На рисунке 1 изображена БД в Visual Studio

Изображение выглядит как текст, снимок экрана, программное обеспечение, число

Автоматически созданное описание

Рисунок 1 - база данных

На рисунке 2 изображена форма для ввода логина и пароля пользователя (не запущена)

Изображение выглядит как текст, снимок экрана, программное обеспечение, число

Автоматически созданное описание

Рисунок 2 – форма

На рисунках 3 - 8 изображены код для данной формы

Изображение выглядит как текст, снимок экрана

Автоматически созданное описание

Рисунок 3 – код

Изображение выглядит как текст, снимок экрана, программное обеспечение

Автоматически созданное описание

Рисунок 4 – код

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

Рисунок 5 – код

Изображение выглядит как снимок экрана, текст, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

Рисунок 6 – код

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

Рисунок 7 – код

Изображение выглядит как текст, снимок экрана, программное обеспечение

Автоматически созданное описание

Рисунок 8 – код

На рисунке 9 изображена главная форма (не запущена)

Изображение выглядит как текст, снимок экрана, программное обеспечение, Значок на компьютере

Автоматически созданное описание

Рисунок 9 – форма

На рисунках 10 - 24 изображены код для формы

Изображение выглядит как текст, снимок экрана

Автоматически созданное описание

Рисунок 10 – код

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

Рисунок 11 – код

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

Рисунок 12 – код

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

Рисунок 13 – код

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

Рисунок 14 – код

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

Рисунок 15 – код

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

Рисунок 16 – код

Изображение выглядит как текст, снимок экрана, программное обеспечение

Автоматически созданное описание

Рисунок 17 – код

Изображение выглядит как текст, Мультимедийное программное обеспечение, программное обеспечение, снимок экрана

Автоматически созданное описание

Рисунок 18 – код

Изображение выглядит как текст, снимок экрана

Автоматически созданное описание

Рисунок 19 – код

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

Рисунок 20 – код

Изображение выглядит как текст, снимок экрана, программное обеспечение

Автоматически созданное описание

Рисунок 21 – код

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 22 – код

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

Рисунок 23 – код

Изображение выглядит как снимок экрана, Мультимедийное программное обеспечение, программное обеспечение, Графическое программное обеспечение

Автоматически созданное описание

Рисунок 24 – код

На рисунке 25 изображена форма с данными о заявках (не запущена)

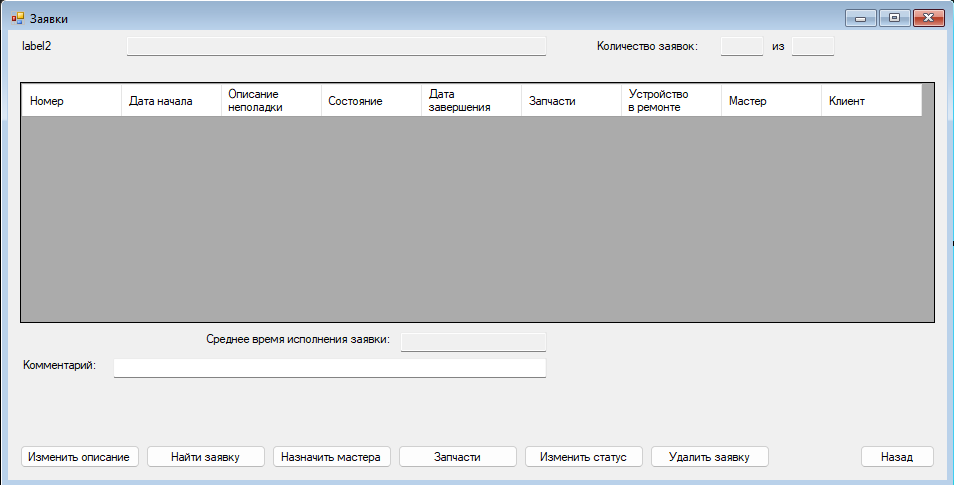


Рисунок 25 – форма

На рисунках 26 - 38 изображены код для этой формы

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

Рисунок 26 – код

Изображение выглядит как текст, снимок экрана, программное обеспечение, Шрифт

Автоматически созданное описание

Рисунок 27 – код

Изображение выглядит как текст, снимок экрана, программное обеспечение

Автоматически созданное описание

Рисунок 28 – код

Изображение выглядит как текст, Шрифт, снимок экрана

Автоматически созданное описание

Рисунок 29 – код

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

Рисунок 30 – код

Изображение выглядит как текст, снимок экрана, программное обеспечение, дисплей

Автоматически созданное описание

Рисунок 31 – код

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

Рисунок 32 – код

Изображение выглядит как текст, программное обеспечение, Мультимедийное программное обеспечение, Графическое программное обеспечение

Автоматически созданное описание

Рисунок 33 – код

Изображение выглядит как текст, снимок экрана

Автоматически созданное описание

Рисунок 34 – код

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

Рисунок 35 – код

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

Рисунок 36 – код

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

Рисунок 37 – код

Изображение выглядит как снимок экрана, текст, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

