ДОКУМЕНТАЦИЯ

**ТЕМА**: Проект „Фитнес инструктор“

***Изготвен от****: Славян Христов и Радослав Иванов,*

*учащи в „Технически университет“ – Варна*

Информация за студентите:

*Студент 1: Студент 2:*

*име: Славян Христов име: Радослав Иванов*

*фак. номер: 18621793 фак. номер: 18621797*

*факултет: ФИТА факултет: ФИТА*

*специалност: СИТ специалност: СИТ*

*група: 5б група: 5б*

Съдържание*:*

1. *Заглавна страница*
2. *Техническо задание на проекта*
   1. *Функционални изисквания*
   2. *Нефункционални изисквания*
3. *Определяне на роли*
4. *Структура и проектиране на системата*
5. *Детайлно обяснение на компонентите и тяхната функционалност*
6. *Начин на работа на програмата*
7. *Възможни бъдещи подобрения*

Техническо задание на проекта

Тема на проекта „Фитнес инструктор“. Системата съдържа главен актьор, “инструктор”, който има възможността да добавя свои клиенти, които след това може да им редактира данните, да им изготвя фитнес програми, да им изчислява калориен прием за ден и също да калкулира индекс на телесната маса.

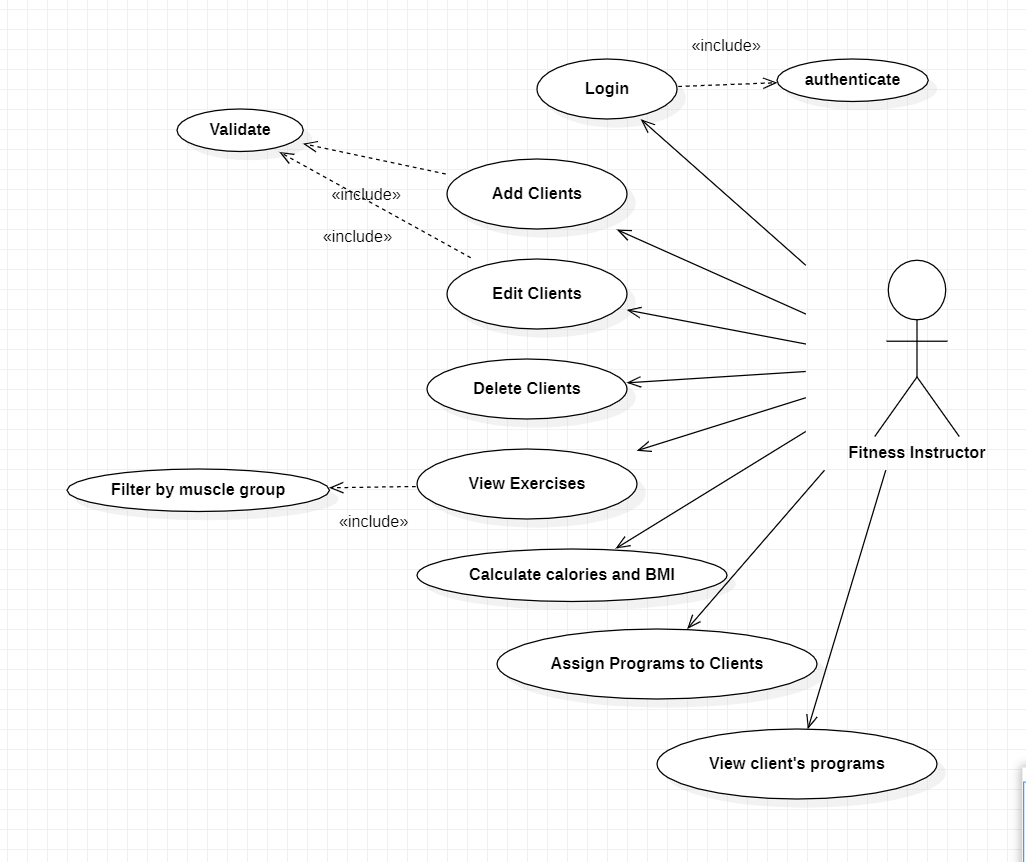
1. Функционални изисквания

Функционалността на системата трябва да бъде съобразена с нуждите на потребителите, които ще я използват. Интерфейсът на системата трябва да бъде прост, приятен и да позволява лесно и безпрепятствено използване, без грешки в процеса на работа.

Лицата, които ще използват системата, са фитнес инструкторите. Те трябва да могат да изпълняват следните действия:

1. Фитнес инструктор
   1. Влизане в системата.
   2. Добавяне на клиенти.
   3. Редактиране на клиенти.
   4. Изтриване на клиенти.
   5. Преглед на вкараните упражнения в системата.
   6. Калкулация на калориен прием за ден и индекс на телесната маса(BMI) за избран клиент.
   7. Възлагане на тренировъчна програма за избран клиент.
   8. Преглед на тренировъчна програма за избран клиент.

Системата трябва да предостави възможност за изпълнение на всички действия описани горе. Допълнителни функционалности трябва да бъдат имплементирани за валидиране на данните.



1. Нефункционални изисквания

Технологични средства за реализацията:

1. SQL Server – база данни
2. Windows Forms - Реализация на потребителски и графичен интерфейс
3. C# - използван език

Необходими ресурси: работна ръка, техника(компютри, интернет), електричество

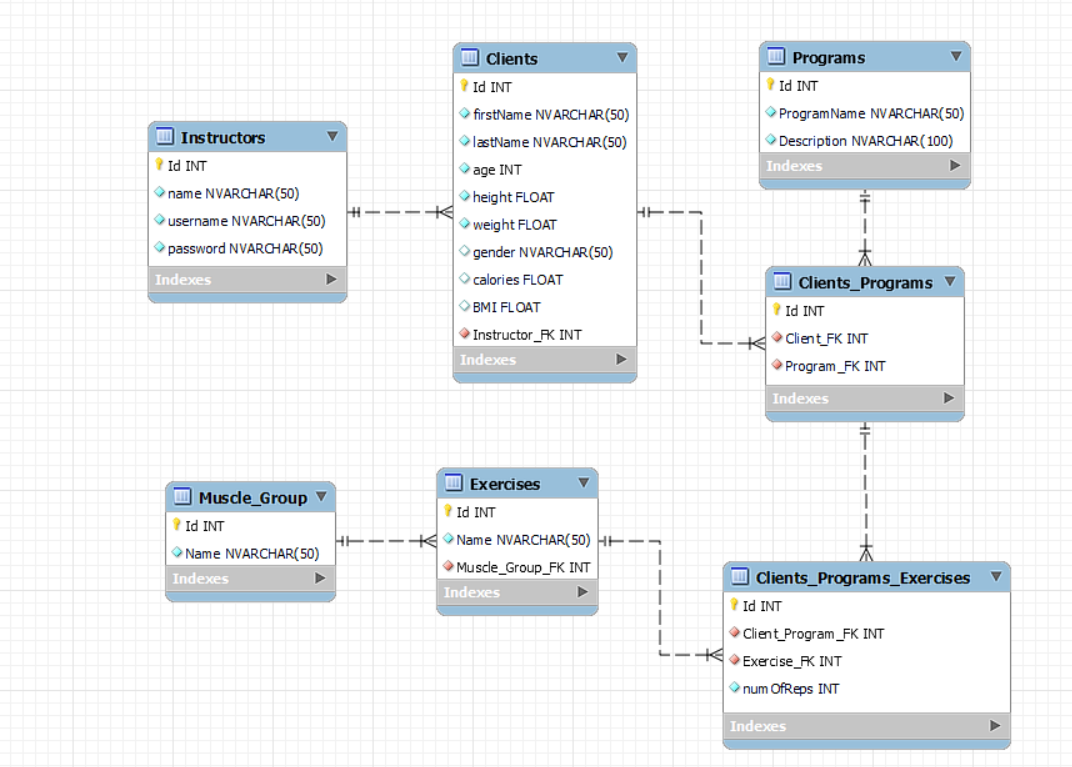
1. Определяне на роли

А) Славян Христов – разработчик, тестер, мениджър

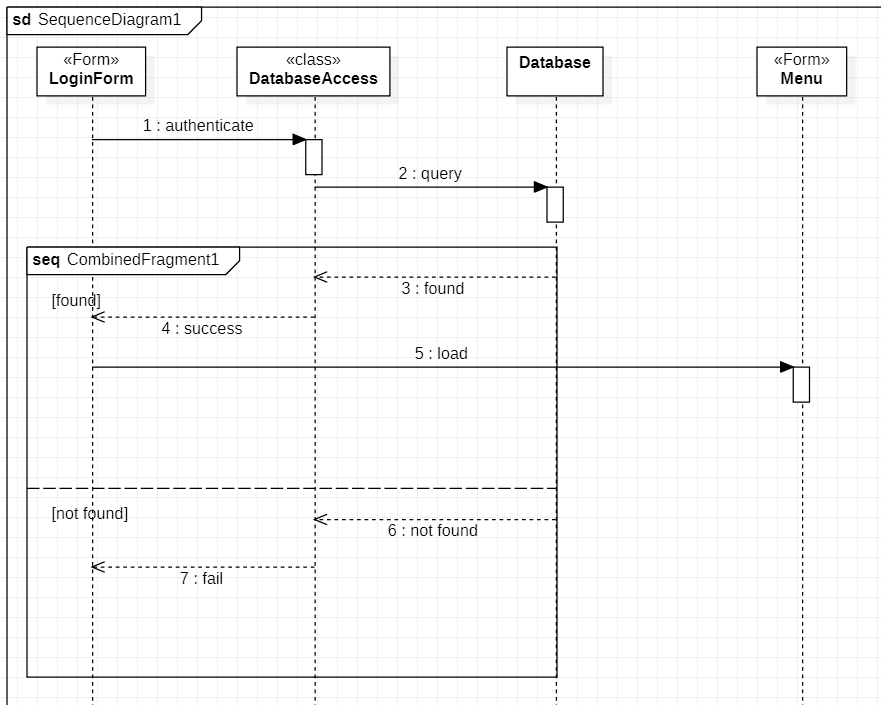
Б) Радослав Иванов – разработчик, тестер, мениджър

