ДИПЛОМЕН ПРОЕКТ ЗА ДЪРЖАВЕН ЗРЕЛОСТЕН ИЗПИТ

по професия код 481030 „Приложен програмист“

специалност код 4810301 Приложно програмиране“

Тема: „Настолно приложение за фитнес и здравословно хранене“

Автор:

Антон Славчев Славов, клас XII В

Ръководител:

Красимир Ватев

БургасСЪДЪРЖАНИЕ

[1 Увод 3](#_Toc119855807)

[2 Цели и обхват на софтуерното приложение 3](#_Toc119855808)

[3 Анализ на решението 3](#_Toc119855809)

[3.1 Потребителски изисквания и работен процес 3](#_Toc119855810)

[3.2 Примерен потребителски интерфейс 5](#_Toc119855811)

[3.3 Диаграми на анализа 5](#_Toc119855812)

[3.4 Модел на съдържанието / данните 5](#_Toc119855813)

[4 Дизайн 6](#_Toc119855814)

[4.1 Реализация на архитектурата на приложението 6](#_Toc119855815)

[4.2 Описание на слоевете, предназначението им, библиотеки и методи включени в съответния слой. 6](#_Toc119855816)

[4.3 Организация и код на заявките към база от данни 6](#_Toc119855817)

[4.4 Наличие на потребителски интерфейс (конзолен, графичен, уеб) 6](#_Toc119855818)

[5 Ефективност и бързодействие на решението 6](#_Toc119855819)

[6 Тестване 6](#_Toc119855820)

[7 Заключение и възможно бъдещо развитие 6](#_Toc119855821)

[8 Използвани литературни източници и Уеб сайтове 7](#_Toc119855822)

[9 Приложения 7](#_Toc119855823)

[10 Критерии и показатели за оценяване 8](#_Toc119855824)

# Увод

Идеята на проекта за приложение за фитнес и здравословно хранене е да предостави на потребителите лесен и удобен начин да поддържат здравословен начин на живот. Чрез приложението потребителите могат да изберат подходяща диета и тренировки, да следят своя прогрес и да получават съвети за здравословен начин на живот. Това може да помогне на потребителите да постигнат своите цели за физическа форма и здраве.

Основната идея на проекта е да съчетае функционалността на приложението с лесно използваем интерфейс и качествена информация, която да помага на потребителите да постигнат своите цели.

* Проектът за приложение за фитнес и здравословно хранене е изключително актуален в днешно време, когато все повече хора се стремят да живеят по-здравословен начин на живот. Здравословното хранене и физическата активност са от решаващо значение за поддържането на здравето и доброто самочувствие на човека.
* Приложението за фитнес и здравословно хранене решава редица проблеми, свързани с управлението на здравословния начин на живот на потребителите. Някои от тези проблеми включват:

1. Недостатъчна информация за здравословното хранене: Мнозина хора не разполагат с достатъчно знания за здравословното хранене и как да изградят здравословни хранителни навици. Приложението предоставя информация и съвети за здравословно хранене, като помага на потребителите да изберат правилните храни и да съставят здравословни диети.
2. Трудности при мониторинга на прогреса: За да се подобри здравословния начин на живот, е важно да се мониторира прогреса на потребителя. Приложението предоставя инструменти за мониторинг на физическата активност и консумацията на храни, които да помогнат на потребителите да следят своя прогрес и да се вдъхновят да постигнат по-високи цели.
3. Трудности при мониторинг на консумация на храна: Мониторингът на консумацията на храни може да бъде труден за много хора поради липсата на информация за количествата на храни, които консумират и за съдържанието на хранителните съставки в тях. Това може да доведе до неправилно избрани диети и крайни резултати, които не отговарят на желанията на потребителя.
4. Приложенията за мониторинг на консумацията на храни, като например приложението за фитнес и здравословно хранене, предлагат лесен и ефективен начин за мониторинг на консумацията на храни. Потребителите могат да въведат хранителните продукти, които консумират, и да получат детайлна информация за техните хранителни съставки, като калории, макронутриенти и други. Това им помага да правят по-добри избори за здравословно хранене и да съставят персонализирани диети, съобразени с техните нужди и цели.
5. Изборът на правилните тренировки може да бъде труден за много хора поради липсата на знания или опит в областта на фитнеса. Това може да доведе до забавяне на постигането на желаните резултати или дори до появата на контузии и наранявания. Приложенията за фитнес и здравословно хранене могат да помогнат на потребителите да изберат правилните тренировки за техните нужди и цели. Много от тези приложения предлагат персонализирани програми за тренировки, които са съобразени с тяхното ниво на фитнес, целите, които искат да постигнат, и наличното им време.

Следователно, проектът за приложение за фитнес и здравословно хранене е високо актуален и може да предостави ценни услуги на хората, които се стремят към по-здравословен начин на живот.

* Lucidchart е уеб базиран софтуер за създаване на диаграми, който предоставя голям избор от шаблони, включително за UML диаграми. Това прави процеса на създаване на диаграми на UML много по-лесен и достъпен за потребителите, които не разполагат с много опит в тази област. В крайна сметка, използването на онлайн инструменти като Lucidchart може да помогне за решаване на проблема със създаването на UML диаграми и да улесни процеса на проектиране на софтуерните системи за приложението.
* Структурата на документацията за проекта за фитнес и здравословно хранене включва следните елементи:

