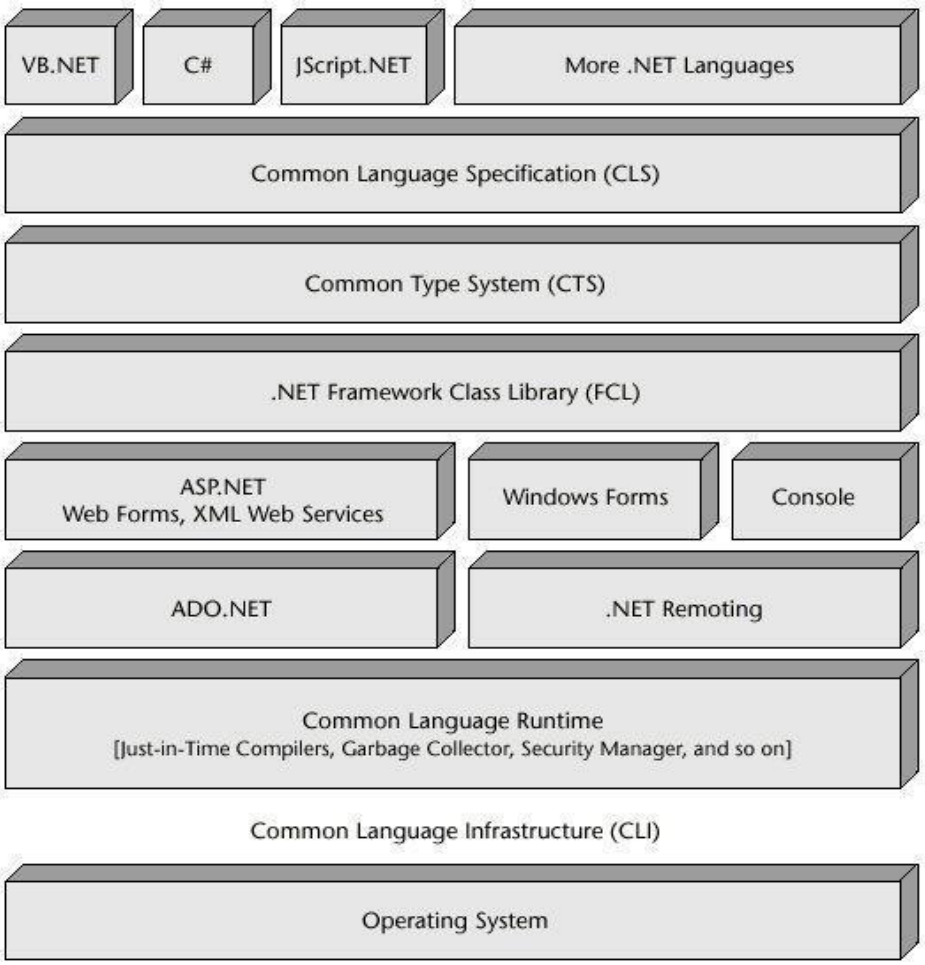
Svojstva koja se smatraju osnovnim načelima su:

- Nasljeđivanje – prilikom stvaranja nove klase, ta nova klasa sadrži sve članove

„nadklase“ koju je naslijedila tj. nasljeđuje njezine članove i metode [7]

- Polimorfizam – omogućava funkcijama da imaju isto ime, no s različitom funkcionalnostima [7]
- Enkapsulacija – omogućavanje određivanja prava pristupa ostalim klasama ili metodama [8].

Iz objektno-orijentiranoga jezika Java je dobrim dijelom preuzeta sintaksa i semantika, te zahvaljujući .NET platformi omogućava kreiranje vizualnih aplikacija čak i onim korisnicima koji nemaju programerskoga iskustva. C# koristi na desetke ugrađenih tipova podataka i samo osamdeset ključnih riječi. Pri pokretanju, Windows komponenta na .NET platformi pokreće Common Language Runtime (CLR) tj. aplikacijski virtualni stroj i zbirku razrednih biblioteka, a to je osnova kako bi se kreiralo izvršno okruženje u kojima zajedno rade biblioteke i jezici[9].



Slika 3.1. Arhitektura .NET platforme [9]

Na slici 3.1. je prikazana arhitektura .NET platforme, a sastoji se od [10]:

- Common Language Specification (CLS) je skup specifikacija ili smjernica koje definiraju .NET jezike. CLS je podskup objekata i definira uobičajene tipove upravljanih jezika te pruža tri razine sukladnosti
- Common Type System (CTS) sprječava pogrešno emitiranje, određuje pravila povezana s tipovima parametara koje jezici moraju slijediti. CTS je knjižnica podataka .NET tipova
- .NET Framework Class Library (FCL) je skup upravljanih klasa koje pružaju pristup sistemskim uslugama koji se mogu koristiti za razvoj UI ili web aplikacija
- Korisničko i programsko sučelje (ASP.NET, Windows Forms, Console) sadrže Windows obrasce koji pružaju korisničko sučelje za web
- Standardne usluge sustava (ADO.NET, .NET Remoting) univerzalno su dostupne i standardizirane na svim jezicima dovodeći ih pod kontrolu .Net framework-a
- Common Language Runtime (CLR) nadzire izvršenje .NET aplikacija, upravlja kodom u vrijeme izvršenja, pruža osnovne usluge kao što su daljinsko upravljanje memorijom te upravlja nitima i ekvivalentan je JVM (Java Virtual Machine-u).