1. Структура и проектиране на системата

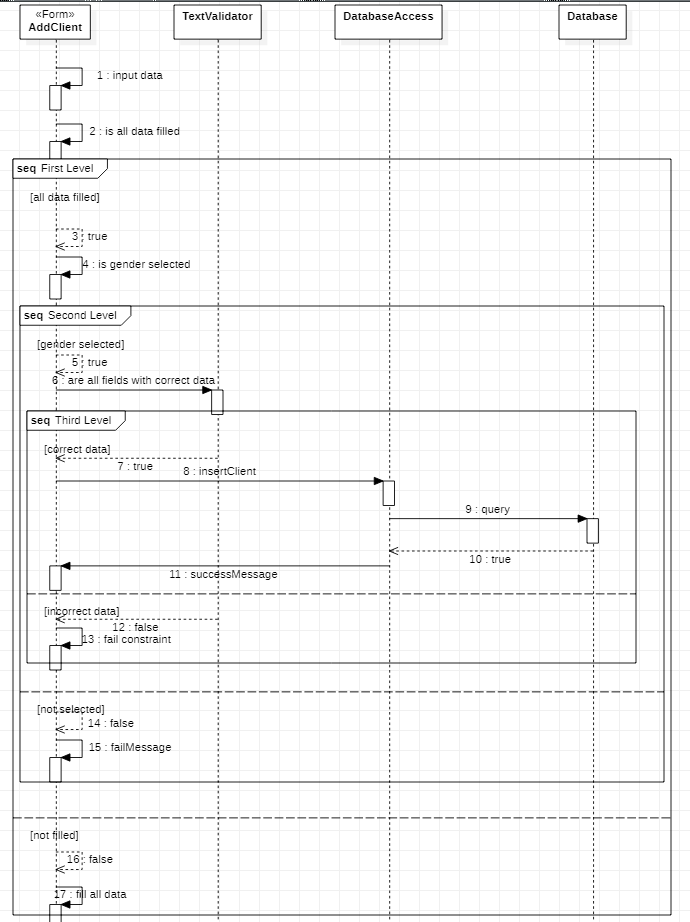
EER Diagram



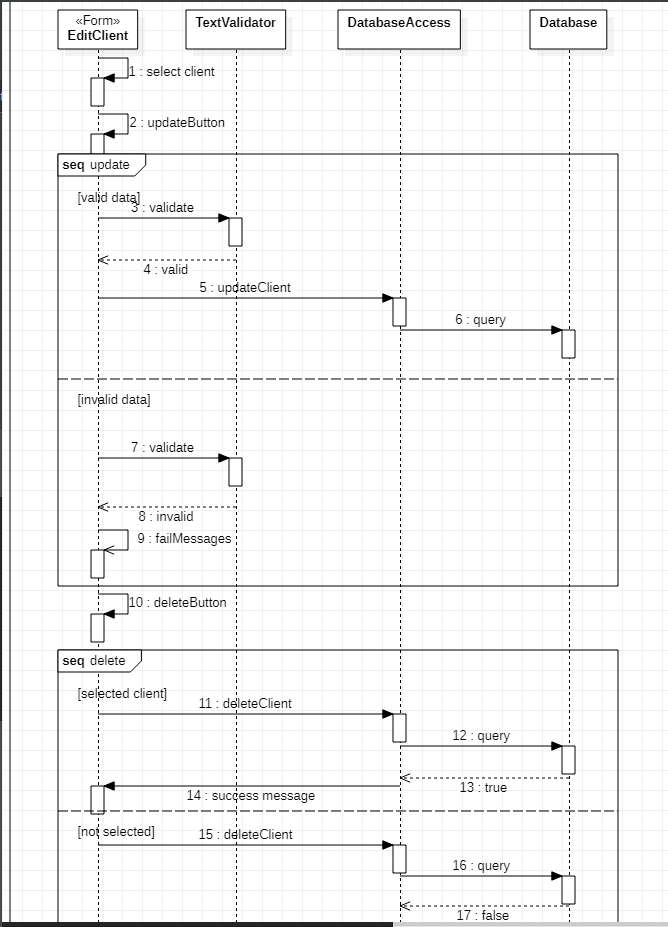
Sequence Diagram for Login



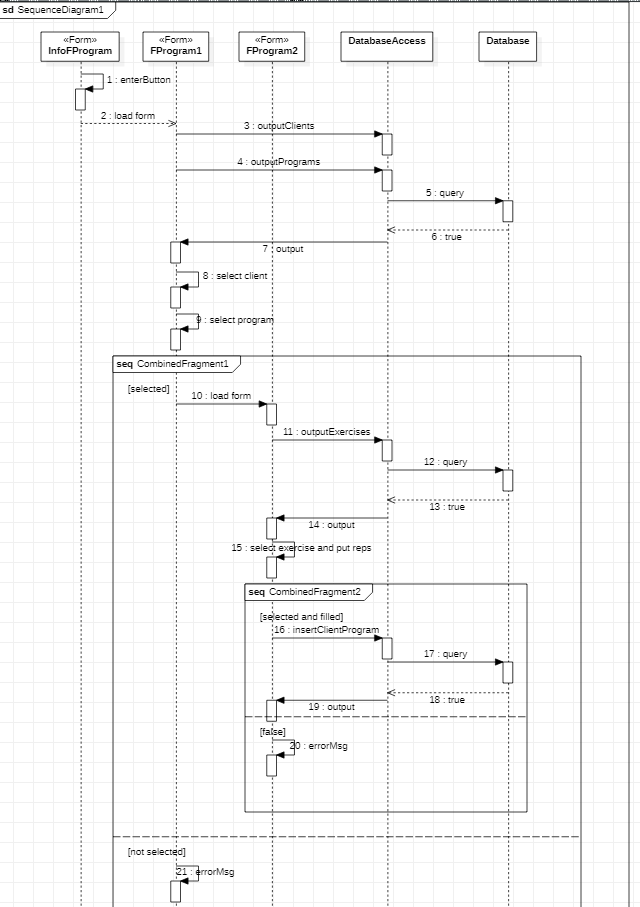
Sequence Diagram for Adding Clients



Sequence Diagram for Editing Clients



Sequence Diagram for assigning fitness programs



1. Детайлно обяснение на компонентите и тяхната функционалност
2. **DatabaseAccess** – Този клас може да се определи като среден слой между базата от данни и потребителския интерфейс на апликацията. Тук се осъществява връзката с база данни. Самата връзка се получава чрез деклариране на конекцията към SQL като се предаде пътят към базата данни.Класът съдържа функциите, които ще комуникират(изпълняват заявки, команди) с база данни. В DatabaseAccess също е декларирана член променливата SqlCommand command, която чрез нея ще се изпълняват всички заявки(queries).

Функция, която вмъква рекорд в таблица от база данни:

public void insertClient(Client client, int fk)

{

String query = "insert into Clients (firstName, lastName, age, height, weight, gender, Instructor\_FK) values(N'" + client.FirstName + "',N'" + client.LastName + "','"

+ client.Age.ToString() + "','" + client.Height.ToString() + "','" + client.Weight.ToString() + "','" + client.Gender + "','" + fk + "')";

command = new SqlCommand(query, connection);

try

{

command.Connection.Open(); // отваряне на конекцията

command.ExecuteNonQuery(); // изпълнение на заявката

command.Connection.Close(); // затваряне на конекцията

MessageBox.Show("Успешно добавен клиент!"); // уведомяване за успех

}

catch (Exception ex) // прихващане на exception

{

MessageBox.Show("Неуспех"); // уведомяване за грешка при insert

}

}

Функция, която връща резултат от SELECT:

public object outputClients(int fk)

{

String query = "SELECT \* FROM Clients WHERE Instructor\_FK = '" + fk + "'";

command = new SqlCommand(query, connection);

SqlDataAdapter dataAdapter = new SqlDataAdapter(command); // позволява комуникация между базата данни и някакъв DataSet

DataTable dt = new DataTable(); // колекция от редове и колони, която складира данните.

dataAdapter.Fill(dt); // запълване на DataТable

return dt; // върнатия резултат ще се присвои от DataSource на dataGridView

}

1. **IntroForm –** Това е първата форма, която ще се отвори в това приложение. Тя няма някаква важна роля. Съдържа Progress Bar, който се инкрементира с отминаването на определен интервал от време. При достигането на максималната стойност на прогрес бара, Intro Form се затваря и се отваря следващата форма: „LoginForm“.
2. **LoginForm –** Тази форма осъществява логването на фитнес инструктор в системата. Успешното логване се реализира при правилно въвеждане на поисканата от потребителя информация. При натискането на бутона Login се изпълняват самите валидации. Това става чрез заявка към база данни, която проверява дали съществува въведения инструктор. Ако той съществува Login Form се затваря и се отваря следващата. Ако не, потребителят ще бъде информиран чрез MessageBox, че данните не са правилни.

DataTable dataTable = databaseAccess.authenticateInstructor(usernameBox.Text.Trim(), passwordBox.Text.Trim());

if(dataTable.Rows.Count == 1)

{

dataRetriever = DataRetriever.Instance;

dataRetriever.setUsername(usernameBox.Text);

Menu menu = new Menu();

this.Hide();

menu.ShowDialog();

this.Close();

}

else

{

MessageBox.Show("Incorrect data!");

}

1. **DataRetriever –** Това е Singleton клас, който притежава само една инстанция на себе си и дава право на достъп до тази инстанция. Основната цел на този клас е да прехвърля данни от една форма на друга.
2. **Menu –** След правилно въведените данни в Login формата, се отваря Menu. От тук инструкторът има достъп до функционалностите на приложението, а те са следните:

* Добавяне на клиент.
* Редактиране на клиент.
* Преглед на всички упражнения
* Калкулаторът за изчисляване на калориите и BMI
* Преглед на тренировъчните програми на клиент/и
* Възлагане на тренировъчни програми на избран клиент.

Тази форма е оформена, така че при отварянето на форма за избраната функционалност, Menu формата да е родителска. Тоест ново отворената форма да е вградена вътре в Menu формата и да няма фрейм(рамка) :

private void openChildForm(Form childForm)

{

if(currentChildForm != null)

{

//open single form

currentChildForm.Close();

}

currentChildForm = childForm;

childForm.TopLevel = false;

childForm.FormBorderStyle = FormBorderStyle.None;

childForm.Dock = DockStyle.Fill;

panelDesktop.Controls.Add(childForm);

panelDesktop.Tag = childForm;

childForm.BringToFront();

childForm.Show();

lblTitleChildForm.Text = childForm.Text;

}

А при натискане на бутон от менюто за отваряне на избрана форма, селектирания бутон променя дизайна си, когато формата на която съответства е отворена:

private void activateButton(object senderBtn, Color color)

{

if (senderBtn != null)

{

disableButton(); // премахване на дизайна за избран бутон, от предишно избрания бутон

//Button

currentBtn = (IconButton)senderBtn;

currentBtn.BackColor = Color.FromArgb(37, 36, 81); // промяна на цвета на бутона

currentBtn.ForeColor = color;

currentBtn.TextAlign = ContentAlignment.MiddleCenter; // подравянване на текста

currentBtn.IconColor = color; // промяна на цвета на иконата на бутона

currentBtn.TextImageRelation = TextImageRelation.TextBeforeImage; // промяна на позицията на бутона

currentBtn.ImageAlign = ContentAlignment.MiddleRight;

// Icon Current Child Form

iconCurrentChildForm.IconChar = currentBtn.IconChar;

iconCurrentChildForm.IconColor = color;

}

}

1. **AddClientForm –** При натискането на бутона „Add Client” от менюто се отваря AddClient Form. Тук логналия се фитнес инструктор може да добавя свои клиенти към системата. Това става чрез попълване на поисканите данни, които са:

* Имена
* Години
* Височина (в см.)
* Килограми
* Пол

В формата всяко поле за въвеждане на данни има errorProvider, който чрез класа TextValidator проверява дали данните съответстват на дефинирания regex pattern за съответния textbox.

Ако инструкторът не е въвел никаква информация във всички textbox-ове и натисне бутона за добавяне на клиент ще се изведе MessageBox и добавянето няма да се осъществи. Ако пък е въвел правилни данни, но не е избрал пол, пак добавянето на клиент няма да се реализира и също ще се изведе съобщение за грешка. Вече, когато е въвел правилната информация и е избрал пол, тогава ще се добави въведения клиент.