1. Увод - включва общо описание на проекта, целите му и очакваните резултати.
2. Цели и обхвати – описание на търсените резултати за потребителите на приложението и последиците за самите тях.
3. Анализ за решението – съдържа описание на работния процес като вход, обработка и изход.
4. Дизайн - този раздел включва детайлно описание на дизайна на потребителския интерфейс на приложението, включително изгледа и усещането на приложението, потребителските преживявания и други важни аспекти.
5. Ефективност и бързодействие на решението – Съдържа описание и анализ на изпълнението на приложението и факторите, от които зависи скоростта му.
6. Тестване - тестване на приложението и други процеси, свързани с разработката, както и визуална репрезентация на тестовете.
7. Заключение и бъдещо развитие – Обобщение на резултатите и работата по проекта както и описание на използваемостта на подобни решения в практиката и какво предложения като за бъдещо развитие на приложението.
8. Използвани литературни източници и уебсайтове – Съдържа препратки към използвани външни източници за информация и ресурси.
9. Приложения – Прикачена допълнителна информация под формата на диаграми, таблици, графики, апендикси и други.
10. Критерии и показатели за оценяване – точкуване на различни аспекти от проекта например:
    1. Съответствие с изискванията за съдържание и структура на дипломния проект
    2. Съответствие между поставените цели на дипломния проект и получените резултати
    3. Представяне на дипломния проект
    4. Отговори на зададените въпроси от рецензента и/или членовете на комисията за защита на дипломен проект
    5. Използване на професионалната терминология, добър и ясен стил, обща езикова грамотност

# Цели и обхват на софтуерното приложение

Целта на проекта е да се разработи приложение за фитнес и здравословно хранене, което да предоставя на потребителите персонализирана информация и съвети, свързани с тренировки и хранене. Приложението ще бъде предназначено за хора, които търсят начини за подобряване на физическата си форма и здравословен начин на живот.

Основните функционалности на приложението включват предоставяне на информация за хранителните стойности на храни и напитки, препоръки за дневни количества на макронутриенти и калории, както и функционалност за проследяване на консумацията на храни и напитки, както и на физическата активност.

Освен информацията за храненето, приложението предоставя и персонализирани препоръки за тренировки, които ще отговарят на нуждите и целите на всеки потребител. Това ще даде възможност на потребителите да получат максимален ефект от тренировките си и да постигнат по-бързо своите цели.

Друга важна функционалност на приложението е възможността за проследяване на консумацията на храни и напитки, както и на физическата активност. Това ще помогне на потребителите да следят дали следват своите цели и да правят корекции в тренировъчния и хранителния си режим, ако е необходимо.

Приложението съдържа база данни с информация за хранителните стойности на храни и напитки, което ще позволи на потребителите да получават детайлна информация за това, какво съдържат продуктите, които консумират. Това ще помогне на потребителите да взимат по-добри решения относно храненето си и да се уверят, че консумират правилното количество хранителни вещества.

Обхватът на проекта включва следните функционалности:

* + 1. Регистрация на потребители: потребителите ще могат да се регистрират в приложението и да създадат свой профил.
    2. Персонализирани тренировки: приложението ще предлага на потребителите персонализирани тренировки, в зависимост от техните цели, физическа форма и опит.
    3. Хранителен дневник: потребителите ще може да записват информация за храненето си и да получават съвети за здравословна храна.
    4. Информация за храни: приложението ще предоставя информация за храни и хранителни стойности, както и рецепти за здравословни ястия.
    5. Съвети за здравословен начин на живот: потребителите ще получават съвети за здравословен начин на живот, свързани с тренировки, хранене и общо благополучие.
    6. Социална функционалност: приложението ще предоставя възможности за споделяне на информация и взаимодействие между потребителите.
    7. Анализ на напредъка: приложението ще предоставя анализ на напредъка на потребителите, като ще отчита промените в техните физически данни, тренировъчен режим и хранене.

На кратко приложението е лесно за използване и предоставя полезна информация и съвети, които да помогнат на потребителите да постигнат своите цели за фитнес и здравословен начин на живот.

# Анализ на решението

## Потребителски изисквания и работен процес

Diagram

Description automatically generated

Фигура 1 – Диаграма на случаи на употреба

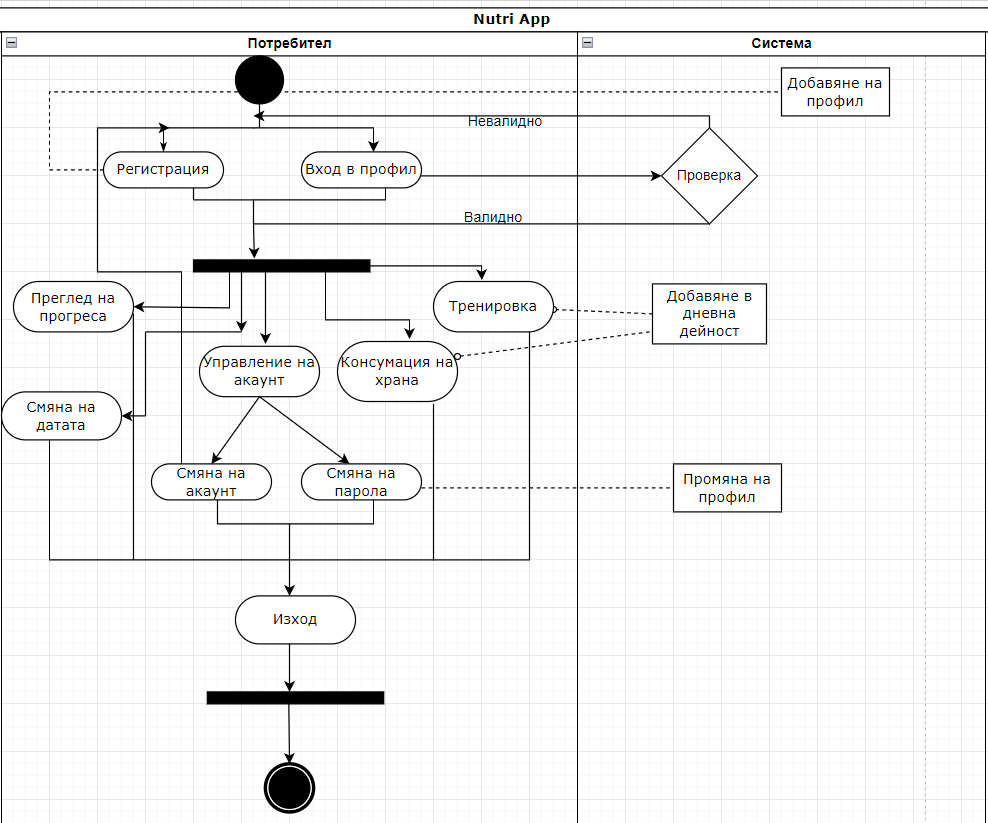
При отваряне на приложението потребителят може да се впише в своя акаунт или да създаде нов. Ако потребителят въведе грешна информация при опит за вход в своя акаунт, приложението ще изведе прозорец със съобщение за грешка и ще му позволи да опита отново да въведе коректните данни. След това потребителят има пет опции. Тренировка, консумация на храна, преглед на прогреса, управление на акаунт и промяна на дата.