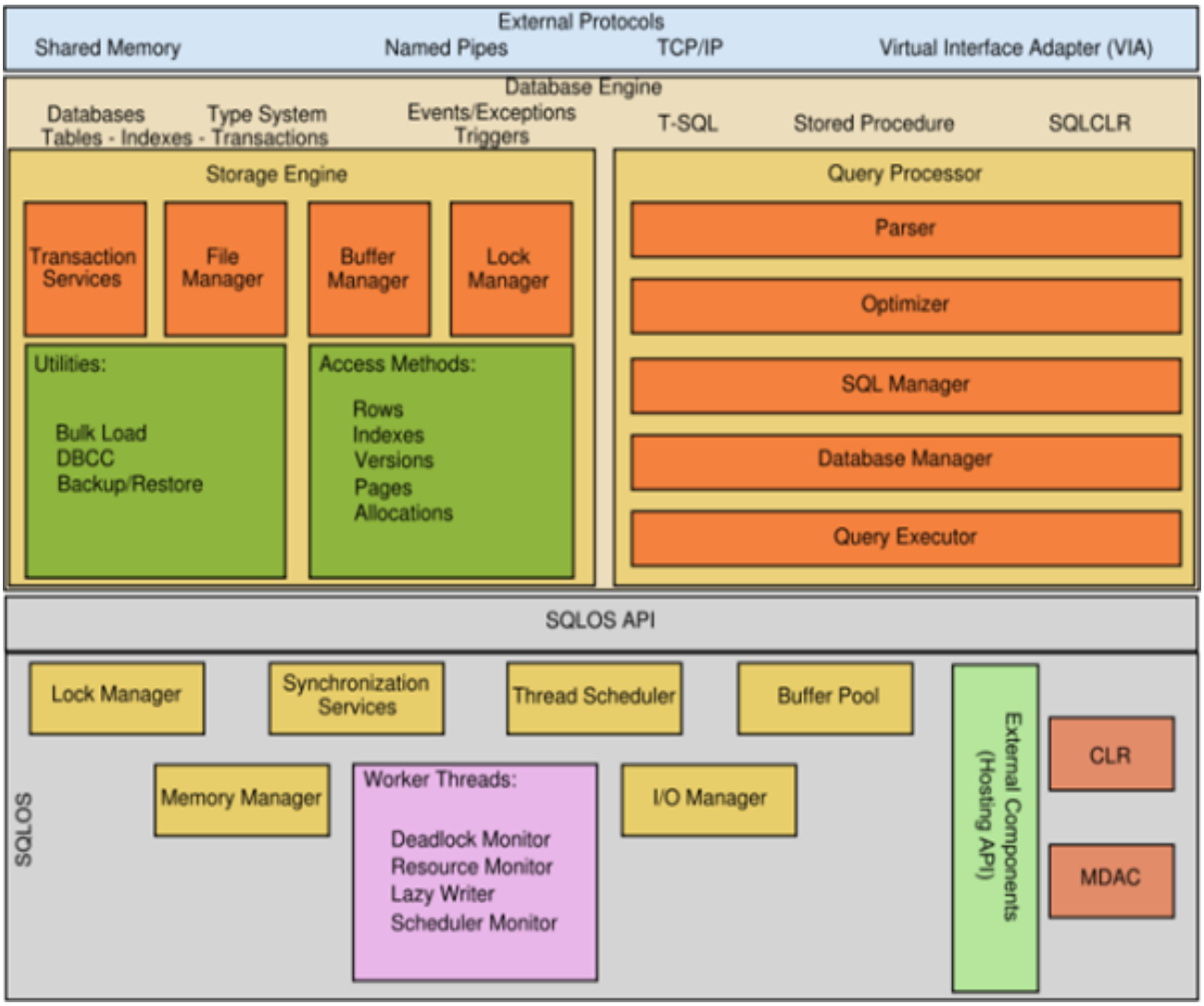
3.2. Baza podataka

Kao baza podataka se koristi *Service based database*, odnosno servisna baza podataka kojoj se pristupa putem poslužitelja. Kao datoteku podataka koristi se MDF (Master Database File) tip podatka tj. primarna datoteka baze podataka koja sadrži shemu i podatke. Svim podacima pristupa se putem *Connection string*-a, preko kojega tada otvaramo vezu prema bazi podataka i pišemo željene naredbe, a nakon što se one izvrše, zatvaramo vezu (slika 3.2).

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

    string name = listBoxSveNamirnice.GetItemText(listBoxSveNamirnice.SelectedItem);

    string cn_string = Properties.Settings.Default.dblNamirniceConnectionString.ToString();