Пример за проверка на textbox:

private void base\_Validating(TextBox textBox, string errorMsg, String pattern, CancelEventArgs e)

{

validator = new TextValidator(pattern, textBox.Text);

bool state = validator.regexValidator();

if (state)

{

e.Cancel = false;

errorProvider1.SetError(textBox, "");

}

else

{

e.Cancel = true;

errorProvider1.SetError(textBox, errorMsg);

}

}

private void firstNameBox\_Validating(object sender, CancelEventArgs e)

{

base\_Validating(firstNameBox, "Enter valid first name!", @"(^[А-ЯA-Z][а-яa-z]{2,}$)", e);

}

При натискането на бутона за добавяне на клиент:

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

{

try

{

Client client = new Client();

client.FirstName = firstNameBox.Text;

client.LastName = lastNameBox.Text;

client.Age = int.Parse(ageBox.Text);

client.Height = float.Parse(heightBox.Text);

client.Weight = float.Parse(weightBox.Text);

client.Gender = gender;

if (client.Gender == null)

MessageBox.Show("Please choose a gender!");

else

{

if (Equals(dataRetriever.getUsername(), "slavcho44"))

db.insertClient(client, 1);

else

db.insertClient(client, 2);

}

}

catch (Exception)

{

MessageBox.Show("Error");

}

}

*Regex Pattern за first name и last name:* (^[А-ЯA-Z][а-яa-z]{2,}$)

*Regex Pattern за age, height, weight:* ^\d{0,3}$

1. **TextValidator –** Класът, който отговаря дали въведените данни отговарят на regular expression pattern-а. Член променливи: String data – данни, String strRegex – regular expression. Самата проверка се получава чрез функцията:

public bool regexValidator()

{

Regex re = new Regex(strRegex);

if (re.IsMatch(data))

return (true);

else

return (false);

}

Функцията връща истина ако данните съответстват с pattern-a, лъжа ако не.

1. **EditClientForm -** След като са добавени клиенти в системата, инструкторът може да редактира данните им или да изтрие избрания клиент от системата. А тези актуализации се случват в Edit Client формата. За да има инструкторът право да редактира клиент, първо трябва да го избере от dataGridView. Когато бъде избран, textbox-овете се запълват със съответните данни за клиент.

Споменатите textbox-ове също имат errorProvider, както Add Client формата, който валидира въведените данни.

При избран клиент:

private void dataGridView\_CellDoubleClick(object sender, DataGridViewCellEventArgs e)

{

clientId = Convert.ToInt32(dataGridView.Rows[e.RowIndex].Cells[0].Value.ToString());

firstNameBox.Text = dataGridView.Rows[e.RowIndex].Cells[1].Value.ToString();

lastNameBox.Text = dataGridView.Rows[e.RowIndex].Cells[2].Value.ToString();

ageBox.Text = dataGridView.Rows[e.RowIndex].Cells[3].Value.ToString();

heightBox.Text = dataGridView.Rows[e.RowIndex].Cells[4].Value.ToString();

weightBox.Text = dataGridView.Rows[e.RowIndex].Cells[5].Value.ToString();

}

При натискане на update бутон:

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

{

if(String.IsNullOrEmpty(firstNameBox.Text) || String.IsNullOrEmpty(lastNameBox.Text) || String.IsNullOrEmpty(ageBox.Text) || String.IsNullOrEmpty(heightBox.Text) || String.IsNullOrEmpty(weightBox.Text))

{

MessageBox.Show("Empty fields", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

else

{

databaseAccess.updateClient(clientId, firstNameBox.Text, lastNameBox.Text, ageBox.Text, heightBox.Text, weightBox.Text);

dataGridView.DataSource = GetDataGridView();

}

}

При натискане на delete бутон:

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

{

databaseAccess.deleteClient(clientId);

dataGridView.DataSource = GetDataGridView();

}

1. **ExerciseForm –** В тази форма, инструктор може да види всички упражнения вкарани в системата. Те се филтрират по избрана мускулна група. Филтрирането става чрез натискане на pictureBox, който изобразява на снимка даден мускул. Когато се натисне pictureBox-a, dataGridView-a на формата се запълва с всички упражнения от избраната мускулна група.

1. **CaloriesCalculatorForm –** Когато в база данни има добавени клиенти, след това инструкторът може да изчислява техния калориен прием за ден и коефициентът на телесна маса. В формата CaloriesCalculatorForm чрез dataGridView се отпечатват всички рекорди на клиенти. При селектиран клиент от dataGridView-a, фитнес инструкторът първо трябва да избере от комбо бокс, нивото на активност през деня на клиент. Това означава дали има работа, която изисква някакъв физически труд, дали спортува и колко често. След като се избере нивото на активност, той трябва да избере целта на клиента. Тя може да бъде една от следните три: Сваляне на килограми, качване или поддържане. Когато е завършена и тази стъпка, вече инструкторът може да направи актуализация на клиента и съответните стойности получени от калкулациите ще се въведат в рекорда на клиента.

Изчисление на коефициент на телесна маса: *BMI = тегло / височина^2*

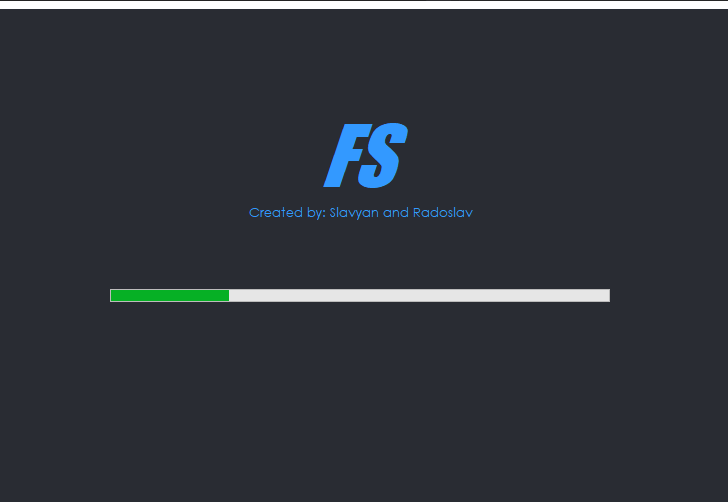
Изчисление на калориен прием за ден: за мъж: *Calories = [(10 \* тегло + 6.25 \* височина – 5 \* възраст + 5) \* ниво на активност] + или – 500(ако сме избрали да качваме или сваляме)*

*за жена: Calories = [(10 \* тегло + 6.25 \* височина – 5 \* възраст - 161) \* ниво на активност] + или – 500(ако сме избрали да качваме или сваляме)*

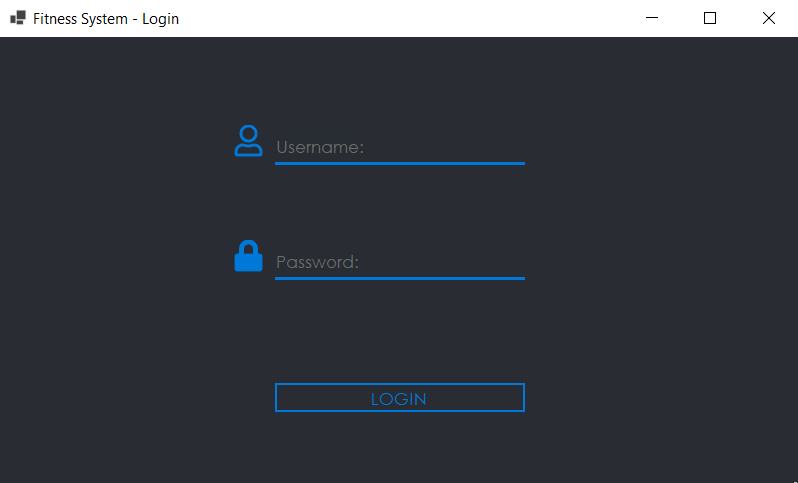
1. **FProgramForm1 –** Друга способност, която притежава фитнес инструкторът в тази система е да съставя тренировъчни фитнес програми за клиенти. В първата форма FProgramForm1 от тази функционалност се избира клиента, на когото ще бъде правена тази програма и също се избира името на програмата. След като са избрани, потребителят натиска бутона Next и сегашната форма се затваря и се продължава към следващата, която е FProgramForm2.
2. **FProgramForm2 –** След като е избран клиент и име на програма се прави insert в Clients\_Programs таблицата в база данни, която е създадена поради причината че таблиците Clients и Programs имат връзка много към много. След този insert инструкторът трябва да избере и упражненията, които ще съдържа тази фитнес програма. При всяко избиране на упражнение, той трябва също да въведе броя на повторения за упражнение. Когато извърши тези действия, той вече може да insert-не в таблица Clients\_Programs\_Exercises, тя също е създадена поради връзка много към много(Clients\_Programs към Exercises). Когато и това е извършено се отпечатва резултатът от тези всички действия, тоест принтира се новосъздадената програма с нейните

упражнения, клиентът, който ще я използва и инструкторът, който я създал.