При избор на опцията "Тренировка", потребителят може да търси и избере тренировка от различни категории, като например кардио, сила или гъвкавост. След като потребителят избере тренировка, приложението ще запише извършването ѝ в дневната дейност.

При избор на опцията "Консумация на храна", потребителят може да търси храни по категории и да въвежда консумираните от него храни за даден ден. Приложението ще му предоставя информация за калориите, макронутриентите и други хранителни вещества, които той приема.

При избор на опцията "Преглед на прогреса", потребителят може да види своя прогрес в тренировки и консумация на храна. Приложението предоставя графики и статистика, които помагат на потребителя да следи своя напредък.

При избор на опцията "Управление на акаунт", потребителят може да редактира информацията си, да промени паролата си или да се впише в друг акаунт.

Опцията за промяна на датата позволява на потребителя да избере нова дата, на която да се въведат тренировките и консумацията на храна. Това е полезно в случаи, когато потребителят не е могъл да извърши тренировка в определен ден и иска да я премести на друг ден. Потребителят може да въведе нови тренировки за тази дата, да ги редактира или да ги изтрие. Това позволява на потребителя да контролира своя тренировъчен график и да го променя в зависимост от своите нужди и предпочитания.

Фигура 2 – Диаграма на дейностите

Входът на приложението е информацията, която потребителят въвежда при избор на опция, като например избор на тренировка или консумирана храна. Обработката включва търсене на информацията в базата данни на приложението, изчисляване на необходимата информация като калории, макронутриенти и други хранителни вещества, както и предоставяне на резултатите на потребителя. Изходът на приложението е информацията, която се предоставя на потребителя, като например описание на тренировка, информация за консумирана храна или статистика за прогреса в тренировки и консумация на храна.

## Примерен потребителски интерфейс

Graphical user interface, application, Teams

Description automatically generated

Фигура 3 – Страница за регистрация

Страницата за регистрация позволява на нови потребители да създадат акаунт в приложението. На страницата са предоставени полета за въвеждане на име и парола. Потребителят трябва да попълни полетата, като системата ще провери дали информацията е валидна и дали потребителското име е уникално. Ако има грешки, потребителят ще получи съобщение за грешка и ще трябва да коригира въведената информация. Например за несъответваща парола, което е показано във фигура 4.

Graphical user interface, application

Description automatically generated

Фигура 4 – Съобщение за несъответваща парола

Graphical user interface, application, Teams

Description automatically generated

Фигура 5 – Страница за вход в профил

Страницата за вход в профил на приложението предоставя форма за въвеждане на потребителско име и парола. Потребителят трябва да въведе своите регистрационни данни, за да влезе в системата и да получи достъп до функционалностите на приложението.

Ако потребителското име и/или паролата са невалидни, приложението ще изведе подходящо съобщение за грешка и потребителят ще трябва да въведе валидни данни, за да продължи. Ако потребителят не е регистриран, има възможност да създаде нов акаунт чрез линк към страницата за регистрация.

Graphical user interface, application

Description automatically generated

Фигура 6 – Бутон за показване на парола

Бутонът за показване на паролата е функционалност, която позволява на потребителя да види символите, които е въвел в полето за парола, вместо да се показва само замаскираният текст с точки. Това може да бъде полезно, когато потребителят иска да се увери, че е въвел правилната парола, преди да я потвърди. Въпреки това, тази функционалност може да бъде рискована, ако се използва на обществени места, където може да има други хора, които да гледат екрана на потребителя.

Graphical user interface, application

Description automatically generated

Фигура 7 – Начална страница

Началната страница на приложението е първото нещо, което потребителят вижда след като отвори приложението. Тази страница има за цел да предостави на потребителя достъп до основните функционалности на приложението чрез бутони с ясни и разбираеми надписи.

Тези бутони водят до различните раздели на приложението - страница за тренировки, страница за консумация на храна, страница за преглед на прогреса, страница за управление на акаунта и страница за промяна на датата.

Целта на началната страница е да предостави лесен и удобен начин за навигация към различните функционалности на приложението и да помогне на потребителите да започнат своята здравословна пътека.

