“Frizētavas aplikācija”

Programmatūras apraksts (PPA, PPS, LI)

Veidoja : Rihards Stūrītis 12.G

Gulbenes novada vidusskola

Saturs

1. Ievads

1.1 Darba nolūks :

Šis darbs tika izveidots, lai izskaidrotu veidoto programmatūras risinājumu.

1.2 Darbības sfēra :

Piedāvātā programma ir veidota frizētavām, kurām nav pakalpojumu pieteikšanas un produktu pasūtīšanas sistēmas. Programma veidota C# vidē ar Windows forms .NET framework.

1.3 Definīcijas, akronīmi, saīsinājumi :

- PPS - programmatūras prasību specifikācija
- ER - Relāciju diagramma
- LI – Lietotāja interfeis

1.4 Pārskats

2.nodaļā "Darba apraksts" ir aprakstīts :

1) **Problēmas izpēte un analīze – Kāda problēma pastāv un kādi ir pieejamie risinājumi**

2) **Programmatūras izstrādes plāns – Kā programma tiks izveidota**

3) **Programmatūras dzīves cikls – Izmantotais dzīves cikls programmas izstrādes laikā**

4) **Datubāzes plānojums – Datubāzes vizuāls attēlojums un izskaidrojums**

5) **Bibliotēkas – projektā izmantotās bibliotēkas**

6) **Programmas datu tipi – programmā izmantotie datu tipi**

7) **Programmas funkcijas – Programmā izveidotās un izmantotās funkcijas**

8) **Programmas testēšana – Testēšanas rezultātu apkopojums**

9) **Projekta ierobežojumi – Vispārējie un programmas ierobežojumu apraksts**

2. Darba apraksts

2.1 Problēmas izpēte un analīze

Galvenā izpētes metode projekta izvēlei ir anketēšana. Ar anketēšanas kvantitatīvajiem datiem var noteikt, kurš projekts citiem liekas visaktuālākais un attiecīgi var piedāvāt veidus, kā uzlabot to gan vizuāli gan funkcionāli.

2.2 Programmatūras izstrādes plāns

Projekta izpēte un analīze – Veikt izpēti par frizētavas darbības procesiem, esošajām sistēmām un noteikt vajadzīgos punktus priekš programmas izstrādes.

Lietotāja saskarnes izstrāde – Izstrādāt lietotāja saskarni un programmas plānojumu. Noteikt kādus vizuālos elementus izmantot programmā, lai tas būtu moderns un glīts.

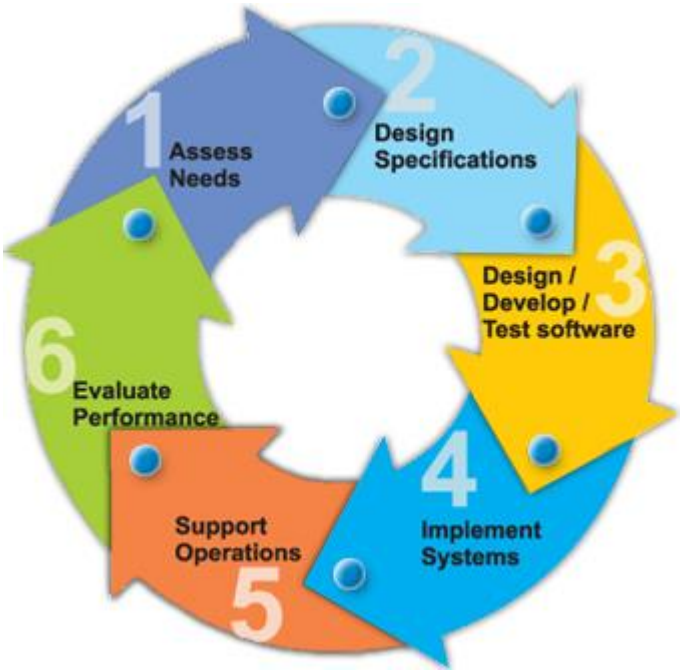
Datubāzes izstrāde – Izveidot izplānotu datubāzes ER modeli un implementēt to SQLite vidē.

Drošības un datu validācijas funkcijas – Izveidot dažādas funkcijas, kas pārbaude lietotāja datu ievadi, kā arī pasargā lietotāja informāciju datubāzē ar šifrēšanu.

Programmas testēšana - Veikt dažādas pārbaudes ar atšķirīgiem datiem un analizēt rezultātu ar paredzēto.

Trūkumu analīze un iespējamo uzlabojumu atrašana – Noteikt programmas trūkumus pēc testēšanas un noteikt punktus, kurus var uzlabot vai ieviest.

2.3 Programmēšanas dzīves cikls



Projekta izveides laikā tiks izmantots Sistēmu izstrādes dzīves cikls (SDLC) dzīves cikls. Sekojot dzīves cikla modelim var efektīvi izveidot kvalitatīvu programmatūru īsā laikā. Dzīves cikls samazina projekta riskus plānojot uz priekšu, lai programmatūra atbilst klientu vēlmēm ražošanas laikā un pēc tam.