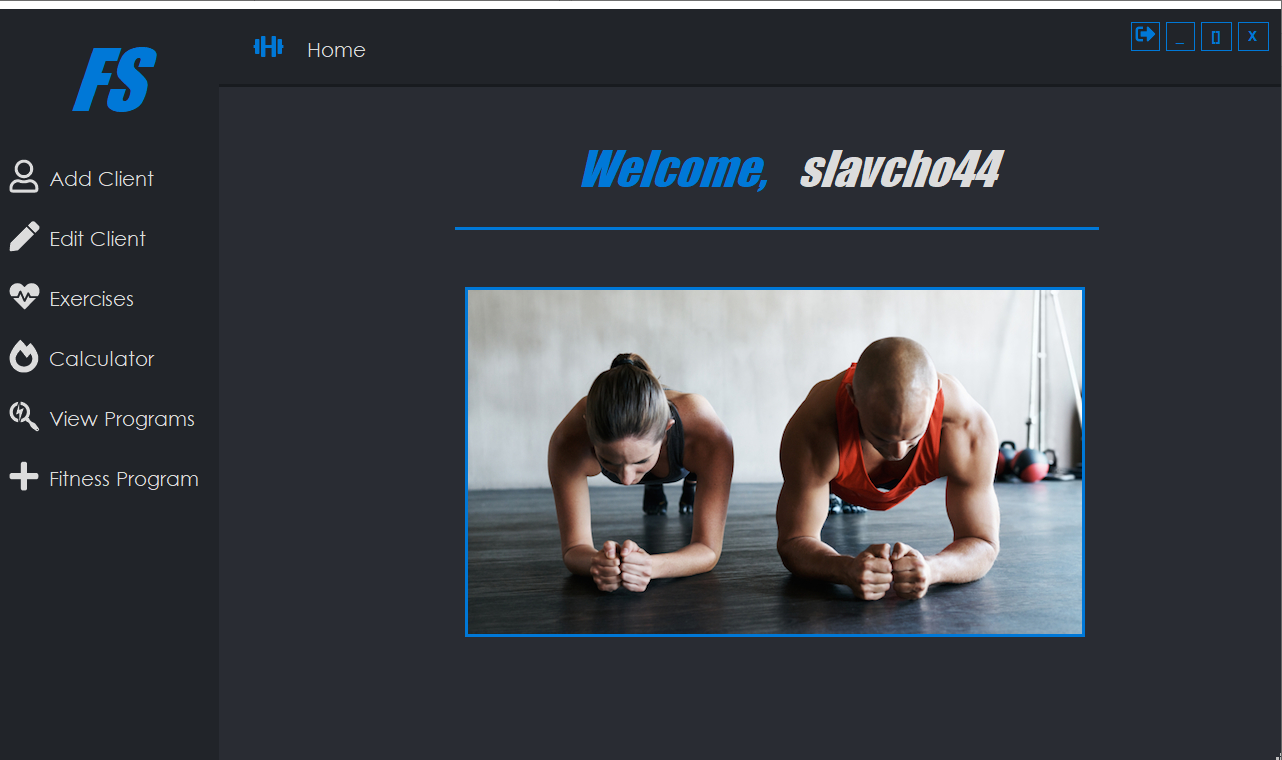
1. **Reports –** В тази форма инструкторът може да преглежда всички направени фитнес програми до момента.
2. Начин на работа на програмата.
3. Стартиране на апликацията. Първа форма.