## Диаграми на анализа

Diagram, schematic

Description automatically generated

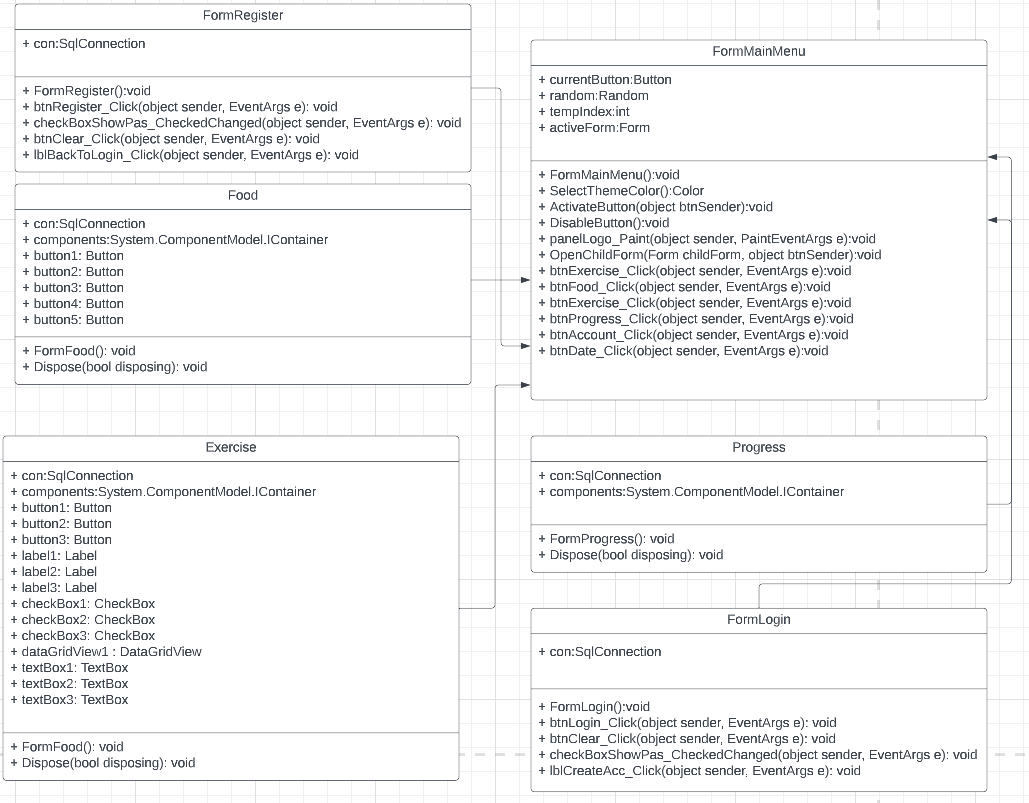
Фигура 8 – ER диаграма

Таблица „Users“ е във връзка едно към много с таблица „UserProgress“. „Users“ съдържа потребителските данни за достъп до профилите. Има първичен ключ, който е потребителското име и поле с потребителксата парола.

Таблица „UserProgress“ е във връзка едно към едно или много с другите две таблици, „FoodItems“ и „Exercises“. Има първичен ключ за датата на прогреса и вторични ключове за потребителското име, на когото е прогреса. Както и вторични ключове за консумираната храна, изпълнените упражнения и името на потребителя. Други полета са количеството храна, продължителността на тренировката и изгорени и консумирани калории. Таблицата съдържа информацията, която е показана в страницата за прогрес.

Таблица „FoodItems“ има един първичен ключ „FoodId“ и съдържа информация за различните видове храна, която потребителите могат да консумират. Други полета са името на храната, мазнините, които съдържа, както и въглехидрати и белтъчини. Последните две полета са калории и тегло на храната. Информацията е използвана главно в страницата за консумация на храна.

Таблица „Exercises“ има един първичен ключ „ExerciseID“ и съдържа информация за упражненията, които потребителите могат да изпълняват. Други полета са името на упражнението, мускулите, върху които акцентира, трудност на упражнението и колко калории гори за минута. Информацията от таблицата е използвана главно в страницата за упражнения.



Фигура 9 – Клас диаграма на анализа

Основните класове в програмата са тези за различни страници, а именно – за регистрация, вход в профил, консумация на храна, тренировки, прогрес и разбира се основната страница. Всеки клас има конструктор и полета, които съдържат връзката с базата данни и са от тип „SqlConnection“.

Друго често срещано поле е „components“ от тип „System.ComponentModel.IContainer“. Това поле се среща в класовете за страниците за консумация на храна, прогрес и упражнения. Полето съдържа различни компоненти в страниците.

Класовете за регистрация и вход в профил съдържат методи за натискане на кутията за показване на парола, метод за връзка между двете страници и множество методи за натискане на различните бутони.

Класа за основната страница съдържа най – много методи и полета. Някои от тях са за текущ бутон, активна форма, временен индекс и др. Методите на класа са конструктор, избор на случаен цвят за тема на страницата и отваряне на различни форми при натискане на бутон.

Всички класове имат връзка с класа за основната форма, което е показно с показалците на Фигура 9.

Предоставям сорс код на основната страница:

namespace NutriApp

{

public partial class FormMainMenu : Form

{

//Fields

private Button currentButton;

private Random random;

private int tempIndex;

private Form activeForm;

//Constructor

public FormMainMenu()

{

InitializeComponent();

random = new Random();

}

//Methods

//Selects a random color for the button theme

private Color SelectThemeColor()

{

int index = random.Next(ThemeColor.ColorList.Count);

while (tempIndex == index)

{

index = random.Next(ThemeColor.ColorList.Count);

}

tempIndex = index;

string color = ThemeColor.ColorList[index];

return ColorTranslator.FromHtml(color);

}

//When a button is pressed the background becomes the random color

//And the text becomes white

private void ActivateButton(object btnSender)

{

if (btnSender != null)

{

if (currentButton != (Button)btnSender)

{

DisableButton();

//DisableButton();

Color color = SelectThemeColor();

currentButton = (Button)btnSender;

currentButton.BackColor = color;

currentButton.ForeColor = Color.White;

currentButton.Font = new System.Drawing.Font("Microsoft Sans Serif", 12.5F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(0)));

panelTitleBar.BackColor = color;

panelLogo.BackColor = ThemeColor.ChangeColorBrightness(color, -0.3);