Viens no galvenajiem iemesliem kāpēc tika izvēlēts šis dzīves cikls ir tāpēc, ka tas ir cieši saistītas ar pirms tam izveidoto programmatūras izstrādes plānu. Dzīves ciklam sekos līdz 3. punktam, jo programma nav paredzēta komerciālai izmantošanai.

2.4 Licences

Projekts izmantos MIT licenci, jo programma netiks izmantota komerciāli un citiem būs iespēja projektu pilnveidot vai pārveidot prieks savam vajadzībām.

Newtonsoft.Json izmanto MIT licenci, kas nosaka maz ierobežojumu atkārtotai izmantošanai gan privāti, gan komerciāli.

2.5 Izmantotās programmas

GitHub - izstrādātāju platforma, kas ļauj izstrādātājiem izveidot, uzglabāt, pārvaldīt un koplietot savu kodu. Piedāvātā programma ir pieejama visiem GitHubā bez nekādiem ierobežojumiem

Visual studio – Microsoft izveidota vide priekš programmu izveides c# valodā. Piedāvātā programma ir rakstīta, izmantojot Windows forms .NET framework, ko veidoja Microsoft un ir iekļauta Visual studio.

DB browser for SQLite - DB Browser for SQLite ir augstas kvalitātes, vizuāls, atvērtā pirmkoda rīks, lai izveidotu, izstrādātu un rediģētu ar SQLite saderīgus datu bāzes failus. Programmas datubāze tika veidota šeit.

DBdiargam.io - Bezmaksas tiešsaistes rīks attiecību diagrammu zīmēšanai, rakstot kodu. Programmas ER modelis tika veidots šajā rīkā.

2.6 Datubāzes plānojums

Šajā nodaļā tiks paskaidrota datu tipu lietošana katrā tabulā. Zem paskaidrojuma pievienots datubāzes attēlojums kā ER modelis, kā arī kā SQLite tabulas. Tabulas veidotas programmā DB browser for SQLite.

Datu tipi

Tabula Klients :

Klients_ID – Integer – AI (auto increment) - PK (Primary key) - Lietotāja unikāls identifikators, ar katru jaunu lietotāju skaitlis paliek lielāks par vienu vērtību.

Vards – Text – Lietotaja vārds, kas paredzēts kā string tipa vērtība bez skaitļiem un simboliem.

Uzvards – Text – Lietotaja uzvārds, kas paredzēts kā string tipa vērtība bez skaitļiem un simboliem.

Telefona_numurs - Integer – Lietotāja telefona numurs, kas paredzēts kā skaitlis, tāpēc Int tipa vērtība.

E_pasts – Text – Lietotāja E-pasts, kas var sastāvēt no skaitļiem un burtiem, tāpēc string tipa vērtība.

Parole – Text – Lietotāja parole, kas glabāsies datubāzē kā paroles Hash, kas sastāv no cipariem un burtiem, tāpēc string tipa vērtība.

Tabula Pasutijums :

Pasutijuma_ID - Integer – AI (auto increment) – PK (Primary key) - Pasūtijuma unikāls identifikators, ar katru jaunu pasūtijumu skaitlis paliek lielāks par vienu vērtību.

Klients_ID – Integer – FK (foreign key) – Lietotāja ID, kas nāk no tabulas Klients.

PasutijumaDatums – Text – Datums, kad veikts pasūtijums.

Produkta_ID – Integer – FK (foreign key) – Produkta ID, kas nāk no tabulas Produkts.

Skaits – Integer – vesels skaitlis, kas apraksta nopirkto produktu skaitu.

Tabula Produkts :

Produkta_ID - Integer – AI (auto increment) – PK (Primary key) - Produkta unikāls identifikators, ar katru jaunu produktu skaitlis paliek lielāks par vienu vērtību.

Produkta_nosaukums – Text – Produkta nosaukums.

Produkta_cena – Integer – Produkta vesela cena.

Razotajs – Text – Produkta ražotāja nosaukums.

Tabula Frizieris :

Frizieris_ID - Integer – AI (auto increment) – PK (Primary key) - Pasūtijuma unikāls identifikators, ar katru jaunu pasūtijumu skaitlis paliek lielāks par vienu vērtību.

Vards – Text - Friziera vārds.

Uzvards – Text -Friziera uzvārds.

Telefona_numurs – Integer – Friziera telefona numurs.

Tabula Pieteikums :

Pieteikuma_ID - Integer – AI (auto increment) – PK (Primary key) - Pieteikuma unikāls identifikators, ar katru jaunu pieteikumu skaitlis paliek lielāks par vienu vērtību.

Klients_ID – Integer -Lietotāja ID, kas nāk no tabulas Klients.

Frizieris_ID – Integer - Friziera ID, kas nāk no tabulas Frizieris.