    SqlConnection cn_connection = new SqlConnection(cn_string);
    if (cn_connection.State != ConnectionState.Open)
    {
        cn_connection.Open();
    }
    string sql_Text = "DELETE FROM tbl_Namirnice WHERE Namirnica='" + name + "'";
    SqlCommand cmdCommand = new SqlCommand(sql_Text, cn_connection);
    cmdCommand.ExecuteNonQuery();
    cn_connection.Close();
    load_list();
}
```

Slika 3.2. Prikaz koda za brisanja namirnice iz baze podataka

Microsoft SQL Server format ima zadaću dohvatiti i pohraniti podatke od strane aplikacije, a razvila ga je tvrtka Microsoft. Microsoft SQL podržava ODBC (*Open Database Connectivity*). Primarni jezik joj je *Transact SQL* (T-SQL) preko kojega korisnik ima mogućnost uz, osnovne upite (npr. SELECT), koristiti i složenije naredbe (npr. IF naredbe), a nastao je suradnjom Microsofta i Sybasea. T-SQL utječe na podršku za transakciju i donosi dodatnu sintaksu prilikom pisanja procedure. Postoje razne inačice, pa tako poznajemo: SQL95, Hydra, Sphinx, Plato, Kilimanjaro i druge, a sve je počelo prvom verzijom SQL Server for OS/2 1.0. Izašla je 1989. godine i bila je identična Sybase SQL serveru 3.0, koji je radio pod Unix-ovim sistemom. Arhitektura Microsoft SQL Servera (slika 3.3.) se sastoji od dvije glavne komponente: *Database Engine* i *SQLOS*. *Database Engine* obrađuje upite i upravlja pohranom podataka, a sastoji se od *Storage Engine* i *Query Processor*. *Query Processor* određuje najbolji način za izvršenje upita i obrađuje podatke koje dobiva od *Storage Engine* na temelju ulaznog upita. *Storage Engine* je zadužen za pohranu i preuzimanje podataka iz sustava za pohranu. *SQLOS* pruža brojne usluge operativnog sustava, kao što je ulaz/izlaz upravljanje, upravljanje memorijom, upravljanje radom niti te sinkronizacijskim servisom i slično [11].



Slika 3.3. Arhitektura Microsoft SQL Servera [11]

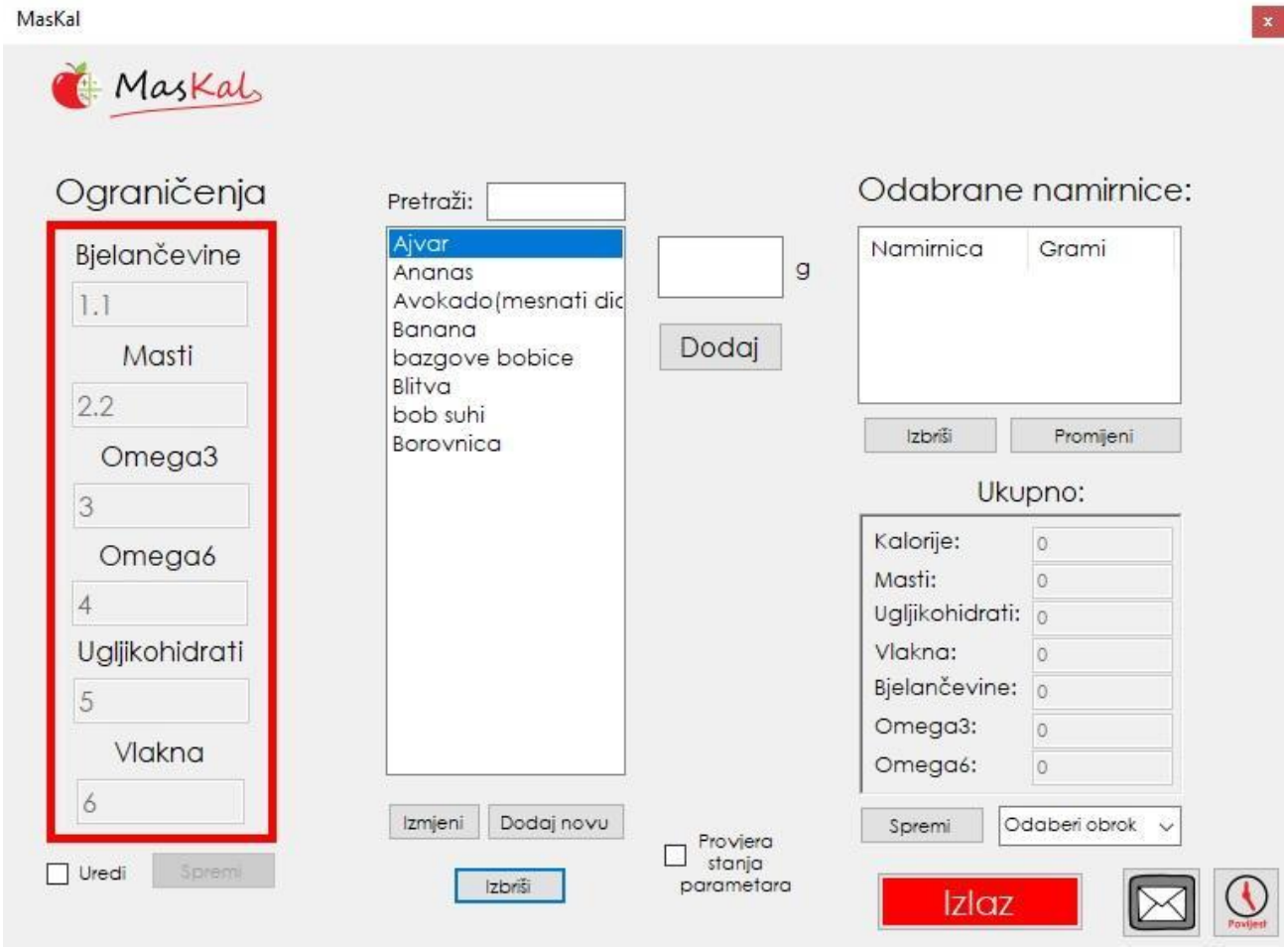
Osim ta dva glavna sloja, tu se nalazi i *External Protocols*. Kada aplikacija komunicira s *Server Database Engine*, programsko sučelje aplikacijskih programa (API) izloženo slojem protokola, oblikuje komunikaiju koristeći tabelarni paket podataka (TDS – *tabular data stream* packet). Postoje *Net – Libraries* i na poslužiteljskom i na klijentskom računalu koji obuhvaćaju TDS paket unutar standardnog protokola komunikacije, poput TCP/IP – protokol za komunikaciju putem interneta ili *Named Pipes* - protokol razvijen za lokane mreže (LAN).

4. C# APLIKACIJA ZA GENERIRANJE DOKUMENTA

Izrada aplikacije započeta je stvaranjem najjednostavnije forme u programskom jeziku C#, dok je izgled same aplikacije definiran osmišljavanjem korisničkog sučelja.

4.1. Dizajn i korisničko sučelje

Unutar aplikacije su uzeti osnovni alati poput *Button-a*, *TextBox-a*, *ListBox-a*, *Labela*, *PictureBox-a* i sl.



Slika 4.1. Izgled glavnog izbornika aplikacije

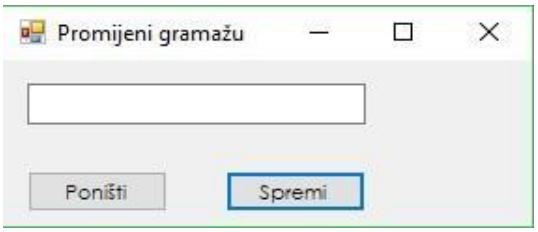
Slika 4.1. prikazuje izgled glavnog izbornika aplikacije koji sadrži korisnikov preporučeni dnevni unos parametara (bjelančevine, masti, omega3, omega6, ugljikohidrati, vlakna), popis svih namirnica i njihov pretraživač, mjesto za unos gramaže odabrane namirnice, provjeru stanja parametara, izračun unesenih namirnica i sl.

Slika 4.2. Prikaz promjene parametara namirnice *Banana*

Slika 4.2. prikazuje izgled izbornika za promjenu parametara pojedine namirnice, koji kada se otvori prikazuje trenutne vrijednosti namirnice koje zatim možemo promijeniti i tu promjenu pohraniti pritiskom na gumb *Spremi*. Ukoliko smo unutar prozora unijeli krive vrijednosti, te ne želimo da se kao takve pohrane, pritiskom na gumb *Odustani* zatvaramo prozor i vrijednosti ostaju nepromijenjene.

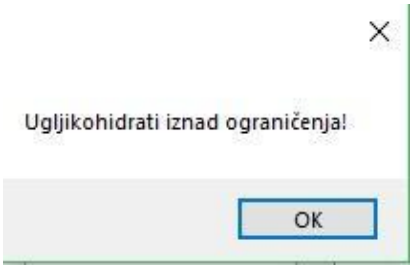
Slika 4.3. Prikaz dodavanja nove namirnice

Slika 4.3. prikazuje izgled izbornika za dodavanje nove namirnice, kod kojega je potrebno popuniti sva polja s odgovarajućim vrijednostima. Ukoliko se taj uvjet ne zadovolji, tada dolazi obavijest u obliku iskočnog prozora. Pritiskom na gumb *Spremi* se prvo provjerava gore navedeni uvjet, te tek kada je on zadovoljen , rezultat se sprema u bazu podataka. Pritiskom na gumb *Odustani* odbacujemo sve unesene vrijednosti, te se zatvara ovaj prozor.



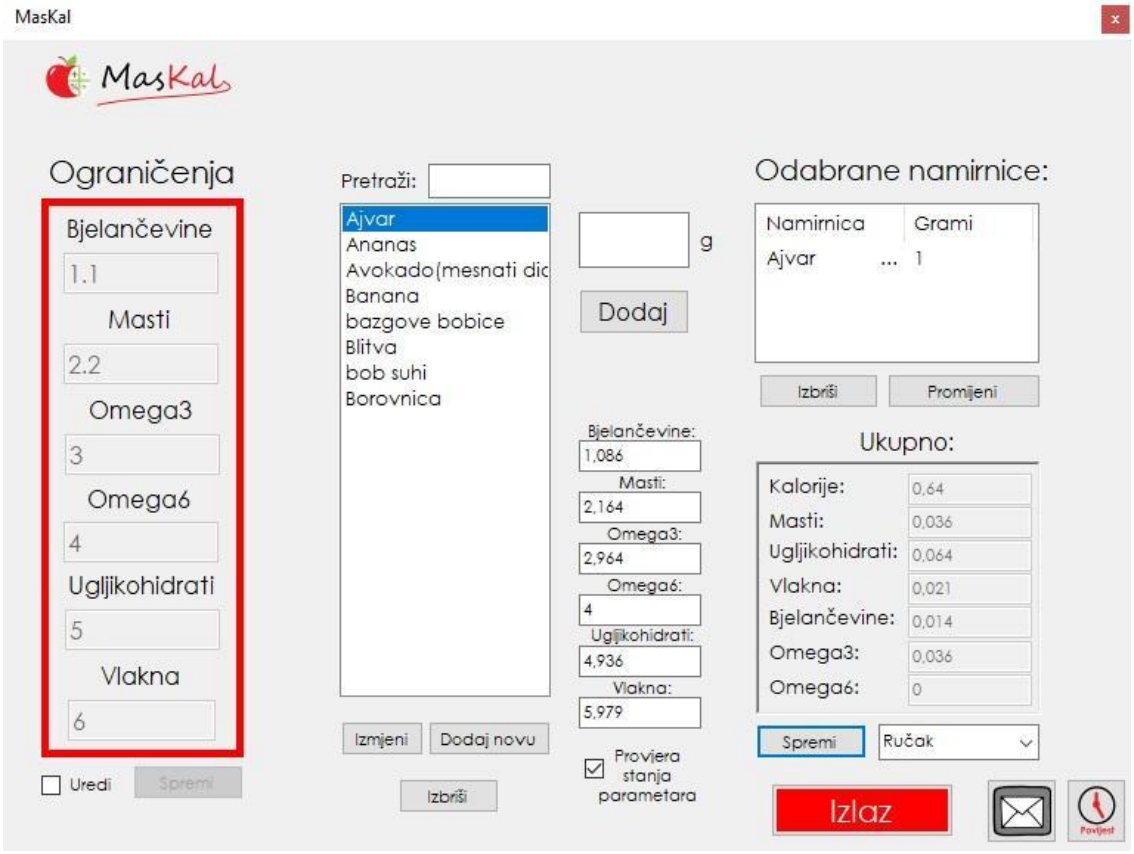
Slika 4.4. Izgled opcije promjene gramaže

Slika 4.4. prikazuje izbornik promjene gramaže u procesu izračuna vrijednosti pojedine namirnice, kod kojega pritiskom na gumb *Spremi* program prvo provjerava je li zadovoljen uvjet unosa, odnosno provjerava da li unos sadrži slova ili je pak prazan. Ukoliko sadržaj ne zadovoljava uvjete, program to javlja u obliku iskočnog prozora, a ako su uvjeti ispunjeni, tada program sprema promjene. Pritiskom na gumb *Poništi* program odbacuje promjene i zatvara ovaj prozor.



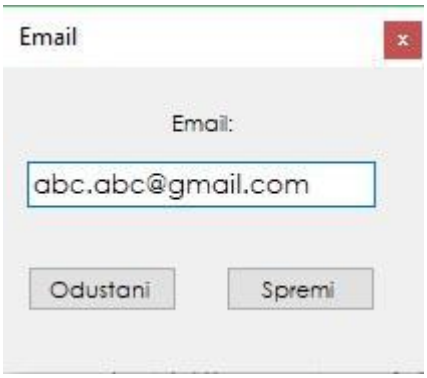
Slika 4.5. Izgled opcije prekoračenje parametara ugljikohidrata

Slika 4.5. prikazuje obavijest koja se pojavljuje u trenutcima kada korisnikov unos prijeđe dnevna ograničenja.



Slika 4.6. Prikaz trenutno stanja parametara

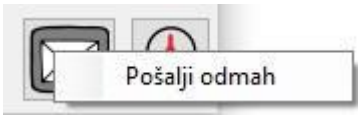
Slika 4.6. prikazuje vrijednosti trenutnog stanja parametara. Pri samom dnu slike možemo vidjeti checkbox *Provjera stanja parametara* koji kada je označen prikazuje trenutno stanje korisnikovih parametara (vrijednosti iznad checkbox-a). Kada checkbox nije označen, tada su ti parametri skriveni.



Slika 4.7. Izgled prozora za unos email-a

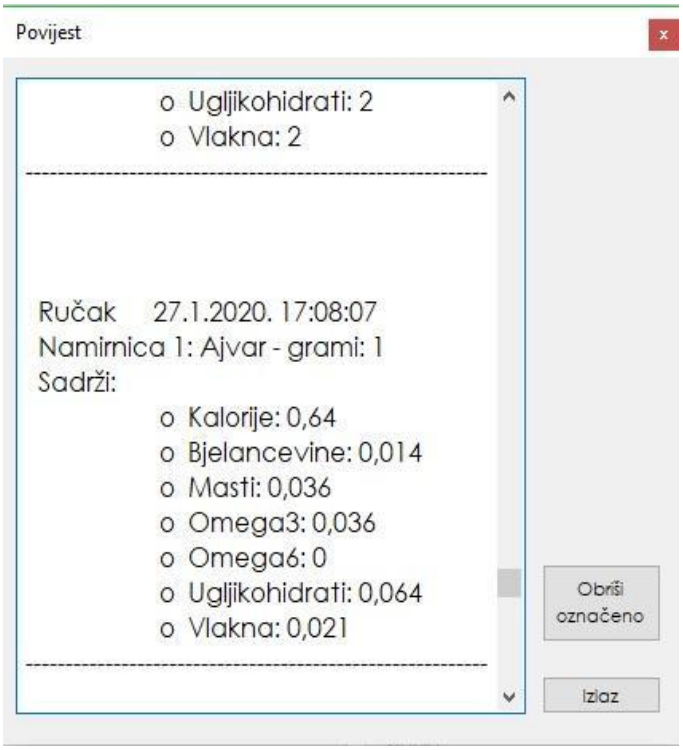
Slika 4.7. prikazuje izgled prozora za unos email adrese na koju će pristizati mjesečno izvješće prehrane korisnika. Pritiskom na gumb *Spremi*, program provjerava je li uneseni mail ispravan, te ukoliko nije, tada korisniku u obliku skočnog prozora dolazi obavijest. Pritiskom na

gumb *Odustani*, prozor *Email* se zatvara i odbacuje se novoupisani tekst.



Slika 4.8. Prikaz opcije proizvoljnog slanja povijesti događaja na email

Slika 4.8. prikazuje gumb email-a preko kojega je desnim klikom miša omogućeno trenutno slanje izvješća prehrane korisnika.



Slika 4.9. Prikaz povijesti upisanih događaja

Slika 4.9. prikazuje izgled upisanih obroka korisnika. U ovome prozoru možemo vidjeti cjelokupan korisnikov unos namirnica. Ukoliko želimo nešto izbrisati, potrebno je označiti dio koji želimo obrisati i pritisnuti na gumb *Obriši označeno*. Pritiskom na gumb *Izlaz* zatvaramo prikazani prozor.

4.2. Osnovne funkcionalnosti aplikacije

Glavni izbornik (slika 4.1) sadrži informacije o svim namirnicama i nudi mogućnost izmjene postojećih parametara: naziv, kalorije, masti, ugljikohidrati, vlakna, bjelančevine, omega 4 i omega 6 (slika 4.2.) ili dodavanja novih namirnica (slika 4.3). Kada se unutar tekstualnog polja namijenjenog za unos grama unese vrijednost i odabere namirnica, pritiskom na gumb *Dodaj* unutar liste, možemo vidjeti unesenu namirnicu, a ispod sadržaj njezinih parametara pri određenoj masi u gramima. Tako možemo kombinirati željene namirnice prije nego što ih odlučimo pohraniti, kako bi mogli izbalansirati njihov unos.

Ukoliko nakon unosa mase odabrane namirnice želimo izmijeniti istu, pritiskom na gumb *Izmjeni* ispod liste unesenih namirnica otvara se novi prozor (slika 4.4), gdje to možemo napraviti. Ako pak želimo ukloniti namirnicu iz popisa unesenih, to možemo učiniti pritiskom na gumb *Izbriši* ispod liste unesenih namirnica.

Kada smo zadovoljni sa unesenim namirnicama i želimo ih pohraniti, tada moramo odabrati jedan od ponuđenih obroka: doručak, ručak, večera, međuobrok, a zatim pritisnuti gumb *Spremi*. Pritiskom gumba *Spremi* sumirani parametri odabranih namirnica oduzimaju se od korisnikovog ograničenja i ukoliko se ta brojka spusti na vrijednost nule ili ispod nje, tada korisniku dolazi obavijest unutar aplikacije putem novoga prozora koji parametri su iznad ograničenja (Slika 4.5.).

Korisnik u svakome trenutku može provjeriti trenutno stanje parametara – bjelančevine, masti, omega3, omega6, ugljikohidrati, vlakna; odnosno unutar kojih vrijednost još može kombinirati svoj obrok, pritiskom na kućicu za označavanje *Provjera stanja parametara* (slika 4.6.). Pritiskom na kućicu za označavanje *Uredi*, ispod korisnikovih ograničavajućih parametara, korisnik može izmijeniti ograničenja i postaviti nova pritiskom na gumb *Spremi*, koji se nalazi odmah s desne strane kućice za označavanje. Pritiskom na gumb sa slikom poruke, korisniku se otvara novi prozor unutar kojeg unosi svoju e-mail adresu, kako bi svakoga prvoga dana u mjesecu dobio mail s prikazom povijesti unosa za cijeli mjesec (slika 4.7.). Ukoliko korisnik želi odmah ispis stanja na email, pritiskom desnoga klika na gumb sa slikom poruke, otvara se ta mogućnost (slika 4.8.). Pritiskom na gumb sa slikom *Povijest* u novom prozoru otvara se povijest upisanih događaja (slika 4.9.). Pritiskom na crveni gumb *Izlaz* korisnik izlazi iz aplikacije.