//ThemeColor.PrimaryColor = color;

//ThemeColor.SecondaryColor = ThemeColor.ChangeColorBrightness(color, -0.3);

//btnCloseChildForm.Visible = true;

}

}

}

private void DisableButton()

{

foreach (Control previousBtn in panelMenu.Controls)

{

if (previousBtn.GetType() == typeof(Button))

{

previousBtn.BackColor = Color.FromArgb(51, 51, 76);

previousBtn.ForeColor = Color.Gainsboro;

previousBtn.Font = new System.Drawing.Font("Microsoft Sans Serif", 10F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(0)));

}

}

}

private void panelLogo\_Paint(object sender, PaintEventArgs e)

{

ActivateButton(sender);

}

private void OpenChildForm(Form childForm, object btnSender)

{

if (activeForm != null)

{

activeForm.Close();

}

ActivateButton(btnSender);

activeForm = childForm;

childForm.TopLevel = false;

childForm.FormBorderStyle = FormBorderStyle.None;

childForm.Dock = DockStyle.Fill;

this.panelDesktopPane.Controls.Add(childForm);

this.panelDesktopPane.Tag = childForm;

childForm.BringToFront();

childForm.Show();

lblTitle.Text = childForm.Text;

}

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

{

OpenChildForm(new Forms.Exercise(), sender);

//ActivateButton(sender);

}

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

{

ActivateButton(sender);

}

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

{

ActivateButton(sender);

}

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

{

ActivateButton(sender);

}

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

{

ActivateButton(sender);

}

}

}

Както и код за страницата за регистрация:

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Reflection.Metadata;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.Data.SqlClient;

namespace NutriApp.Forms

{

public partial class FormRegister : Form

{

public FormRegister()

{

InitializeComponent();

}

SqlConnection con = new SqlConnection("Data Source=.\\SQLEXPRESS;Initial Catalog=dbNutriApp;Integrated Security=True;MultipleActiveResultSets=True");

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

{

if (txtUsername.Text == "" && txtPassword.Text == "" && txtComPassword.Text == "")

{

MessageBox.Show("Username and Password fields are empty", "Registration Failed", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

else if (txtPassword.Text == txtComPassword.Text)

{

con.Open();

string query = "INSERT INTO Users VALUES ('" + txtUsername.Text + "','" + txtPassword.Text + "')";

SqlCommand cmd = new SqlCommand(query, con);

cmd.ExecuteNonQuery();

con.Close();

txtUsername.Text = "";

txtPassword.Text = "";

txtComPassword.Text = "";

MessageBox.Show("Your Account has been Successfully Created", "Registration Success", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

else

{

MessageBox.Show("Passwords do not match, Please Re-enter", "Registration failed", MessageBoxButtons.OK, MessageBoxIcon.Error);

txtPassword.Text = "";

txtComPassword.Text = "";

txtPassword.Focus();

}

}

private void checkBoxShowPas\_CheckedChanged(object sender, EventArgs e)

{

if (checkBoxShowPas.Checked)

{

txtPassword.PasswordChar = '\0';

txtComPassword.PasswordChar = '\0';

}

else

{

txtPassword.PasswordChar = '•';

txtComPassword.PasswordChar = '•';

}

}

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

{

txtUsername.Text = "";

txtPassword.Text = "";

txtComPassword.Text = "";

txtUsername.Focus();

}

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

{

new FormLogin().Show();

this.Hide();

}

}

}

## Модел на съдържанието / данните

В проекта се използват множество мултимедийни елементи, като снимки и лога, за да предоставят визуална репрезентация на различните страници и функционалности на приложението.

На страницата за тренировки се използва снимка на гира, за да подчертае фокуса на приложението върху тренировките.

На страницата за смяна на дата се използва снимка на календар, за да покаже на потребителите къде са и коя страница използват.

Основните функционалности на приложението имат визуално представяне в началната страница, като се използват икони и кратки описания за всяка функционалност.

Общо казано, в проекта се използват мултимедийни елементи за да се подчертае визуалното представяне на приложението и да се даде представа за функционалностите, които то предлага.