Pakalpojums_ID – Integer - Pakalpojuma ID, kas nāk no tabulas Pakalpojums.

Vai_atcelts – Text – Glabā atslēgas vārdus “Jā” un “Nē”, lai zinātu vai pieteikums ir atcelts. Katrs pieteikums automātiski tiek reģistrēts kā “Nē”. Tiek izmantoti atslēgas vārdi, jo SQLite nav vieda kā definēt Bool tipa vērtības datubāzē.

Datums – Text – Datums, kad tika pieteikts pieteikums.

Tabula Pakalpojums :

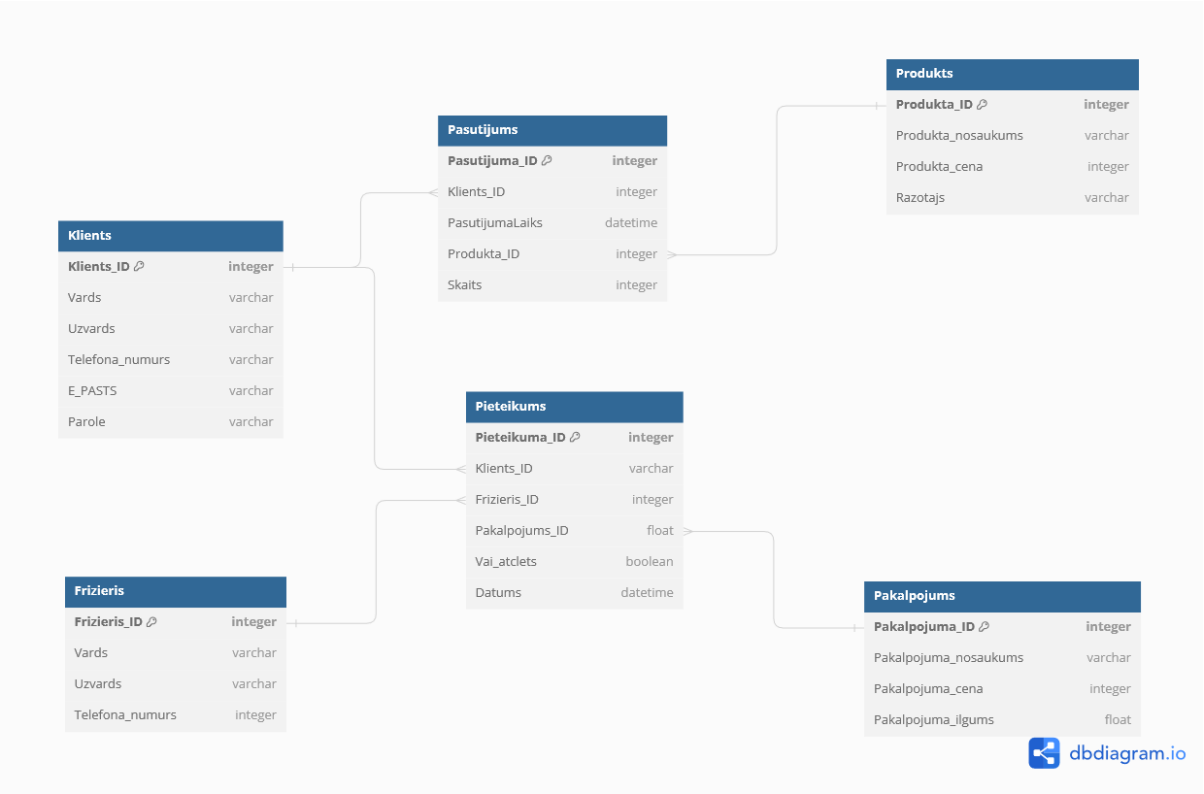
Pakalpojuma_ID - Integer – AI (auto increment) – PK (Primary key) - Pakalpojuma unikāls identifikators, ar katru jaunu pakalpojumu skaitlis paliek lielāks par vienu vērtību.

Pakalpojuma_nosaukums – Text – Pakalpojuma nosaukums.

Pakalpojuma_cena – Integer – Pakalpojuma maksa pilnā cenā.

Pakalpojuma_ilgums – Integer – Pakalpojuma ilgums pilnās stundās.