```
private void btnIzbrisiOdabranuNamirnicu_Click(object sender, EventArgs e)
{
    if (listViewOdabraneNamirnice.TopItem == null)
    {
        MessageBox.Show("Neispravna naredba!!");
        return;
    }
    var selected = listViewOdabraneNamirnice.SelectedItems[0];

    float kalorijeNamirnice = float.Parse(selected.SubItems[8].Text);
    float bjelancevineNamirnice = float.Parse(selected.SubItems[2].Text);
    float mastiNamirnice = float.Parse(selected.SubItems[3].Text);
    float omega3Namirnice = float.Parse(selected.SubItems[4].Text);
    float omega6Namirnice = float.Parse(selected.SubItems[5].Text);
    float ugljikohidratiNamirnice = float.Parse(selected.SubItems[6].Text);
    float vlaknaNamirnice = float.Parse(selected.SubItems[7].Text);

    listViewOdabraneNamirnice.Items[selected.Index].Remove();
    RemoveFromTotal(kalorijeNamirnice, bjelancevineNamirnice, mastiNamirnice, omega3Namirnice, omega6Namirnice,
            ugljikohidratiNamirnice, vlaknaNamirnice);
}
```

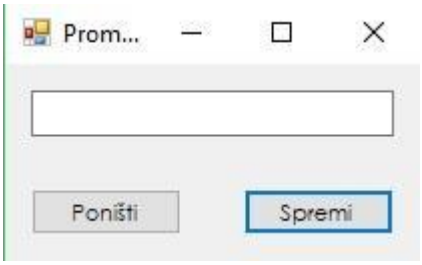
Slika 4.10. Prikaz koda za brisanje namirnice iz liste odabranih namirnica

Kao što možemo vidjeti na slici 4.10. prvo se provjerava je li odabrana namirnica unutar liste i ukoliko nije, tada dolazi poruka u obliku iskočnoga prozora s porukom *Neispravna naredba!!*. Zatim *var* tip podatka *selected* prima vrijednost odabranog itema tj. pokazuje koja je namirnica odabrana. Tada *float* tip podatka *kalorijeNamirnice* prima vrijednost *selected* u obliku teksta i mijenja ga u *float* preko *float.Parse* (pretvara niz znakova u brojčanu vrijednost), a to vrijedi i za sve ostale *float* varijable: *bjelancevineNamirnice*, *mastiNamirnice*, *omega3Namirnice*, *omega6Namirnic*e, *uglikohidratiNamirnice* i *vlaknaNamirnice*. *listViewOdabraneNamirnice.Items[selected.Index]*. *Remove();* znači da će iz liste biti obrisana vrijednost koja je označena unutar liste. Tada se funkciji *RemoveFromTotal* prosljeđuju vrijednosti varijabli: *kalorijeNamirnice*, *bjelancevineNamirnice*, *mastiNamirnice*, *omega3Namirnice*, *omega6Namirnice*, *uglikohidratiNamirnice* i *vlaknaNamirnice* , a ona zatim oduzima njihove vrijednost od ukupnih za trenutni proračun unesenih namirnica.

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

        if (listViewOdabraneNamirnice.TopItem == null)
        {
            MessageBox.Show("Neispravna naredba!!");
            return;
        }


    selectedItem = listViewOdabraneNamirnice.SelectedItems[0].Index;
    FormPromijeniGramazu formPromijeniGramazu = new FormPromijeniGramazu(this);
    formPromijeniGramazu.Owner = this;
    formPromijeniGramazu.Show();
}
```