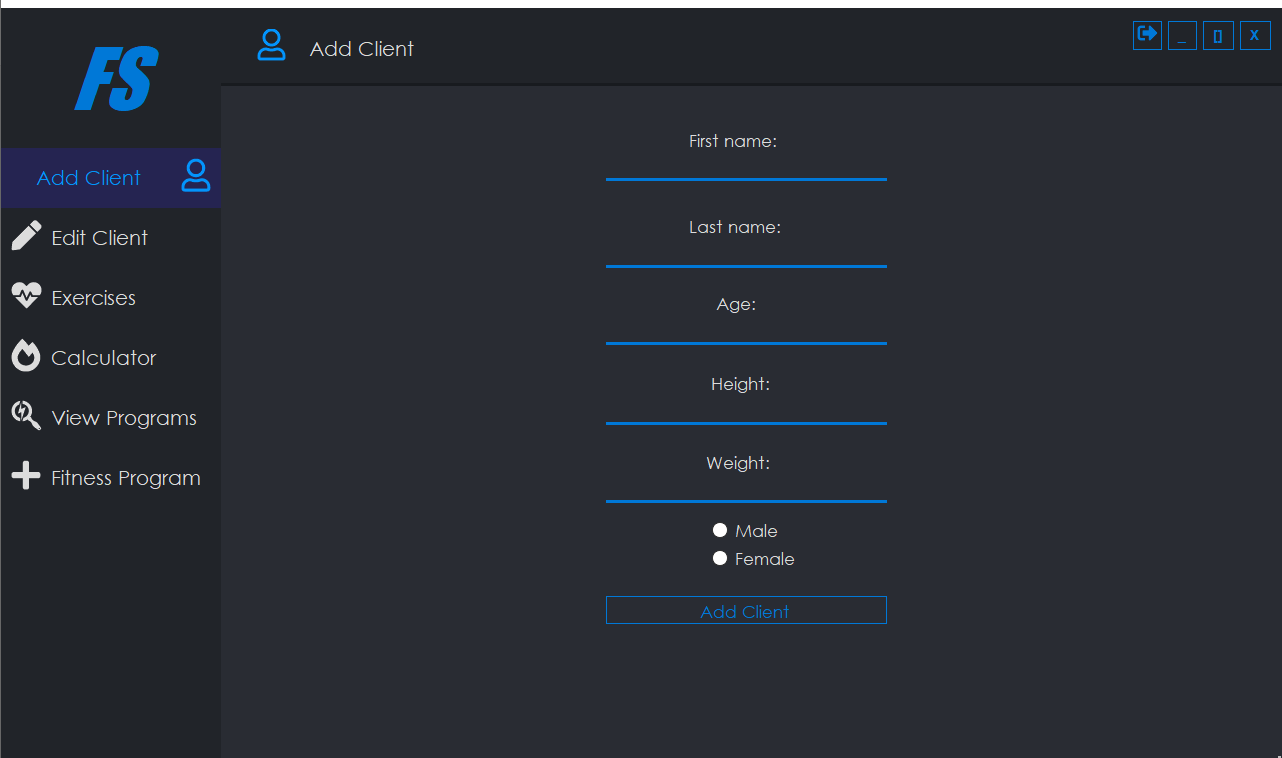
1. Логване в системата



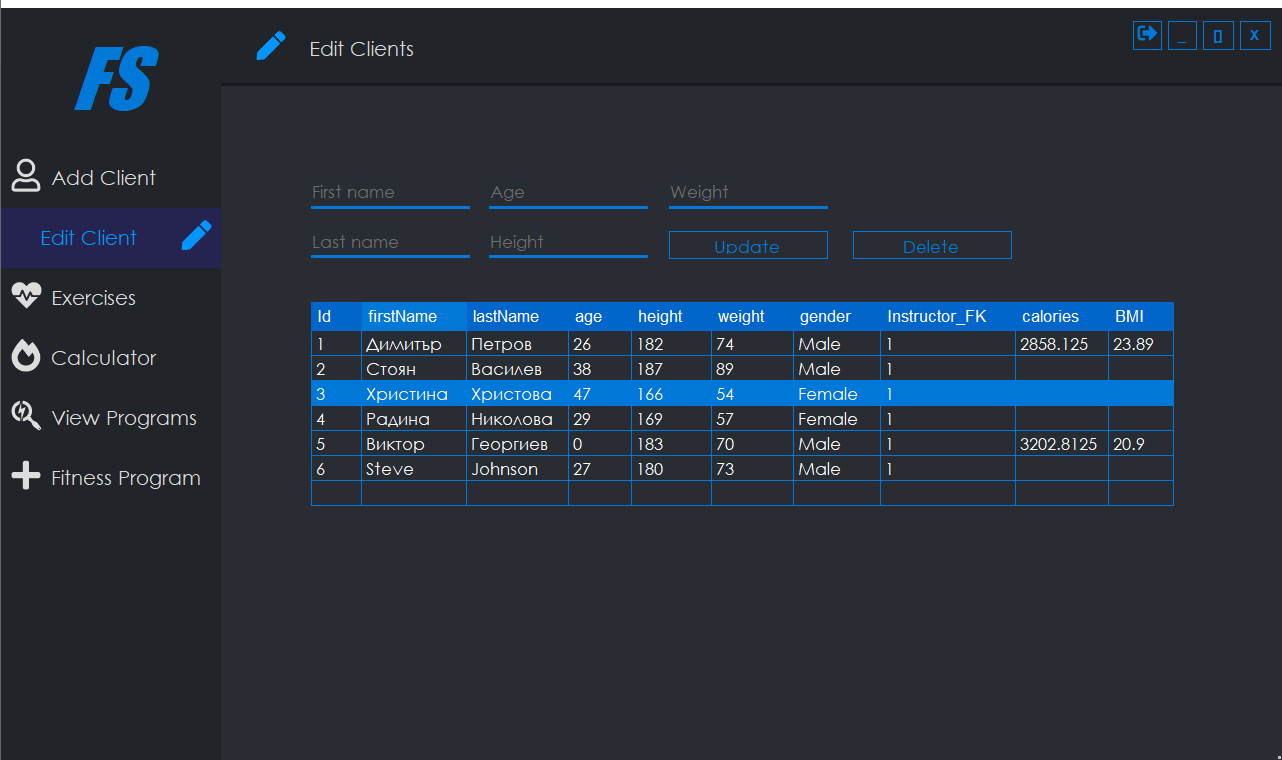
1. Главно меню



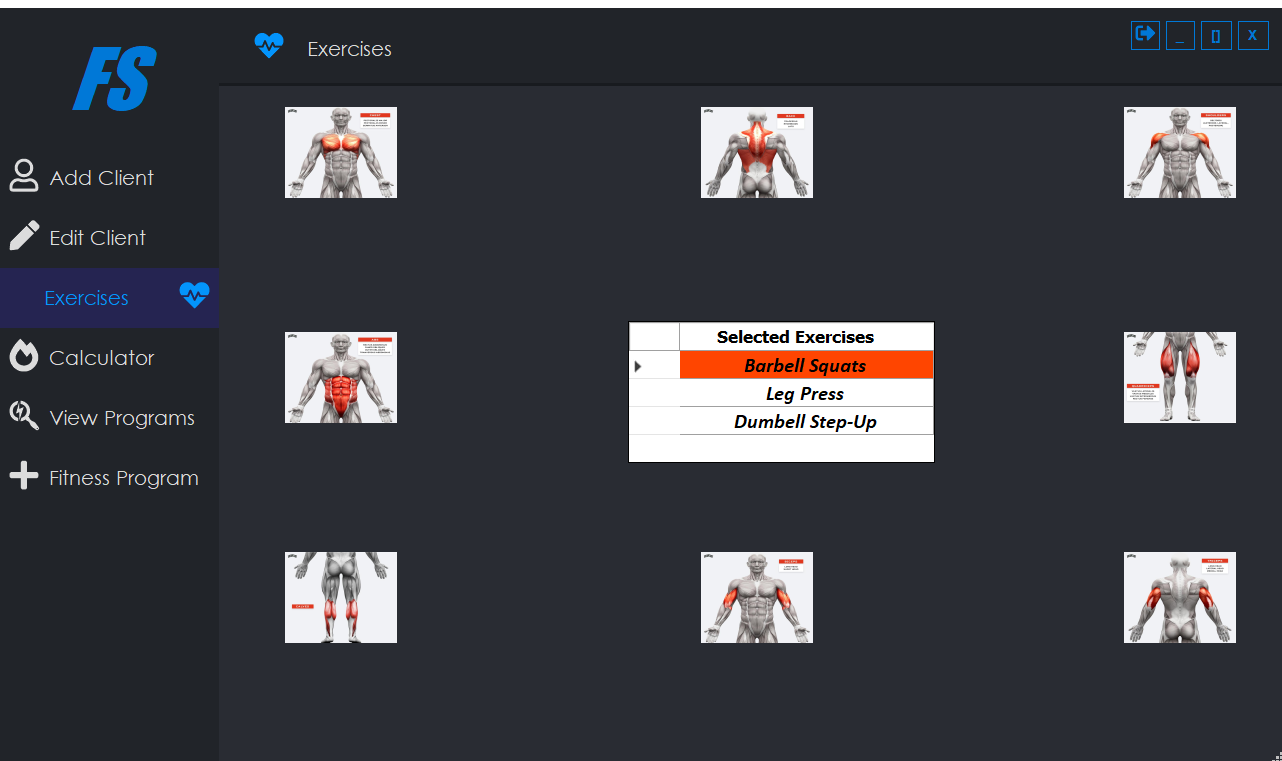
1. Вкарване на клиент в системата



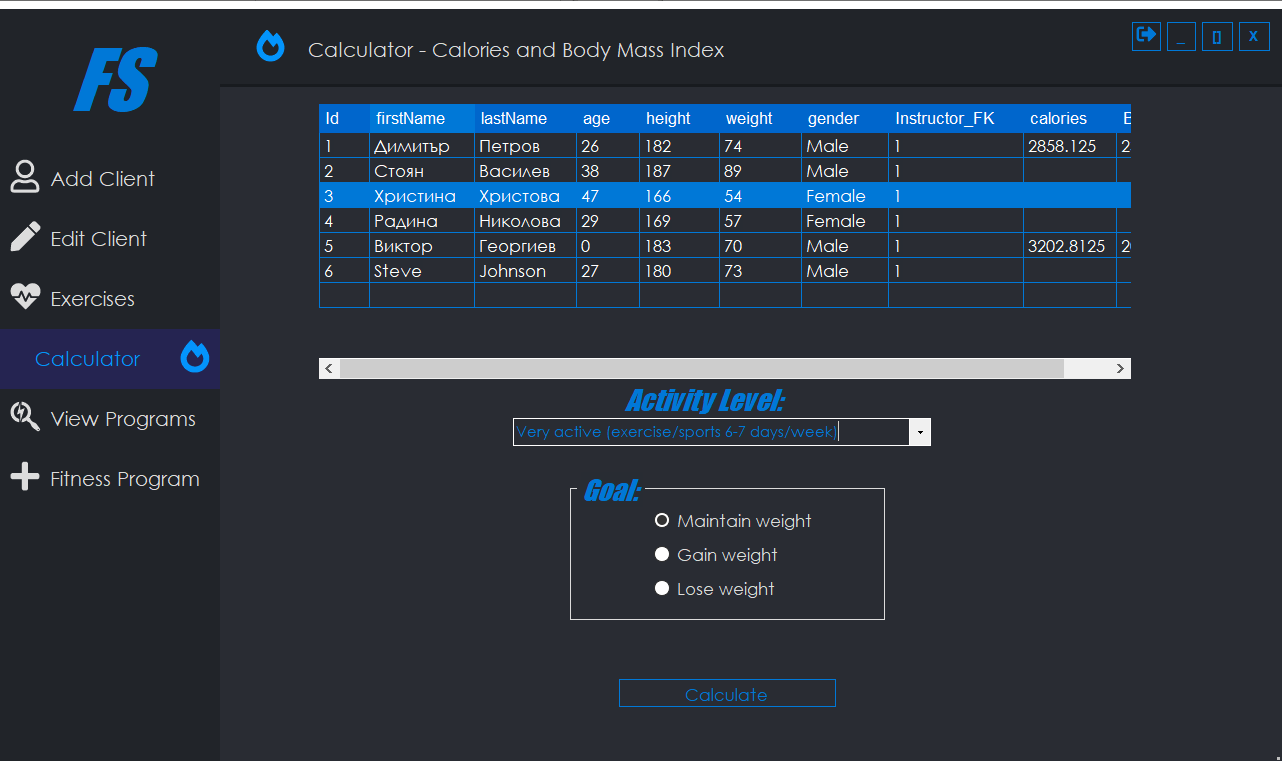
1. Редактиране на клиенти



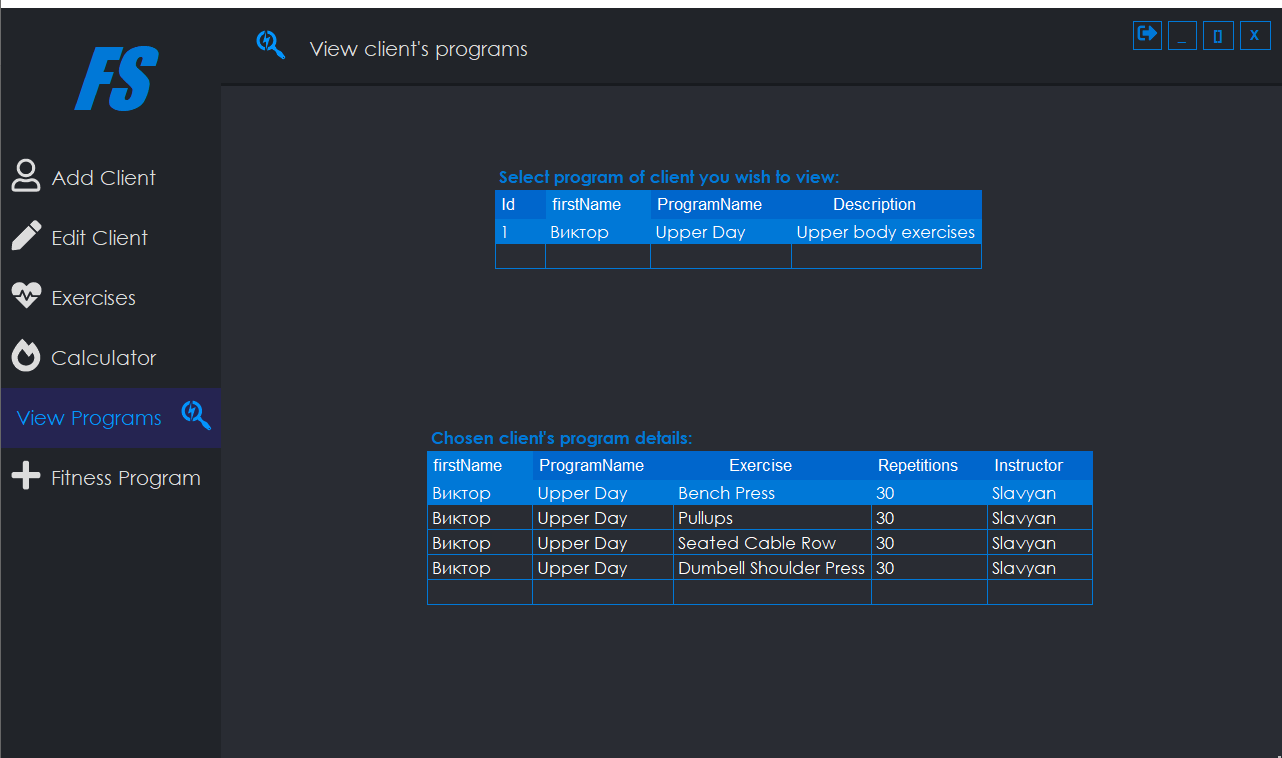
1. Преглед на упражнения



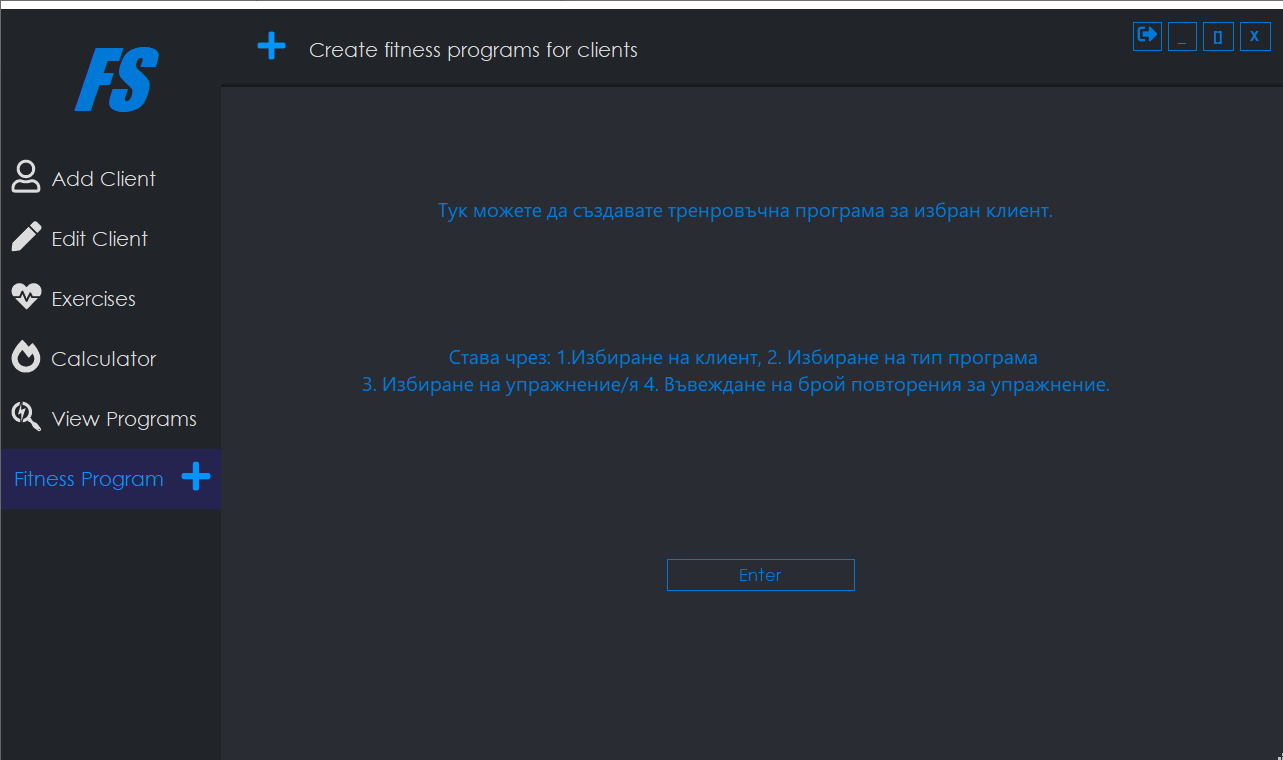
1. Калкулатор



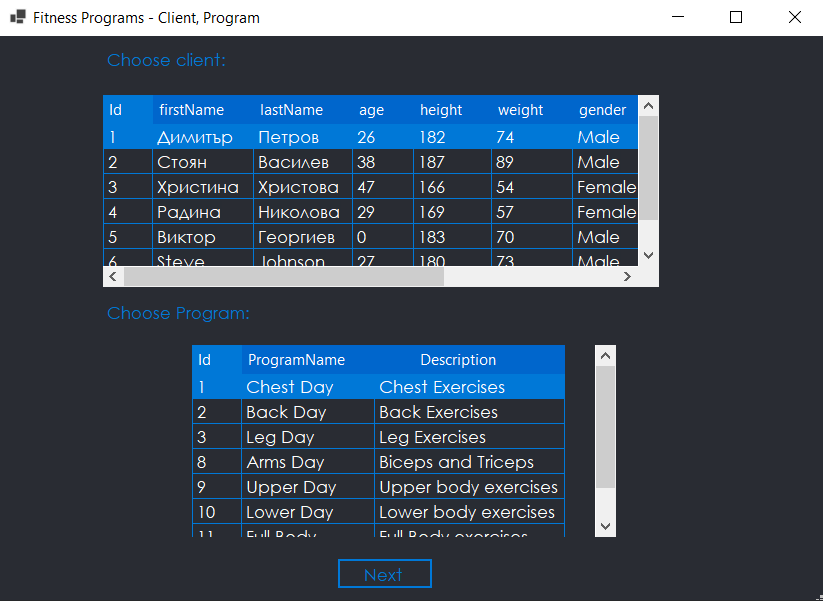
1. Преглед на всички фитнес програми в системата