ER modelis :



Datubāzes tabulas SQLite datubāzē :

Name	Type	Schema
Tables (7)		
Frizieris		CREATE TABLE "Frizieris" ("Frizieris_ID" INTEGER, "Vards" TEXT, "Uzvards" TEXT, "Telefona_numurs" INTEGER, PRIMARY KEY("Frizieris_ID" AUTOINCREMENT))
Klients		CREATE TABLE "Klients" ("Klients_ID" INTEGER, "Vards" TEXT, "Uzvards" TEXT, "Telefona_numurs" INTEGER, "E_pasts" TEXT, "Parole" TEXT, PRIMARY KEY("Klients_ID" AUTOINCREM
Pakalpojums		CREATE TABLE "Pakalpojums" ("Pakalpojuma_ID" INTEGER, "Pakalpojuma_nosaukums" INTEGER, "Pakalpojuma_cena" INTEGER, "Pakalpojuma_ilgums" INTEGER)
Pasutijums		CREATE TABLE "Pasutijums" ("Pasutijuma_ID" INTEGER, "Klients_ID" INTEGER, "Pasutijuma_datums" TEXT, "Produkta_ID" INTEGER, "Skaits" INTEGER, PRIMARY KEY("Pasutijuma_ID"
Pieteikums		CREATE TABLE "Pieteikums" ("Pieteikuma_ID" INTEGER, "Klients_ID" INTEGER, "Frizieris_ID" INTEGER, "Pakalpojums_ID" INTEGER, "Vai_atcelts" TEXT, "Datums" TEXT, PRIMARY KEY(
Produkts		CREATE TABLE "Produkts" ("Produkta_ID" INTEGER, "Nosaukums" TEXT, "Cena" REAL, "Razotajs" TEXT, PRIMARY KEY("Produkta_ID" AUTOINCREMENT))
sqlite_sequence		CREATE TABLE sqlite_sequence(name,seq)

2.7 Bibliotēkas

System - nodrošina vispārīgas sistēmas funkcionalitātes piekļuvi.

System.Collections.Generic - nodrošina datu struktūras.

System.ComponentModel - satur klases un interfeisus.

System.Data - nodrošina datu piekļuvi un manipulāciju iespējas.

System.Drawing - ļauj veikt grafikas operācijas.

System.Text - nodrošina klases un metodes darbam ar tekstuāliem datiem.

System.Threading.Tasks - piedāvā iespēju izveidot un pārvaldīt asinhronas darba vienības.

System.Windows.Forms - ļauj izveidot Windows Forms lietotāja saskarni un interaktīvus lietotāja pielāgojumus.

System.Security.Cryptography - nodrošina funkcijas un algoritmus datu šifrēšanai, hash funkcijas aprēķināšanai un drošības funkcijām.

System.Data.SQLite - ļauj veikt datu piekļuvi un manipulāciju izmantojot SQLite datu bāzi.

System.Net.Http - nodrošina iespēju veikt HTTP komunikāciju no .NET aplikācijām.

Newtonsoft.Json - populāra JSON datu formāta apstrādes bibliotēka .NET vidē. Tā piedāvā plašas funkcijas, lai konvertētu objektus no/uz JSON formātu, analizētu JSON virknes un veiktu manipulācijas ar JSON datiem.

2.8 Programmas datu tipi

Klases :

public class Registresanas_dati – Klase, kas satur lietotāja ievadītos datus reģistrēšanās formā, kā arī satur funkcijas registrelietotaju() un Konekcija().

static class Datu_manipulesana – Glabā visas vajadzīgās funkcijas priekš datu ievades pārbaudes, šifrēšanas un lauku notīrīšanas. Iekļautās funkcijas – NotiritTekstu(params MaskedTextBox[] textBoxes), SHA_256HASH(string parole), VaiTuksiLauki(params MaskedTextBox[] textBoxes), Varda_parbaude(string vards), VaiTikaiCipari(string teksts).

2.9 Programmas funkcijas

private async Task laikanolase() – funkcija, kas savienojas ar timeapi.io API, lai iegūtu Latvijas pašreizējo laiku.

private bool IsValidUser(string E_pasts, string Parole) – Pārbaude vai ievadītie dati pierakstīšanās formā sakrīt ar kādiem no reģistrētajiem lietotājiem datubāzē.

public void registrelietotaju() – Ievieto lietotāja ievadītos datus datubāzē.

public static SQLiteConnection Konekcija() – Izveido savienojumu ar datubāzi.

public static string SHA256_HASH(string Parole) public static bool - Pārveido ievadīto paroli uz hashu, izmantojot SHA256 algoritmu.

VaiTuksiLauki(params MaskedTextBox[] textBoxes) – Notīra visus ierakstītos datus visos laukos

public static bool Varda_parbaude(string vards) – pārbauda, vai ievadītais teksts sastāv tikai no burtiem.