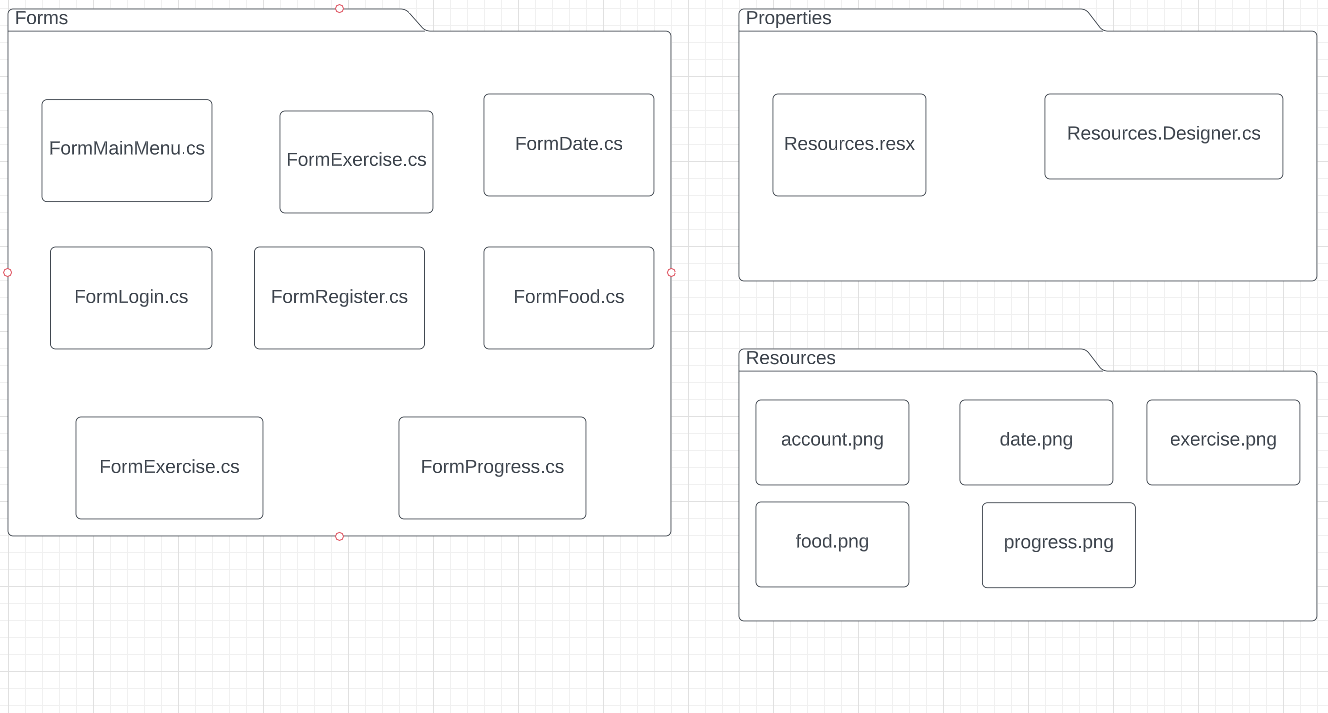
# Дизайн

За приложението ми избрах да използвам функционалността на Visual Studio за направа на Windows Forms настолно приложение с .NET. Смятам, че това е удобен и достъпен начин за изпълнението на дадените ми изисквания.

С помощта на Windows Forms, може да бъде създадено професионално изглеждащо приложение, като се използват готови компоненти и контроли, които имат вградена функционалност и могат да бъдат персонализирани спрямо нуждите на приложението. Допълнително приложението може да бъде компилирано в единичен екземпляр, което го прави по-бързо и по-лесно за инсталиране и използване.

Използването на .NET дава възможност за по-лесно управление на данните, тъй като той предоставя библиотеки за работа с бази данни и файлове, както и готови инструменти за взаимодействие с различни устройства, като принтери, скенери и други периферни устройства.

Освен това, .NET приложенията са по-безопасни, тъй като се изпълняват в специално създадена виртуална машина, която предпазва операционната система от вредни действия.



Фигура 10 – диаграма на архитектурата

„Forms“ частта съдържа всички форми в проекта, като всяка форма представлява отделен прозорец с определена функционалност.

„Rescources“ частта съдържа всички изображения, икони и други ресурси, необходими за визуализацията и работата на приложението.

„Properties“ частта съдържа файлове с настройки и конфигурации на проекта, като настройките за компилация и изпълнение на приложението. Тук можем да намерим файлове с разширение .resx, които задават различни параметри и настройки на проекта.

Тази архитектурна организация на проекта позволява лесното му разширяване и поддръжка в бъдеще. Като имаме разделение на формите, ресурсите и настройките на приложението, можем да се фокусираме върху дадената част от проекта, която желаем да модифицираме, без да се налага да променяме целия проект.

## Реализация на архитектурата на приложението

MVC (Model View Controller) е архитектурен шаблон, който се използва широко в софтуерното инженерство за разделяне на системата на трите основни компонента: модел (Model), изглед (View) и контролер (Controller). Това позволява по-лесно разбиране и поддръжка на кода, намаляване на зависимостите между компонентите и повишаване на преизползваемостта на кода.

При използване на MVC за настолни приложения с .NET, моделът представлява данните и бизнес логиката, изгледът е отговорен за визуалната част на приложението, а контролерът обработва входните данни и взаимодейства с модела и изгледа.

В сравнение с други архитектурни шаблони като Web Forms или Windows Forms, MVC има няколко предимства. Първо, MVC позволява по-голяма гъвкавост и свобода в програмния код, тъй като всеки компонент е разделен и може да бъде лесно заменен или модифициран. Второ, този шаблон предоставя по-голяма контрол върху заявките на потребителите, което прави работата с тях по-лесна и удобна за програмистите. Трето, използването на MVC може да повиши бързодействието на приложението, тъй като визуализацията на данните се извършва само когато е необходимо.

Затова, за изграждане на настолно приложение с .NET, използването на MVC може да бъде много полезно и удобно за програмистите, тъй като позволява по-голяма гъвкавост, контрол и ефективност.

## Описание на слоевете, предназначението им, библиотеки и методи включени в съответния слой.

### Слоеве

MVC архитектурата се състои от три основни слоя - Model, View и Controller. В настолно приложение с .NET тези слоеве са представени по следния начин:

Модел (Model) - този слой отговаря за представянето на данните и бизнес логиката на приложението. Тук се намират класове, които представят обекти и функционалности, които се използват от приложението. В този слой се намира връзката с базата данни и други източници на данни, които се използват от приложението.

Изглед (View) - този слой отговаря за визуалното представяне на данните. Тук се намират формите и контролите, които потребителят вижда и с които взаимодейства. В този слой може да се използват технологии за изграждане на визуални елементи.

Контролер (Controller) - този слой отговаря за управлението на логиката на приложението. Тук се намират класове, които обработват заявките на потребителя и комуникират с модела и изгледа. В този слой са създадени функционалности за обработка на заявки и извличане на информация от модела.

Тези три слоя работят заедно, за да осигурят едно добре структурирано и организирано приложение, което е лесно за поддръжка и разширяване. Те са създадени така, че да разделят логиката на приложението на различни компоненти, които са отговорни за различни функционалности.

### Библиотеки и методи

За достъп до базата данни използвам библиотеката ADO.NET.