Slika 4.11. Prikaz koda klika na gumb *PromjeniGramazu*

Slika 4.11. prikazuje da se prilikom klika na gumb *PromjeniGramazu* prvo provjerava postoji namirnica unutar liste *listViewOdabraneNamirnice* i ukoliko ne postoji, pojavljuje se poruka sadržaja *Neispravna naredba!!*. Ukoliko postoji jedna ili više namirnica, tada globalna *static int* varijabla *selectedItem* prima indeks odabrane namirnice. Tada se otvara nova *forma* imena *Promijeni gramazu* koja sadrži gumb *Spremi*, gumb *Poništi* i *textBox* unutar kojega se upisuje vrijednost nove vrijednosti (slika 4.12.).



Slika 4.12. Izgled forme promjene gramaže

Kod forme *Promjena gramaže* je prikazan slikom 4.12., gdje možemo vidjeti da se prvo vrši provjera unesenih vrijednosti unutar *textBoxa textBoxGrami* (slika 4.13.) te ukoliko je *textBox* prazan ili je unesena vrijednost u obliku texta/slova, tada se ispisuje poruka sadržaja *Molim, popunite sva polja s odgovarajućim vrijednostima.*. Potom se izvodi još jedna provjera praznoga unosa u obliku if naredbe i ukoliko je ispravna (nije prazan unos), tada se varijabli *float* tipu podatka *grami* pridodaje vrijednost *textBoxa* i označava se da je „ .“ decimalni odvajač. Tada se funkciji *PromijeniGramazu* prosljeđuje vrijednost varijable *grami*.

```
private void buttonSpremi_Click(object sender, EventArgs e)
{
    var regexItem = new Regex("^[a-zA-Z]*$");
    foreach (char c in textBoxGrami.Text)
    {
        if (c.ToString() == "" || regexItem.IsMatch(c.ToString()))
        {
            MessageBox.Show("Molim, popunite sva polja s odgovarajućim vrijednostima.");
            return;
        }
    }

    if (textBoxGrami.Text.Equals(""))
    {
        return;
    }
    CultureInfo ci = (CultureInfo)CultureInfo.CurrentCulture.Clone();
    ci.NumberFormat.CurrencyDecimalSeparator = ".";
    float grami = float.Parse(textBoxGrami.Text, NumberStyles.Any, ci);
    ownerForm.PromijeniGramazu(grami);
    Close();
}
```

Slika 4.13. Prikaz koda gumba *Spremi* unutar forme *Promjena gramaže*

```
public void PromijeniGramazu(float grami)
{
    var selected = listViewOdabraneNamirnice.Items[selectedItem];

    float kalorijeNamirnice = float.Parse(selected.SubItems[8].Text);
    float bjelancevineNamirnice = float.Parse(selected.SubItems[2].Text);
    float mastiNamirnice = float.Parse(selected.SubItems[3].Text);
    float omega3Namirnice = float.Parse(selected.SubItems[4].Text);
    float omega6Namirnice = float.Parse(selected.SubItems[5].Text);
    float ugljikohidratiNamirnice = float.Parse(selected.SubItems[6].Text);
    float vlaknaNamirnice = float.Parse(selected.SubItems[7].Text);

    RemoveFromTotal(kalorijeNamirnice, bjelancevineNamirnice, mastiNamirnice, omega3Namirnice, omega6Namirnice,
            ugljikohidratiNamirnice, vlaknaNamirnice);

    listViewOdabraneNamirnice.Items[selectedItem].SubItems[1].Text = grami.ToString();
    string cn_string = Properties.Settings.Default.dblNamirniceConnectionString;
    string sql_Text = "SELECT * FROM tbl_Namirnice WHERE Namirnica = '" + listViewOdabraneNamirnice.Items[selectedItem].SubItems[0].Text + "'";
    SqlConnection Conn = new SqlConnection(cn_string);
    Conn.Open();
    SqlCommand Comm1 = new SqlCommand(sql_Text, Conn);
    using (SqlDataReader reader = Comm1.ExecuteReader())
    {
        string kalorije = "", bjelancevine = "", masti = "", omega3 = "", omega6 = "", ugljikohidrati = "", vlakna = "";
        while (reader.Read())
        {

            kalorije = (reader["Kalorije"].ToString());
            bjelancevine = (reader["Bjelančevine"].ToString());
            masti = (reader["Masti"].ToString());
            omega3 = (reader["Omega3"].ToString());
            omega6 = (reader["Omega6"].ToString());
            ugljikohidrati = (reader["Ugljikohidrati"].ToString());
            vlakna = (reader["Vlakna"].ToString());
        }
        kalorijeNamirnice = grami * float.Parse(kalorije);
        bjelancevineNamirnice = grami * float.Parse(bjelancevine);
        mastiNamirnice = grami * float.Parse(masti);
        omega3Namirnice = grami * float.Parse(omega3);
        omega6Namirnice = grami * float.Parse(omega6);
        ugljikohidratiNamirnice = grami * float.Parse(ugljikohidrati);
        vlaknaNamirnice = grami * float.Parse(vlakna);

        listViewOdabraneNamirnice.Items[selectedItem].SubItems[2].Text = bjelancevineNamirnice.ToString();
        listViewOdabraneNamirnice.Items[selectedItem].SubItems[3].Text = mastiNamirnice.ToString();
        listViewOdabraneNamirnice.Items[selectedItem].SubItems[4].Text = omega3Namirnice.ToString();
        listViewOdabraneNamirnice.Items[selectedItem].SubItems[5].Text = omega6Namirnice.ToString();
        listViewOdabraneNamirnice.Items[selectedItem].SubItems[6].Text = ugljikohidratiNamirnice.ToString();
        listViewOdabraneNamirnice.Items[selectedItem].SubItems[7].Text = vlaknaNamirnice.ToString();

        AddToTotal(kalorijeNamirnice, bjelancevineNamirnice, mastiNamirnice, omega3Namirnice, omega6Namirnice,
            ugljikohidratiNamirnice, vlaknaNamirnice);

        Conn.Close();

    }
}
```

Slika 4.14. Prikaz koda funkcije *PromijeniGramazu* unutar glavne forme

Na slici 4.14. *Var* tip podatka *Selected* prima parametre koji nam pokazuju kojoj namirnici mijenjamo gramažu preko globalne varijable *SelectedItem*. Tada varijable *float* tipa podatka: *kalorijeNamirnice*, *bjelancevineNamirnice*, *mastiNamirnice*, *omega3Namirnice*, *omega6Namirnice*, *uglikohidratiNamirnice* i *vlaknaNamirnice* primaju odgovarajuće parametre