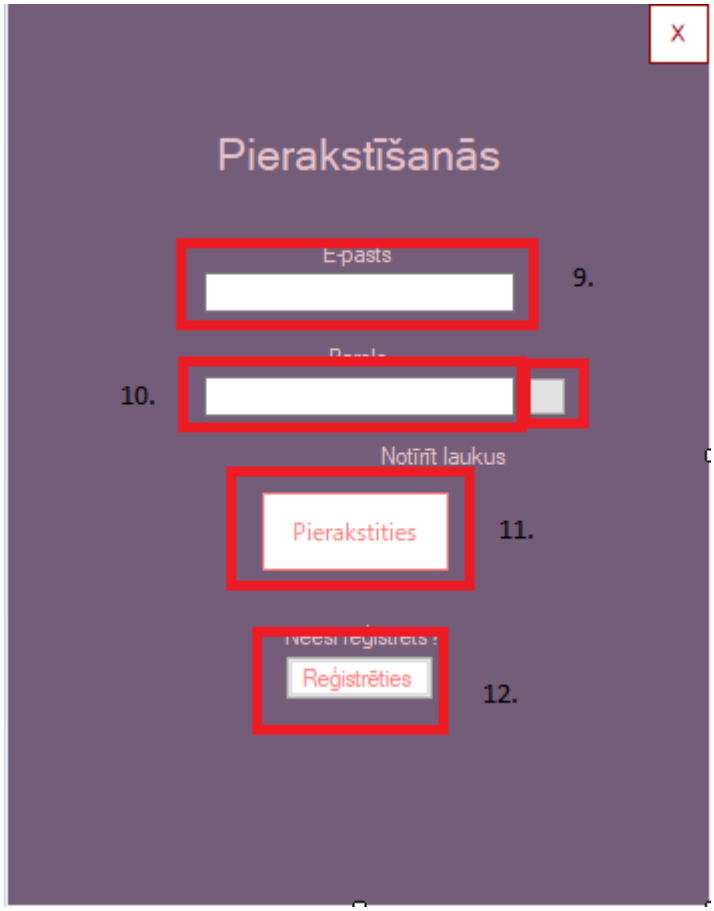
public static bool VaiTikaiCipari(string teksts) – pārbauda, vai ievadītā vērtība sastāv tikai no veseliem cipariem.

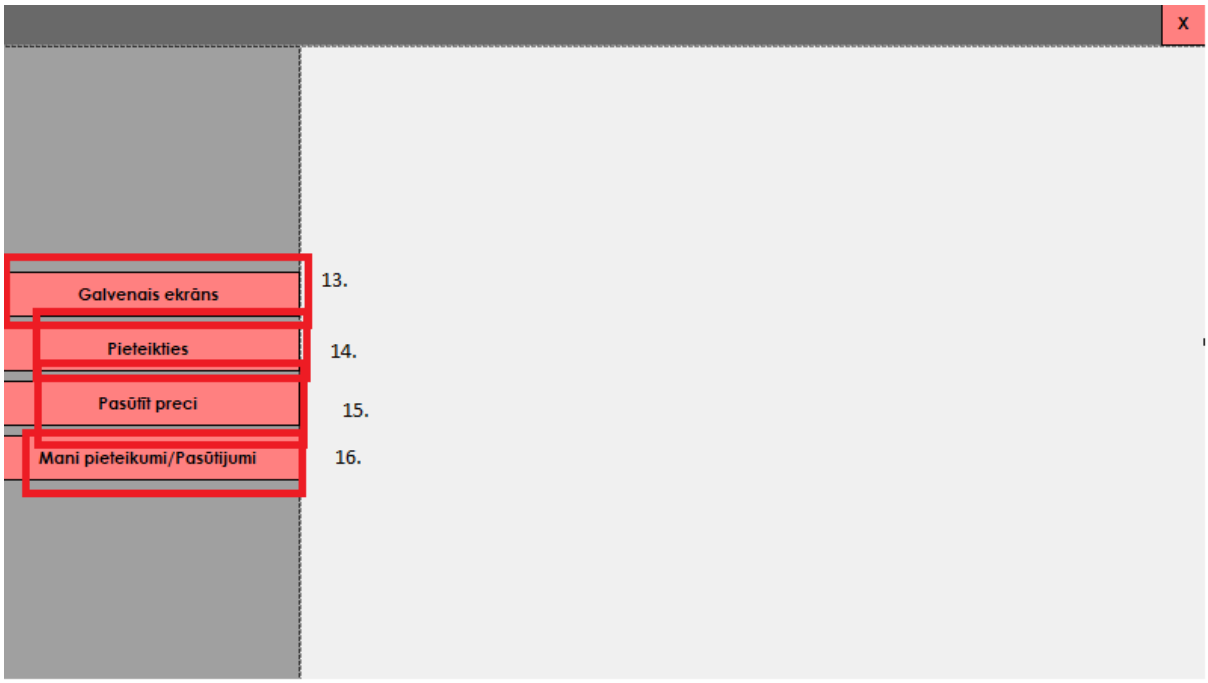
public static void NotiritTekstu(params MaskedTextBox[] textBoxes) – Notīra visas aizpildītos laukus.