ADO.NET е библиотека за достъп до бази данни, която е част от .NET Framework. Тя предоставя удобен начин за свързване с различни видове бази данни, включително Microsoft SQL Server, Oracle, MySQL и други.

За да се използва ADO.NET, трябва да се свържем към базата данни и да използваме SQL заявки за да извършваме операции като добавяне, промяна и изтриване на данни. Библиотеката предоставя класове като SqlConnection, SqlCommand и SqlDataReader за тази цел.

Във всяко Windows Forms приложение, базата данни може да бъде разделена в отделен проект за да може да се използва на различни места в приложението. В този проект ще бъдат създадени класове, които да управляват връзката с базата данни и да извличат необходимата информация.

Един от методите за връзка с базата данни е представен в долната точка – Организация и код. Методът btnRegister\_Click(object sender, EventArgs e) добавя потребител в таблицата Users от базата. За тази цел се използват стойностите от полетата за въвеждане на име и парола от потребителя.

## Организация и код

Програмен код на метод за регистрация на потребител:

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

{

if (txtUsername.Text == "" && txtPassword.Text == "" && txtComPassword.Text == "")

{

MessageBox.Show("Username and Password fields are empty", "Registration Failed", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

else if (txtPassword.Text == txtComPassword.Text)

{

con.Open();

string query = "INSERT INTO Users VALUES ('" + txtUsername.Text + "','" + txtPassword.Text + "')";

SqlCommand cmd = new SqlCommand(query, con);

cmd.ExecuteNonQuery();

con.Close();

txtUsername.Text = "";

txtPassword.Text = "";

txtComPassword.Text = "";

MessageBox.Show("Your Account has been Successfully Created", "Registration Success", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

else

{

MessageBox.Show("Passwords do not match, Please Re-enter", "Registration failed", MessageBoxButtons.OK, MessageBoxIcon.Error);

txtPassword.Text = "";

txtComPassword.Text = "";

txtPassword.Focus();

}

}

Можем да забележим, че при опит за въвеждане на вече съществуващо потребителско име, програмата ще изведе прозорец за грешка и ще върне потребителят в менюто за избор на потребителско име.

Подобна проверка има и в метода, използван в страницата за вход на потребител. Ако потребител въведе несъответваща парола и потребителско име ще последва отваряне на прозорец за грешка и връщане на потребителя в полето за въвеждане на данни.

Програмен код на метод за вход на потребител:

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

{

con.Open();

string query = "SELECT \* FROM Users WHERE username = '" + txtUsername.Text + "' and password='" + txtPassword.Text + "'";

SqlCommand cmd = new SqlCommand(query, con);

SqlDataReader dr = cmd.ExecuteReader();

if (dr.Read())

{

new FormMainMenu().Show();

this.Hide();

}

else

{

MessageBox.Show("Invalid Username or Password, Please Try Again", "Login Failed", MessageBoxButtons.OK, MessageBoxIcon.Error);

txtUsername.Text = "";

txtPassword.Text = "";

txtUsername.Focus();

}

con.Close();

}

## 4.4Наличие на потребителски интерфейс(конзолен, графичен, уеб)

Потребителския интерфейс съдържа страници за регистрация и вход на съществуващ потребител. След като потребителят влезе в профила си и отвори главната страница, той може да избере опцията за тренировка, която го пренасочва към страница, където може да избере тренировъчен план за определен период от време. Тук потребителят може да избере една от наличните опции за тренировки и да я добави към своя график за тренировки. Също така, на страницата за тренировки потребителят може да добавя нови тренировки или да редактира вече съществуващите.

На страницата за консумация на храна, потребителят може да въведе информация за приетите от него храни за определен ден. Той може да избере продукт от базата данни на приложението или да въведе нов продукт, който да бъде добавен към базата данни. След като потребителят добави информацията за приетите от него храни, той може да види дневния си прием на калории и макронутриенти.

На страницата за преглед на прогреса, потребителят може да види статистика за извършените от него тренировки и консумираната от него храна. Тук потребителят може да види своите постижения и да сравни прогреса си между различните периоди от време.

На страницата за управление на акаунта потребителят може да промени информацията си, като например името, паролата и др.

На страницата за промяна на дата потребителят може да избере нова дата за тренировки и да я запази в системата. След това, ако потребителят отиде на страницата за тренировки или за прогрес, той ще вижда информация за тренировки и прогрес за избраната от него дата.

Страницата за промяна на дата може да бъде достъпна от главното меню на приложението или от страницата за тренировки и прогрес. Потребителят може да избере дата, като използва календара или въведе датата ръчно. След като избере датата, потребителят трябва да натисне бутона за запазване, за да потвърди избора си.

# Ефективност и бързодействие на решението

Ефективността и бързодействието на решението са много важни, особено в случаите, когато има множество потребители, които използват приложението едновременно. В случай на забавяне или некачествено функциониране на приложението, потребителите могат да го напуснат и да потърсят друго решение.

Освен това, при бавно функциониращо приложение, потребителите могат да загубят доверие в него и да не го използват повече. Ефективността и бързодействието могат да бъдат определени от множество фактори, като например дизайна на базата данни, ефективността на програмния код, използваните алгоритми и библиотеки и т.н.

Затова е много важно да се обърне внимание на ефективността и бързодействието на приложението от самото начало на разработката. Това може да бъде постигнато чрез използване на оптимални решения и инструменти, тестване на приложението и оптимизация на кода.