odabrane namirnice kojoj mijenjamo gramažu i prosljeđujemo ih funkciji *RemoveFromTotal*, koja zatim oduzima njihove vrijednost od ukupnih vrijednost za trenutni proračun unesenih namirnica. Nadalje, promijenjeni grami upisuju se za odabranu namirnicu i otvara se veza s bazom podataka, te se za označenu namirnicu uzimaju vrijednosti i pridodaju odgovarajućim novim varijablama *string* tipa podatka. Tada se svakoj od varijabli (*kalorijeNamirnice*, *bjelancevineNamirnice*, *mastiNamirnice*, *omega3Namirnice*, *omega6Namirnice*, *uglikohidratiNamirnice* i *vlaknaNamirnice*) dodaje vrijednost varijable *grami* koja se potom množi sa odgovarajućom varijablom *string* tipa podatka, koja se pretvara u float pomoću *float.Parse*. Nakon dobivenog umnoška, označenoj vrijednosti unutar liste *listViewOdabraneNamirnice* dodaju se varijable (*kalorijeNamirnice*, *bjelancevineNamirnice*, *mastiNamirnice*, *omega3Namirnice*, *omega6Namirnice*, *uglikohidratiNamirnice* i *vlaknaNamirnice*), kao njegove vrijednosti u obliku podstringa. Tada se te vrijednosti dodaju ukupnoj vrijednosti za trenutni proračun unesenih namirnica i veza se zatvara.

```
public void provjeri()
{
    string statusParametara = "StatusParametara.txt";
    string putanjaUvjeta = "Uvjet.txt";
    string trenutniDan = DateTime.Now.Day.ToString();
    string trenutniMjesec = DateTime.Now.Month.ToString();
    string trenutnaGodina = DateTime.Now.Year.ToString();
    string trenutniDatum = trenutniDan + "." + trenutniMjesec + "." + trenutnaGodina + ".";
    StreamReader streamReaderUvjet = new StreamReader(putanjaUvjeta);
    if (streamReaderUvjet.ReadLine() != trenutniDatum)
    {
        streamReaderUvjet.Close();
        File.WriteAllText(putanjaUvjeta, trenutniDatum);
        StreamReader streamReader = new StreamReader("podaci.txt");
        string tempBjelancevine = streamReader.ReadLine();
        string tempMasti = streamReader.ReadLine();
        string tempOmega3 = streamReader.ReadLine();
        string tempOmega6 = streamReader.ReadLine();
        string tempUgljikohidrati = streamReader.ReadLine();
        string tempVlakna = streamReader.ReadLine();
        streamReader.Close();

        StreamWriter streamWriterUvjet = new StreamWriter(statusParametara);
        streamWriterUvjet.WriteLine(tempBjelancevine);
        streamWriterUvjet.WriteLine(tempMasti);
        streamWriterUvjet.WriteLine(tempOmega3);
        streamWriterUvjet.WriteLine(tempOmega6);
        streamWriterUvjet.WriteLine(tempUgljikohidrati);
        streamWriterUvjet.WriteLine(tempVlakna);
        streamWriterUvjet.Close();
    }
    StreamReader streamReaderProvjera = new StreamReader(statusParametara);
    textBoxBjelancevine.Text = streamReaderProvjera.ReadLine();
    textBoxMasti.Text = streamReaderProvjera.ReadLine();
    textBoxOmega3.Text = streamReaderProvjera.ReadLine();
    textBoxOmega6.Text = streamReaderProvjera.ReadLine();
    textBoxUgljikohidrati.Text = streamReaderProvjera.ReadLine();
    textBoxVlakna.Text = streamReaderProvjera.ReadLine();
    streamReaderProvjera.Close();
}
```

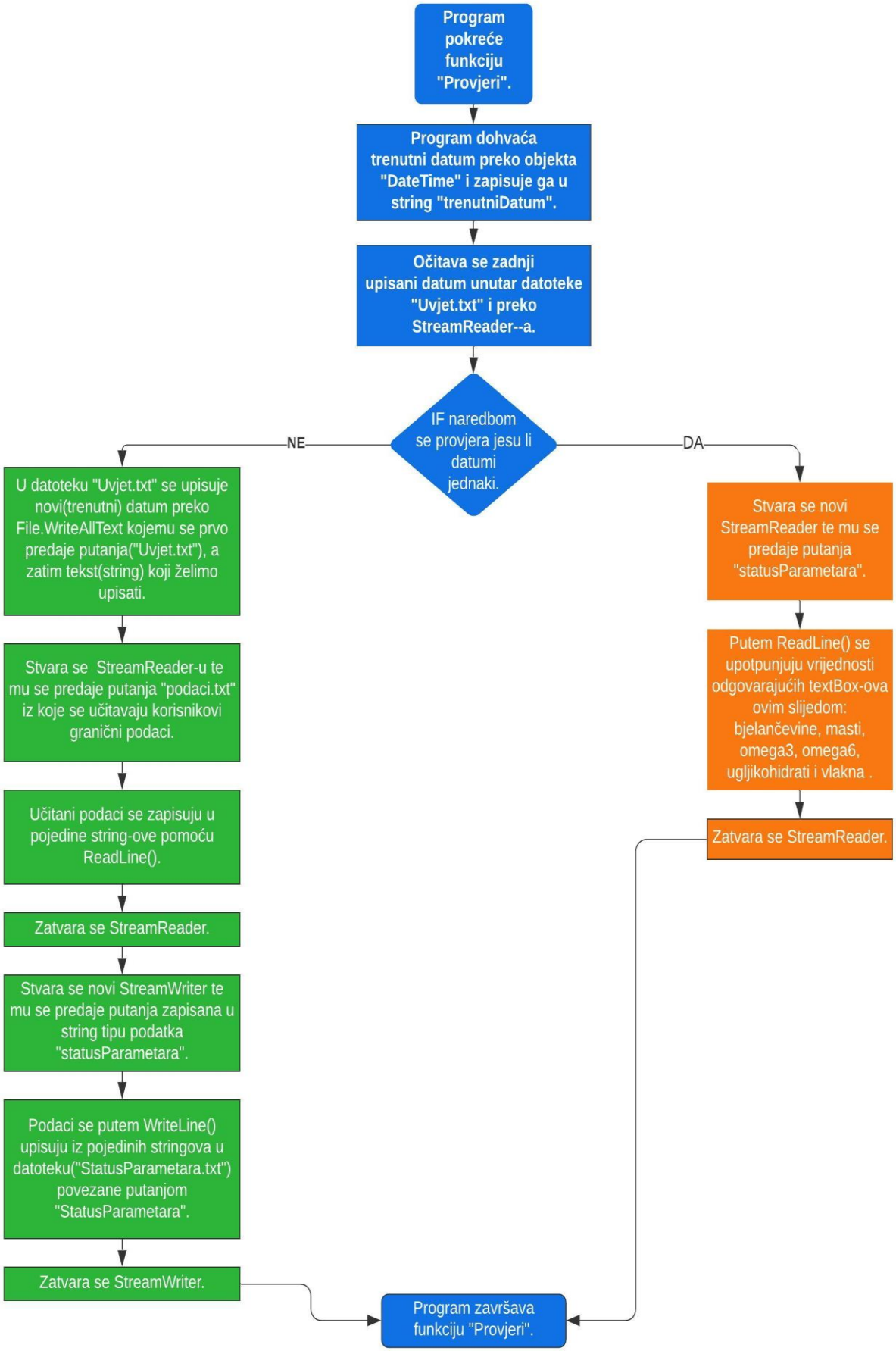
Slika 4.15. Prikaz koda provjere novoga dana unutar glavnog izbornika

Funkcija *provjeri* (slika 4.15.) ima zadaću *reseta* dnevnih vrijednosti koje korisnik može unijeti (opisana dijagramom toka na slici 4.17.), pa tako varijabla *statusParametara* sadrži putanju *statusParametara.txt*, koja ukazuje na korisnikove ograničavajuće parametre bjelančevina, masti, omega3, omega6, ugljikohidrata i vlakana. Uz varijablu *statusParametara* nalazi se i varijabla *putanjaUvjeta* koja sadrži putanju *putanjaUvjeta.txt* koja pokazuje na zadnji zabilježeni datum. *String* tip podatka varijable *trenutniDatum* sadrži tri varijable (*trenutniDan*, *trenutniMjesec* i *trentnaGodina*) odvojene „ .“ koji prikazuju trenutni datum. Tada *StreamReaderu* predajemo varijablu *putanjaUvjeta* i preko *ReadLine()* očitavamo vrijednost koja se u toj txt datoteci nalazi te je uspoređujemo sa varijablom *trenutniDatum*.