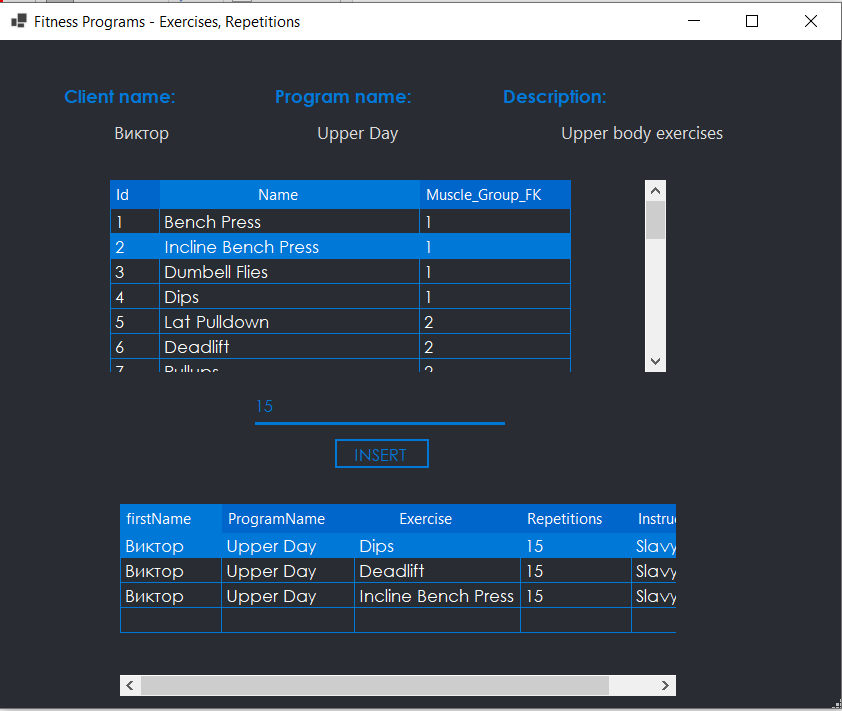
1. Информация за създаване на фитнес програми



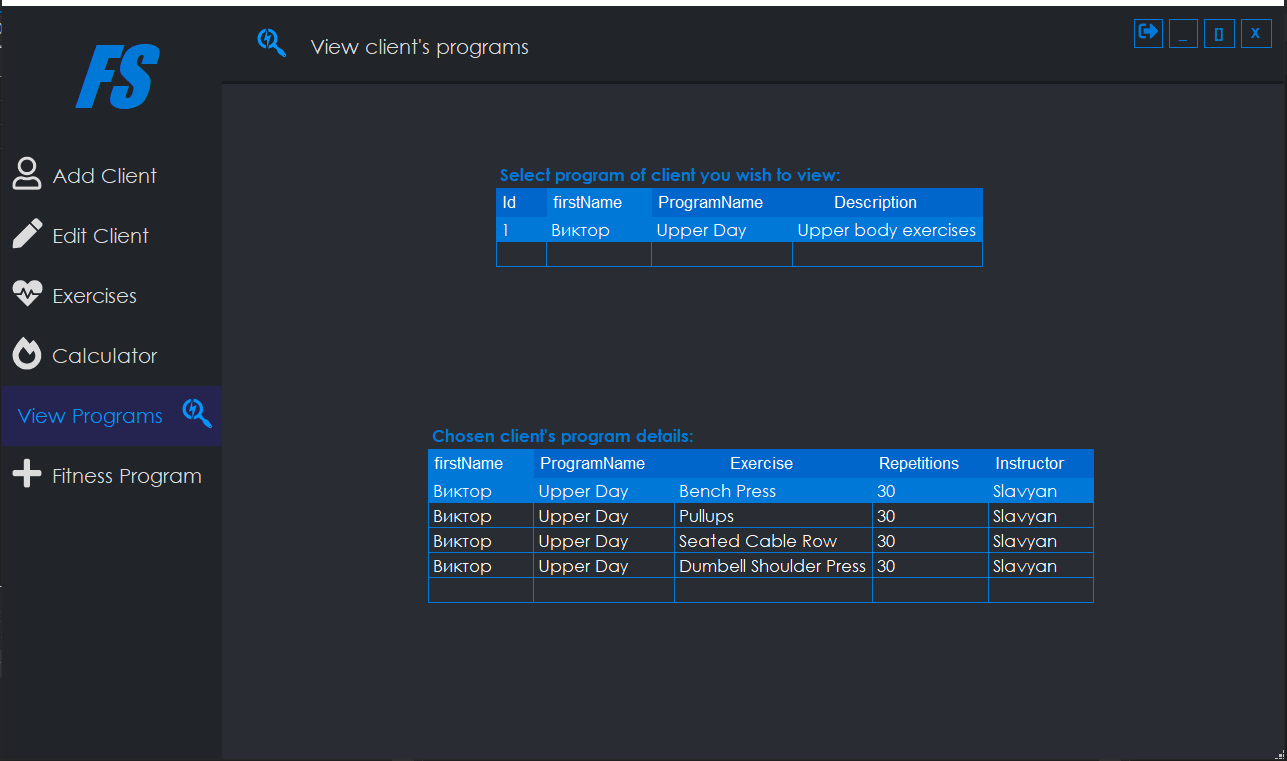
1. Избиране на клиент и име на програма за създаване на фитнес програма



1. Вкарване на упражнения и брой повторения за упражнение



1. Преглед на всички готови фитнес програми



1. Възможни бъдещи подобрения
   1. Усъвършенстване на функционалността за добавяне на тренировъчна програма.
   2. Смяна/Добавяне на други технологии за разработката на приложението. Пример: Windows Forms с WPF, Добавяне на framework за улесняване на работата с база данни.
   3. Добавяне на още функционалности към апликацията. Пример: Възможност клиент-а да се логва в системата и да има свои лични екстри като следене на прогреса, възможност за добавяне на храните, които е консумирал през деня в системата и така да се изчислява колко калории съдържат дадените храни.
   4. Фитнес инструкторът да има възможност да съставя също и хранителни режими.