2.10 Programmas testēšana

Funkcija	Ievade	Izvade	Strādā	Nestrādā	Komentārs
SHA256_HASH	parole	24f102db6da0ceec06d8aa4b0d1fe47b7b81921e7703a046fbc90c30914e2705	x		Teksts veiksmīgi tika pārveidots uz hashu
Varda_parbaude	Andrejs123	("Lūdzu ievadiet vārdu bez simboliem un cipariem!");	x		Vārdā ir cipari, ko funkcija atrada
VaiTikaiCipari	asd	("Telefona numurā atļauti tikai cipari!");	x		Ievadē izmantoti tikai burti
IsValidUser	Abs,abrs	("Ievadiet pareizi E-pastu un paroli!");	x		Lietotājs ar šādu vārdu un paroli nepastāv datubāzē

2.11 Lietotāja saskarnes izskaidrojums





1. “Atpakaļ” poga, kas lietotāju atgriež atpakaļ uz pierakstīšanās logu

2. Lietotāja vārda ievade

3.Lietotāja uzvārda ievade

4. Lietotāja E-pasta ievade

5. Lietotāja telefona numura ievade

6. Lietotāja paroles ievade

7. Poga, kas parāda ievadīto paroli

8. Reģistrēšanās poga

9. Lietotāja E-pasta ievade

10. Lietotāja parole

11. Pierakstīšanās poga

12. Poga, kas noved uz reģistrēšanās logu

13. Galvenā ekrāna poga

14. Poga, kas atver uz pieteikuma sadaļu

15. Poga, kas atver preču pasūtīšanas sadaļu

16. Poga, kas parāda pašreiz pierakstītā lietotāja pieteikumus un pasūtījumus

2.12 Projekta ierobežojumi

Programmas ierobežojumi :

Programmā lietotājs nevar izveidot pieteikumu, pasūtīt preci un aplūkot savus pieteikumus un pasūtījumus, jo programma nav pabeigta. Programma nestrādā ārpus Visual studio.

Pielikumi

```
using System;
using System.Collections;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Runtime.InteropServices.WindowsRuntime;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Windows.Forms.VisualStyles;
using System.Data.SQLite;
using System.Net.Http;
using Newtonsoft.Json;

namespace Projekta_darbs_Rihards_frizetava
{
    public partial class Form1 : Form
    {
        private readonly HttpClient client;
        private readonly Timer timer;
        public Form1()
        {
            InitializeComponent();
            client = new HttpClient();
            timer = new Timer();
            timer.Interval = 1000; // Intervals 1 sekunde (1000 milisekundes)
            timer.Tick += Timer_Tick;
            timer.Start();
            this.StartPosition = FormStartPosition.CenterScreen;
        }
        private async void Timer_Tick(object sender, EventArgs e)
        {
            await laikanolase();
        }

        private void Form1_Load(object sender, EventArgs e)
        {

        }
        public void loadform(object Form)
        {
            if (this.Galvenais_panelis.Controls.Count > 0)
                this.Galvenais_panelis.Controls.RemoveAt(0);
            Form f = Form as Form;
            f.TopLevel = false;
            f.Dock = DockStyle.Fill;
            this.Galvenais_panelis.Controls.Add(f);
            this.Galvenais_panelis.Tag = f;
            f.Show();
        }


        private void Pieteikumi_Pasutijumi_Click(object sender, EventArgs e)
        {
            loadform(new Mani_pasutijumi_pieteikumi());
        }

        private void Pieteikties_Click(object sender, EventArgs e)
```

```
        {
            loadform(new Pieteikties());
        }

        private void Pasutit_preci_Click(object sender, EventArgs e)
        {
            loadform(new Pasutit_preci());
        }

        private void Aizversanas_poga_Click(object sender, EventArgs e)
        {
            this.Dispose();
        }

        private void Galvenais_ekrans_Click(object sender, EventArgs e)
        {
            loadform(new Galvenais_ekrans());
        }

        private async Task laikanolase()
        {
            //izlasa no API laiku un izvada to uz saskarnes
            try
            {
                string apiUrl =
"https://timeapi.io/api/Time/current/zone?timeZone=Europe/Riga";
                HttpResponseMessage response = await client.GetAsync(apiUrl);
                response.EnsureSuccessStatusCode();
                string responseBody = await response.Content.ReadAsStringAsync();
                dynamic jsonObject = JsonConvert.DeserializeObject(responseBody);
                string time = jsonObject.time;
                label1.Text = time;
            }
            catch (HttpRequestException ex)
            {
                MessageBox.Show($"API request failed: {ex.Message}");
            }
        }

    }


}
```

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Security.Cryptography;
using System.Data.SQLite;
namespace Projekta_darbs_Rihards_frizetava
{
    public partial class Pierakstisanas : Form
    {
```

```
        public Pierakstisanas()
        {
            InitializeComponent();
            this.StartPosition = FormStartPosition.CenterScreen;
        }

        private void Pierakstisanas_Load(object sender, EventArgs e)
        {

        }


        private void Registresanas_parvietosana_Click(object sender, EventArgs e)
        {
            Registresanas reg = new Registresanas();
            reg.Show();
            this.Hide();