Ukoliko se trenutni datum razlikuje od zadnjeg zabilježenog, *streanReader* se zatvara i novi datum se zapisuje kao zadnji zabilježeni. Potom se *streamReaderu* predaje putanja koja vodi do korisnikovih parametara graničnih vrijednosti, te se očitaju putem *ReadLinea()* i zapišu u odgovarajuće nove varijable *string* tipa podatka ovim slijedom: bjelančevine, masti, omega3, omega6, ugljikohidrati i vlakna, a *streamReader* se zatrvara. Potom se kreira novi *streamWriter* kojem se predaje varijabla *statusParametara* kao putanja i putem *WriteLine()* zapisuju se parametri točno ovim redoslijedom: *tempBjelančevine, tempMasti, tempOmega3, tempOmega6, tempUgljikohidati* i *tempVlakna* i zatvara se *streamWriter*. Ukoliko je zadnje zabilježeni datum isti kao i trenutni, tada se kreira novi *streamReader* koji prima varijablu *statusParametara* kao putanju i putem *ReadLine()* upotpunjuje vrijednostima odgovarajuće *textBox*-ove ovim slijedom: bjelančevine, masti, omega3, omega6, ugljikohidrati i vlakna te se zatvara *streamReader*. Ukoliko su vrijednosti jednake nuli ili manje od nule tada dolazi obavijest u obliku *MessageBoxa* za svaki parametar koji zadovoljava taj kriterij (slika 4.16).

```
private void CompareToLimit()
{
    if (float.Parse(textBoxBjelancevine.Text.ToString()) <= 0)
    {
        MessageBox.Show("Bjelančevine iznad ograničenja!");
    }
    if (float.Parse(textBoxMasti.Text.ToString()) <= 0)
    {
        MessageBox.Show("Masti iznad ograničenja!");
    }
    if (float.Parse(textBoxOmega3.Text.ToString()) <= 0)
    {
        MessageBox.Show("Omega3 iznad ograničenja!");
    }
    if (float.Parse(textBoxOmega6.Text.ToString()) <= 0)
    {
        MessageBox.Show("Omega6 iznad ograničenja!");
    }
    if (float.Parse(textBoxUgljikohidrati.Text.ToString()) <= 0)
    {
        MessageBox.Show("Ugljikohidrati iznad ograničenja!");
    }
    if (float.Parse(textBoxVlakna.Text.ToString()) <= 0)
    {
        MessageBox.Show("Vlakna iznad ograničenja!");
    }
}
```

Slika 4.16. Primjer koda funkcije *CompareToLimit*

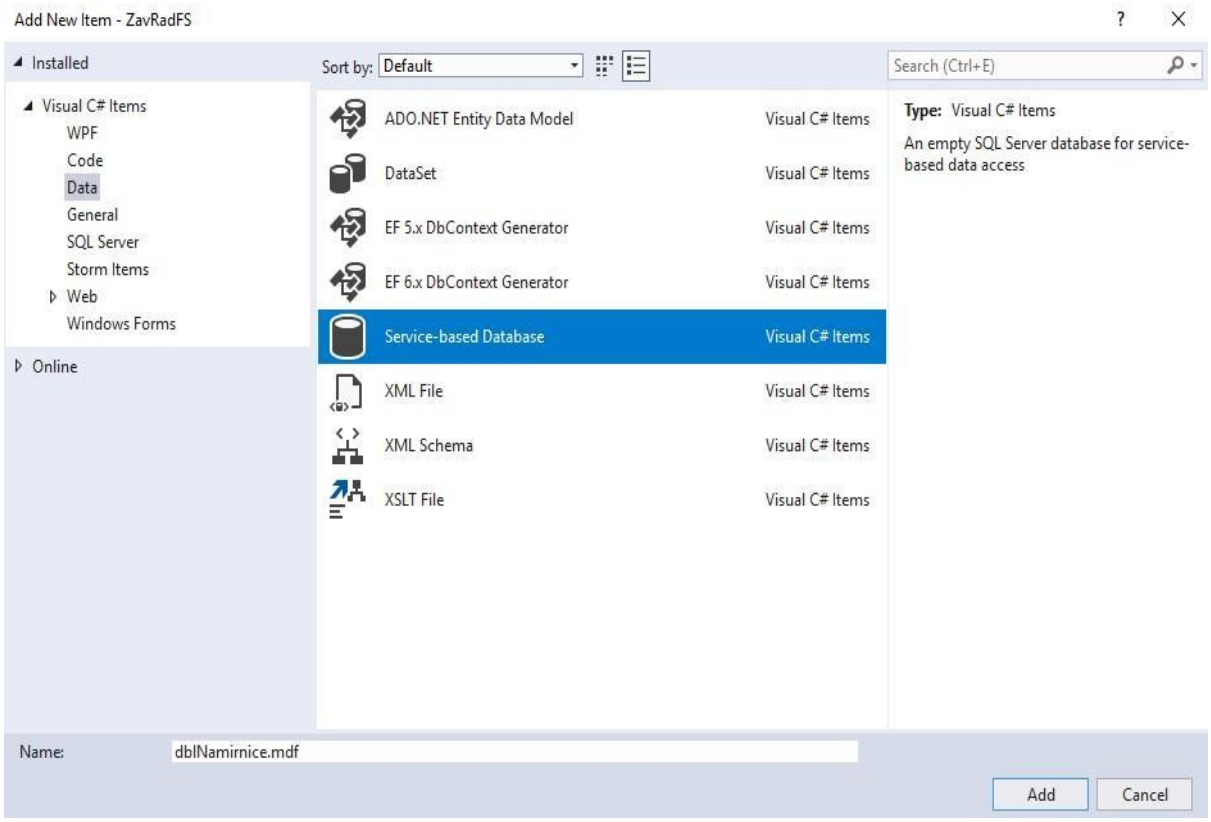


Slika 4.17. Dijagram toka funkcije *provjeri*

4.3. Baza podataka

Baza podataka je potrebna kako bismo pohranili sve namirnice te kako bismo mogli napraviti sve potrebne računske operacije. Osim osnovnih podataka namirnica, nužno je omogućiti dodavanje novih te izmjena postojećih. Upravo zato, kreirana je lokalna baza podataka preko Microsoft Visual Studia u sklopu aplikacije završnog rada kao MDF tip podatka. Postupak za izradu na već kreiranom projektu Windows Form App:

1. Na izbornoj traci odabiremo *Project* > *Add New Item*
2. Na popisu predložaka pod *Data* odabiremo *Service-based Database* (slika 4.18.)
3. Unosimo željeno ime (*dblNamirnice*) i tada kliknemo na *Add*



Slika 4.18. *Service-based Database*

Potom kreiramo sadržaj tablice:

1. Sada pod *Server Explorer* proširimo čvor *Data Connection*, a zatim proširimo i *dblNamirnice.mdf* čvor
2. Desnim klikom miša na *Table* odaberemo *Add New table*
3. Tada dodajemo odgovarajuće entitete (slika 4.19.)

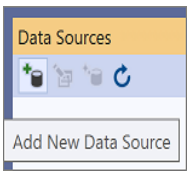
	Name	Data Type	Allow Nulls	Default
🔑	Id	int	☐	
	Namirnica	nchar(50)	☑	
	Kalorije	float	☐	
	Bjelančevine	float	☐	
	Masti	float	☐	
	Omega3	float	☐	
	Omega6	float	☐	
	Ugljikohidrati	float	☐	
	Vlakna	float	☐	
			☐	

Slika 4.19. Prikaz entiteta unutar tablice

4. Postavljamo na ID primarni ključ i desnim klikom na ID odabiremo *Properties* i pod *Identity Specification* dvostruko kliknemo na *is Identity* da bude *True*
5. Tada u gornjem lijevom kutu odabiremo *Update*
6. Kada se otvori novi prozor pritisnemo *Update Database.*

Potom povezujemo bazu podataka sa projektom:

1. U *Data Sources* prozoru odabiremo *Add New Data Source* (slika 4.20)



Slika 4.20. Prikaz *Data Sources*-a

2. Tada se otvara *Data Source Configuration Wizard* te odabiremo *Database* i ¸ zatim kliknemo na *Next*
3. Na *Choose a Database Model* prozoru kliknemo na *Next* za prihvaćanje osnovne opcije, a to je *Dataset*
4. Tada u padajućem izborniku odabiremo *dblNamirnice.mdf* i klinemo na *Next*
5. Potom unutar *Save the Connection String to the Application Configuation File* kliknemo na *Next*
6. Unutar prozora *Choose Your Database Objects* na checkbox-u kliknemo na *Tables* i na *Views* , a zatim kliknemo na *Finish.*

Nakon što se kreirala baza podataka, sada joj možemo pristupiti tako što stringu *cn_ctring* predamo vezu (*Properties.Settings.Default.dblNamirniceConnectionString.ToString();*) koju zatim uvrštavamo unutar *SqlConnection* i otvorimo konekciju putem naredbe .*Open()*. Zatim

stvaramo sql naredbu pomoću *SqlCommand* kojem kao prvi parametar predajemo ono što želimo da se unutar baze podataka napravi i zatim kao drugi parametar predajemo prethodno stvorenu sql konekciju. Tada ovisno o upitu upisujemo jednu od naredbi (*ExecuteScala*r, *ExecuteReader* i *ExecuteNonQuery*) te zatvaramo konekciju putem *SqlConnection* naredbe *.Close().*

5. ZAKLJUČAK

Danas, kada se tehnologija razvija tako velikom brzinom, svjedoci smo svakodnevnih patenata i noviteta kojima čovjek nastoji olakšati svakodnevni život. Aplikacija izrađena u sklopu ovog završnog rada namijenjena je vrlo uskom krugu ljudi te predstavlja novost na našem tržištu.