Нека тестваме бързодействие на метода btnLogin\_Click(object sender, EventArgs e)

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

{

con.Open();

string query = "SELECT \* FROM Users WHERE username = '" + txtUsername.Text + "' and password='" + txtPassword.Text + "'";

SqlCommand cmd = new SqlCommand(query, con);

SqlDataReader dr = cmd.ExecuteReader();

if (dr.Read())

{

new FormMainMenu().Show();

this.Hide();

}

else

{

MessageBox.Show("Invalid Username or Password, Please Try Again", "Login Failed", MessageBoxButtons.OK, MessageBoxIcon.Error);

txtUsername.Text = "";

txtPassword.Text = "";

txtUsername.Focus();

}

con.Close();

Можем да забележим, че SQL заявката, използвана в метода, е прост SELECT оператор с две условия в WHERE клаузата. Той извлича всички колони (\*) от таблицата Users, където потребителското име и паролата съответстват на въведените стойности в текстовите полета „txtUsername“ и „txtPassword“. Заявката не използва индекси или оптимизира търсенето, така че времето за изпълнение нараства линейно с размера на таблицата Users.

Ако таблицата „Users“ съдържа n реда, най-лошият случай е, когато комбинацията от потребителско име и парола не съществува в таблицата, и методът трябва да прочете всичките n реда, за да установи това. В този случай времето за изпълнение на метода ще бъде пропорционално на размера на таблицата, и можем да го оценим като T(n) = O(n), където T(n) е времето за изпълнение на метода за таблица от размер n, а O(n) е голямото O бележение за линейна сложност на времето.

# Тестване

За тестване на приложението използвам xUnit, защото xUnit е библиотека за тестване на код в .NET приложения, която осигурява лесен и ефективен начин за писане на автоматизирани тестове. Използвам я за тестване на различните функционалности на моето настолно приложение, за да гарантирам, че всичко работи коректно и без грешки.

xUnit има много предимства, като например лесна интеграция с различни инструменти за тестване и CI/CD системи, автоматично разпознаване на тестовите методи, което ги прави лесни за използване и много други.

Тестването е изключително важно за да се гарантира, че приложението работи коректно и без грешки. С помощта на xUnit мога да изпълня множество тестове, които да покрият всички възможни сценарии за използване на приложението и да осигурят, че то функционира без проблеми във всички случаи.

Програмен код за тест на 4 метода чрез xUnit:

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Reflection.Metadata;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.Data.SqlClient;

using Xunit;

namespace NutriApp.Forms

{

public partial class FormRegister : Form

{

public FormRegister()

{

InitializeComponent();

}

SqlConnection con = new SqlConnection("Data Source=.\\SQLEXPRESS;Initial Catalog=dbNutriApp;Integrated Security=True;MultipleActiveResultSets=True");

[Fact]

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

{

if (txtUsername.Text == "" && txtPassword.Text == "" && txtComPassword.Text == "")

{

MessageBox.Show("Username and Password fields are empty", "Registration Failed", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

else if (txtPassword.Text == txtComPassword.Text)

{

con.Open();

string query = "INSERT INTO Users VALUES ('" + txtUsername.Text + "','" + txtPassword.Text + "')";

SqlCommand cmd = new SqlCommand(query, con);

cmd.ExecuteNonQuery();

con.Close();

txtUsername.Text = "";

txtPassword.Text = "";

txtComPassword.Text = "";

MessageBox.Show("Your Account has been Successfully Created", "Registration Success", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

else

{

MessageBox.Show("Passwords do not match, Please Re-enter", "Registration failed", MessageBoxButtons.OK, MessageBoxIcon.Error);

txtPassword.Text = "";

txtComPassword.Text = "";

txtPassword.Focus();

}

}

[Fact]

private void checkBoxShowPas\_CheckedChanged(object sender, EventArgs e)

{

if (checkBoxShowPas.Checked)

{

txtPassword.PasswordChar = '\0';

txtComPassword.PasswordChar = '\0';

}

else

{

txtPassword.PasswordChar = '•';

txtComPassword.PasswordChar = '•';

}

}

[Fact]

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

{

txtUsername.Text = "";

txtPassword.Text = "";

txtComPassword.Text = "";

txtUsername.Focus();

}

[Fact]

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

{

new FormLogin().Show();

this.Hide();

}

}

}

# Заключение и възможно бъдещо развитие

В заключение, проектът представлява настолно приложение за управление на фитнес програма и диета. Беше използвана архитектурата MVC, която разделя приложението на три основни слоя - Model, View и Controller, за да се постигне по-добра организация и лесна поддръжка на кода. Използвана е библиотеката ADO.NET за достъп до базата данни, която съхранява информацията за потребителите, тяхната регистрация, информация за тренировките и храненето им.

Потребителският интерфейс е разработен във Visual Studio, като са използвани Windows Forms, което прави интерфейса достъпен и удобен за потребителите. Тестването на приложението се извършва с помощта на xUnit, който позволява изцяло автоматизирано тестване на функционалностите на приложението.

Бъдещото развитие на проекта включва добавяне на допълнителни функционалности като календар за планиране на тренировки и дневник за записване на резултатите от тренировките и диетата. Също така, могат да бъдат добавени нови възможности за хранене и тренировки, за да се предложи по-богато разнообразие на потребителите. Тези допълнения ще увеличат функционалността и полезността на приложението за потребителите му.

# Използвани литературни източници и Уеб сайтове

1. Отдел от Microsoft документация на .NET описващ тестване с xUnit.  
   (https://learn.microsoft.com/en-us/dotnet/core/testing/unit-testing-with-dotnet-test)
2. Форум за въпроси свързани с програмиране.  
   (https://stackoverflow.com/)
3. Урок за вписване на информация в база данни чрез ADO.NET.  
   (https://www.youtube.com/watch?v=ZpGz67BOAFQ)
4. Речник с терминология английски – български.  
   (https://iate.europa.eu/home)
5. Инструмент за направа на таблици, диаграми, графики

(https://app.diagrams.net/)

# Приложения

Table

Description automatically generated

Фигура 9 – Клас диаграма на анализа