        }

        private void Radit_tekstu_MouseDown(object sender, MouseEventArgs e)
        {
            if (e.Button == MouseButtons.Left)
            {
                pierakstisanas_parole.UseSystemPasswordChar = false;
            }
        }

        private void Radit_tekstu_MouseUp(object sender, MouseEventArgs e)
        {
            if (e.Button == MouseButtons.Left)
            {
                pierakstisanas_parole.UseSystemPasswordChar = true;
            }
        }
        // Pieraksta lietotaju aplikacija
        private void Pierakstities_click(object sender, EventArgs e)
        {
            if(Datu_manipulesana.VaiTuksiLauki(Pierakstisanas_e_pasts,
pierakstisanas_parole))
            {
                if(IsValidUser(Pierakstisanas_e_pasts.Text,
pierakstisanas_parole.Text))
                {
                    MessageBox.Show("Ielogjies");
                    Form1 f = new Form1();
                    f.Show();
                    this.Hide();
                }
                else
                {
                    MessageBox.Show("Ievadiet pareizi E-pastu un paroli!");
                }

            }
            else
            {
                MessageBox.Show("Aizpildiet laukus");
            }
        }
        // Notira datus ievaditajos laukos
        private void Notirit_laukus(object sender, EventArgs e)
        {
```

```
            Datu_manipulesana.NotiritTekstu(Pierakstisanas_e_pasts,
pierakstisanas_parole);
        }
        private bool IsValidUser(string E_pasts, string Parole)
        {
            string query = "SELECT * FROM Klients WHERE E_pasts=@Epasts AND
Parole=@Password";

            using (SQLiteConnection connection = Registresanas_dati.Konekcija())
            {
                try
                {
                    connection.Open();

                    if (connection.State == ConnectionState.Open)
                    {
                        using (SQLiteCommand command = new SQLiteCommand(query,
connection))
                        {
                            command.Parameters.AddWithValue("@Epasts", E_pasts);
                            command.Parameters.AddWithValue("@Password",
Datu_manipulesana.SHA256_HASH(Parole));

                            using (SQLiteDataAdapter adapter = new
SQLiteDataAdapter(command))
                            {
                                DataTable dt = new DataTable();
                                adapter.Fill(dt);

                                command.ExecuteNonQuery();
                                connection.Close();

                                if (dt.Rows.Count > 0)
                                {
                                    return true;
                                }
                                else
                                {
                                    return false;
                                }
                            }
                        }
                    }
                }
                catch (Exception ex)
                {
                    MessageBox.Show("Kļūda: " + ex.Message);
                }
            }

            return false; // Return false if any error occurs
        }

        private void aiztaisit_programmu_Click(object sender, EventArgs e)
        {
            this.Dispose();
        }
    }
}


using System;
```

```
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Data.SQLite;
using System.Diagnostics.Eventing.Reader;
using System.Drawing;
using System.Linq;
using System.Security.Cryptography;
using System.Security.Policy;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Projekta_darbs_Rihards_frizetava
{
    public partial class Registresanas : Form
    {
        public Registresanas()
        {
            InitializeComponent();
            this.StartPosition = FormStartPosition.CenterScreen;
        }
        private void Registresanas_Load(object sender, EventArgs e)
        {

        }
        private void notirit_laukus_Click(object sender, EventArgs e)
        {
            Datu_manipulesana.NotiritTekstu(Vards_TTB, Uzvards_TTB, E_pasts_TTB,
Tel_nr_TTB, Parole_TTB);
        }

        private void Registresanas_poga_Click(object sender, EventArgs e)
        {


            if (!Datu_manipulesana.VaiTuksiLauki(Vards_TTB, Uzvards_TTB,
E_pasts_TTB, Tel_nr_TTB, Parole_TTB))
            {
                bridinajums_label.Visible = true;
            }
            else
            {
                // Pārbaude, vai telefona numura laukā ir tikai cipari
                if (!Datu_manipulesana.VaiTikaiCipari(Tel_nr_TTB.Text))
                {
                    MessageBox.Show("Telefona numurā atļauti tikai cipari!");
                    return;
                }

                // Pārbaude, vai e-pasta laukā ir '@' simbols
                if (!E_pasts_TTB.Text.Contains("@"))
                {
                    MessageBox.Show("Nederīgs e-pasta formāts!");
                    return;
                }

                if (Datu_manipulesana.Varda_parbaude(Vards_TTB.Text))
                {
                    Registresanas_dati reg = new Registresanas_dati();
                    reg.Vards = Vards_TTB.Text;
                    reg.Uzvards = Uzvards_TTB.Text;
                    reg.Telefona_nr = Convert.ToInt32(Tel_nr_TTB.Text);
```

```
                    reg.E_pasts = E_pasts_TTB.Text;
                    reg.Parole = Parole_TTB.Text;

                    reg.registrelietotaju();
                    Form1 form1 = new Form1();
                    form1.Show();
                    this.Dispose();
                }
                else
                {