Osobe oboljele od VLCAD moraju svakodnevno same računati unos bjelančevina, masti, vlakana, omega-3 i omega-6 masnih kiselina te ugljikohidrata pomoću tablica s namirnicama i podacima o tome koliko određena gramaža namirnica sadrži gore navedenih parametara. Sukladno tome, određuju kolike su dnevne vrijednosti parametara unijeli putem konzumirane hrane. Postupak izračuna ponavlja se pred svaki sljedeći obrok, kako bi unos bio optimalan s obzirom na individualne potrebe. Obzirom na složenost ovakve prakse u svakodnevnom životu, osmišljena je aplikacija koja pomaže olakšati računanje i praćenje dnevnih vrijednosti svih potrebnih parametara. Korisniku je omogućen unos novih namirnica, izmjene parametara postojećih namirnica kao i brisanje postojećih namirnica. Također, korisnik svakog prvog u mjesecu dobiva izvještaj o unesenim parametrima tijekom proteklog mjeseca i posjeduje pregled sljedećih informacija o svakom obroku:

- Sastav namirnica
- Tip obroka
- Datum i vrijeme unosa
- Ukupan iznos bjelančevina, masti,vlakana,omega3, omega6 i ugljikohidrata.

Uz standardnu evidenciju svakog prvog u mjesecu, korisnik može u svakome trenutku poslati izvještaj s vrijednostima o proteklim obrocima na mail, kako bi lakše kontrolirao svoju prehranu. Korištenje aplikacije olakšava posao i samim liječnicima, koji u svakome trenutku mogu provjeriti prehranu pacijenata i pravovremeno reagirati ukoliko je to potrebno.

Aplikacija je izrađena u programskom jeziku C# zbog svoje jednostavnosti i efikasnosti izrade vizualnoga dijela aplikacije, koja je neophodna za ovaj zadatak. Uz samo kodiranje aplikacije, važan dio predstavlja i baza podataka, izrađena pomoću *Microsoft SQL Servera*, unutar koje se nalaze sve namirnice s pripadajućim parametrima. Na taj način je olakšana njihova pohrana i korištenje unutar same aplikacije. Za razvoj aplikacije korištenja su znanja stečena na kolegijima Programiranje I, Programiranje II, Dizajn korisničkog sučelja i Baze podataka, ali i znanje stečeno tijekom Stručne prakse.

Iskustvo i vještine stečene tijekom izrade ove aplikacije su vrlo korisne i lako se mogu primijeniti za izradu puno složenijih projekata. Što se tiče daljnjih poboljšanja, aplikacija bi mogla imati napredniji dizajn korisničkog sučelja, kao i unos parametara dobivenih od liječnika. Na taj način korisnik bi izbjegao računanje vrijednosti parametara za iznos jednoga grama namirnice.

LITERATURA

[1] Genetic and Rare Disease Information Center, „*VLCAD deficienty*“, https://rarediseases.info.nih.gov/diseases/5508/vlcad-deficiency, pristup ostvaren 23.lipnja 2019.

[2] Leslie, N.D. i sur. 2019. Very Long-Chain Acyl-Coenzyme A Dehydrogenase Deficiency, GeneReviews

[3] Generic Home Reference, „*Very long-chain acyl-CoA dehydrogenase deficiency*“, https://ghr.nlm.nih.gov/condition/very-long-chain-acyl-coa-dehydrogenase-deficiency#definition, pristup 28.6.2019.

[4] MSD priručnik, „*Poremećaji ciklusa β-oksidacije*“, http://www.msd-prirucnici.placebo.hr/msd-prirucnik/pedijatrija/nasljedne-metabolicke-bolesti/poremecaji-ciklusa-beta-oksidacije, pristup 28.6.2019.

[5] „Acute illness protocol fatty acid oxidation disorders very long chain acyl coa dehydrogenase (vlcadd) deficiency“, https://newenglandconsortium.org/protocols/acute_illness/fatty-acid-oxidation-disorders/VLCADD.pdf, pristup ostvaren 28.6.2019.

[6] „C# compilation process“, https://codeasy.net/lesson/c_sharp_compilation_process, pristup ostvaren 1.4.2020.

[7] GoalKicker.com, *„C# Note for Professional Books“*, https://books.goalkicker.com/CSharpBook/, pristup ostvaren 25.lipnja 2019.

[8] Rob Miles, *„C# Programming Yellow Book“*, https://static1.squarespace.com/static/5019271be4b0807297e8f404/t/5824ad58f7e0ab31fc216843/1478798685347/CSharp+Book+2016+Rob+Miles+8.2.pdf, pristup ostvarem 25.lipnja 2019.

[9] Wiley Publishing, *„The .NET arhitecture“*, https://www.codeguru.com/csharp/sample_chapter/article.php/c8245/The-NET-Architecture.htm, pristup ostvaren 23.lipnja 2019.

[10] tutorialIspoint.dev, „C# | .NET Framework (Basic Arhitecture and Component Stack“, https://tutorialspoint.dev/language/c-sharp/c-net-framework-basic-architecture-component-stack, pristup ostvaren 27.1.2020.

[11] SQLServer Tutorial.net, „What is SQL Server“, https://www.sqlservertutorial.net/getting-started/what-is-sql-server/, pristup ostvaren 27.1.2020.

SAŽETAK

Naslov: C# desktop aplikacija za izračunavanje udjela kalorija, masti, ugljikohidrata, bjelančevina i šećera – MasKal

Cilj ovog završnog rada je izraditi C# aplikaciju koja će pomoći osobama sa VLCAD deficitom s izračunom dnevnog unosa kalorija, masti, ugljikohidrata, bjelančevina i šećera. Na početku rada je opisan VLCAD deficit i izazovi s kojima u svakodnevnom životu susreću osobe oboljele od ove bolesti. Zatim je opisan način rada aplikacije i njezine mogućnosti. Nadalje, ukratko su opisane tehnologije koje se koriste u izradi ove aplikacije. Potom je detaljno opisan kod pojedinih funkcionalnosti i sam postupak izrade baze podataka unutar koje se nalaze vrijednosti svih namirnica koje aplikacija koristi.

Ključne riječi: C#, izračun, parametri, pohrana podataka, VLCAD.

ABSTRACT

Title: C # desktop application for calculating calories, fat, carbohydrates, proteins and sugars - MasKal

The goal of this project was to develop an C# application which will help people with VLCAD deficiency calculate their daily intake of calories, fats, carbohydrates, proteins and sugars. At the beginning of the project is described the VLCAD deficit and what challenges people with this disease face in their daily lives. Then the application mode of work and its capabilities are described. Furthermore, the technologies used to create this application are briefly described. Then, the functionalities and the process of creating a database that contain the values of all the foods the application uses are described in detail.

Key words: C#, calculation, data storage, parameters, VLCAD.

ŽIVOTOPIS

Filip Šangut rođen je 12. siječnja 1998. godine u Virovitici. Osnovnoškolsko obrazovanje započinje 2004. godine u OŠ Vladimir Nazor u Daruvaru, a nakon toga upisuje Tehničku školu, smjer Tehničar za računalstvo u Daruvaru. Tijekom školovanja sudjeluje u brojnim športskim i predmetnim aktivnostima. Akademske godine 2016/2017 upisuje Stručni sveučilišni studij elektrotehnike, smjer Informatika na Fakultetu elektrotehnike, računarstva i informacijskih tehnologija u Osijeku. Tijekom studiranja sudjeluje u nekolicini studentskih aktivnosti poput natjecanja u studentskom natječaju Pro-Student (Kompot), natjecanje u pisanju studentskih poslovnih planova Budi uzor, ali i mnogim drugim vezanima za sport, poput Sveučilišnog prvenstva u rukometu za studente.