                }

            }
        }

        private void Paradit_paroli_MouseDown(object sender, MouseEventArgs e)
        {
            Parole_TTB.UseSystemPasswordChar = false;
        }

        private void Paradit_paroli_MouseUp(object sender, MouseEventArgs e)
        {
            Parole_TTB.UseSystemPasswordChar = true;
        }

        private void aiztaisit_programmu_Click(object sender, EventArgs e)
        {
            this.Dispose();
        }

        private void Atpakal_poga_Click(object sender, EventArgs e)
        {
            Pierakstisanas pie = new Pierakstisanas();
            pie.Show();
            this.Dispose();
        }
    }

    public class Registresanas_dati
    {
        public string Vards { get; set; }
        public string Uzvards { get; set; }
        public string E_pasts { get; set; }
        public int Telefona_nr { get; set; }
        public string Parole { get; set; }

        public void registrelietotaju()
        {
            // Open the connection within the method scope
            using (SQLiteConnection connection = Konekcija())
            {
                try
                {
                    // Open the connection explicitly
                    connection.Open();

                    // Check if the connection is open
                    if (connection.State == ConnectionState.Open)
                    {
                        // Execute the command
                        string query = "INSERT INTO Klients (Vards, Uzvards,
Telefona_numurs, E_pasts, Parole) " +
```

```
                                        "VALUES (@Vards, @Uzvards,
@Telefona_numurs, @E_pasts, @Parole)";

                        using (SQLiteCommand command = new SQLiteCommand(query,
connection))
                        {
                            // Set parameters
                            command.Parameters.AddWithValue("@Vards", Vards);
                            command.Parameters.AddWithValue("@Uzvards", Uzvards);
                            command.Parameters.AddWithValue("@Telefona_numurs",
Telefona_nr);
                            command.Parameters.AddWithValue("@E_pasts", E_pasts);
                            command.Parameters.AddWithValue("@Parole",
Datu_manipulesana.SHA256_HASH(Parole));

                            // Execute command
                            command.ExecuteNonQuery();
                            command.Parameters.Clear();
                            connection.Close();
                        }
                    }
                    else
                    {
                        MessageBox.Show("Savienojums netika izveidots");
                    }
                }
                catch (Exception ex)
                {
                    MessageBox.Show("Kļūda: " + ex.Message);
                }
            }
        }

        // Move the connection method outside the class or refactor it to an
instance method
        public static SQLiteConnection Konekcija()
        {
            SQLiteConnection sqlite_conn;
            sqlite_conn = new SQLiteConnection(@"Data Source=
..\..\Faili\Frizetava_datubaze_SQLite.db; Version = 3;");
            try
            {
                // Do not open the connection here
            }
            catch (Exception ex)
            {
                // Handle exception
            }
            return sqlite_conn;
        }

    }

    static class Datu_manipulesana
    {
        public static void NotiritTekstu(params MaskedTextBox[] textBoxes)
        {
            foreach (MaskedTextBox textBox in textBoxes)
            {
                textBox.Clear();
            }
        }
```

```
        public static string SHA256_HASH(string Parole)
        {
            using (SHA256 sha256Hash = SHA256.Create())
            {
                byte[] bytes =
sha256Hash.ComputeHash(Encoding.UTF8.GetBytes(Parole));
                StringBuilder builder = new StringBuilder();
                for (int i = 0; i < bytes.Length; i++)
                {
                    builder.Append(bytes[i].ToString("x2"));
                }
                return builder.ToString();
            }
        }

        public static bool VaiTuksiLauki(params MaskedTextBox[] textBoxes)
        {
            foreach (MaskedTextBox textBox in textBoxes)
            {
                if (string.IsNullOrWhiteSpace(textBox.Text))
                {
                    return false;
                }
            }
            return true;
        }


        public static bool Varda_parbaude(string vards)
        {
            bool parbaude = true;

            foreach (char c in vards)
            {

                if (((c >= 'a' && c <= 'z') || (c >= 'A' && c <= 'Z')))

                {
                    continue;
                }
                else
                {
                    MessageBox.Show("Lūdzu ievadiet vārdu bez simboliem un
ciparіem!");
                    parbaude = false;
                    return parbaude;
                }

            }
            return parbaude;
        }
        public static bool VaiTikaiCipari(string teksts)
        {
            foreach (char c in teksts)
            {
                if (!char.IsDigit(c))
                {
                    return false;
                }
            }
            return true;
        }
    }
```

```
}
```

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Projekta_darbs_Rihards_frizetava
{
    public partial class Pieteikties : Form
    {
        public Pieteikties()
        {
            InitializeComponent();
        }
    }
}
```

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Projekta_darbs_Rihards_frizetava
{
    public partial class Pasutit_preci : Form
    {
        public Pasutit_preci()
        {
            InitializeComponent();
        }
    }
}
```

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Projekta_darbs_Rihards_frizetava
{
    public partial class Galvenais_ekrans : Form
    {
        public Galvenais_ekrans()
        {
```

```
            InitializeComponent();
        }

    }
}
```