Рисунок 38 – код

На рисунке 39 изображена форма для изменения описании проблемы (не запущена)

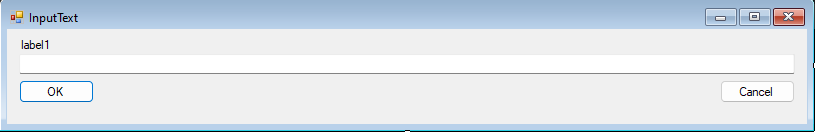


Рисунок 39 – форма

На рисунках 40 - 41 изображены код для данной формы

Изображение выглядит как текст, снимок экрана, программное обеспечение

Автоматически созданное описание

Рисунок 40 – код

Изображение выглядит как текст, снимок экрана, программное обеспечение

Автоматически созданное описание

Рисунок 41 – код

На рисунке 42 изображена форма для нахождения заявки (не запущена)

Изображение выглядит как текст, снимок экрана, программное обеспечение, дисплей

Автоматически созданное описание

Рисунок 42 – код

На рисунках 43 - 48 изображены код для данной формы

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

Рисунок 43 – код

Изображение выглядит как текст, Мультимедийное программное обеспечение, программное обеспечение, снимок экрана

Автоматически созданное описание

Рисунок 44 – код

Изображение выглядит как текст, снимок экрана, программное обеспечение

Автоматически созданное описание

Рисунок 45 – код

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

Рисунок 46 – код

Изображение выглядит как снимок экрана, текст, Мультимедийное программное обеспечение, программное обеспечение

Автоматически созданное описание

Рисунок 47 – код

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

Рисунок 48 – код

На рисунке 49 изображена форма для выбора пользователя (не запущена)

Изображение выглядит как текст, снимок экрана, дисплей, программное обеспечение

Автоматически созданное описание

Рисунок 49 – форма

На рисунках 50 - 52 изображены код для данной формы

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

Рисунок 50 – код

Изображение выглядит как текст, снимок экрана, программное обеспечение

Автоматически созданное описание

Рисунок 51 – код

Изображение выглядит как текст, Шрифт, линия, снимок экрана

Автоматически созданное описание

Рисунок 52 – код

На рисунке 53 изображена форма для указания запчастей (не запущена)

Изображение выглядит как снимок экрана, текст, программное обеспечение

Автоматически созданное описание

Рисунок 53 – форма

На рисунках 54 - 55 изображены код для данной формы

Изображение выглядит как текст, снимок экрана

Автоматически созданное описание

Рисунок 54 – код

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

Рисунок 55 – код

На рисунке 56 изображена форма для изменения статуса заявки (не запущена)

Изображение выглядит как текст, программное обеспечение, Шрифт, Значок на компьютере

Автоматически созданное описание

Рисунок 56 – форма

На рисунке 57 58 изображены код для данной формы

Изображение выглядит как текст, снимок экрана, программное обеспечение

Автоматически созданное описание

Рисунок 57 – код

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 58 – код

На рисунке 59 изображена форма для создания новой заявки (не запущена)

Изображение выглядит как текст, снимок экрана, программное обеспечение, веб-страница

Автоматически созданное описание

Рисунок 59 – форма

На рисунках 60 - 62 изображены код для данной формы

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

Рисунок 60 – код

Изображение выглядит как текст, снимок экрана, программное обеспечение

Автоматически созданное описание

Рисунок 61 – код

Изображение выглядит как текст, программное обеспечение, Мультимедийное программное обеспечение, снимок экрана

Автоматически созданное описание

Рисунок 62 – код

На рисунке 63 изображена форма пользователей (не запущена)

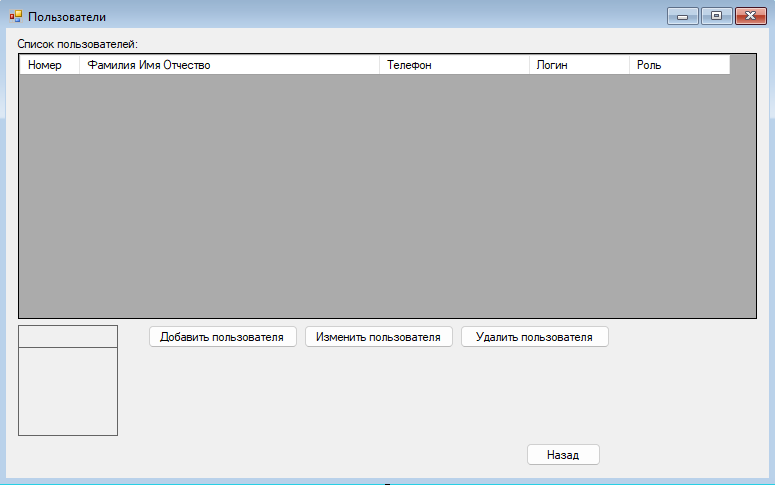


Рисунок 63 – код

На рисунках 64 - 72 изображены код для данной формы

Изображение выглядит как текст, снимок экрана, программное обеспечение, Шрифт

Автоматически созданное описание

Рисунок 64 – код

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

Рисунок 65 – код

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

Рисунок 66 – код

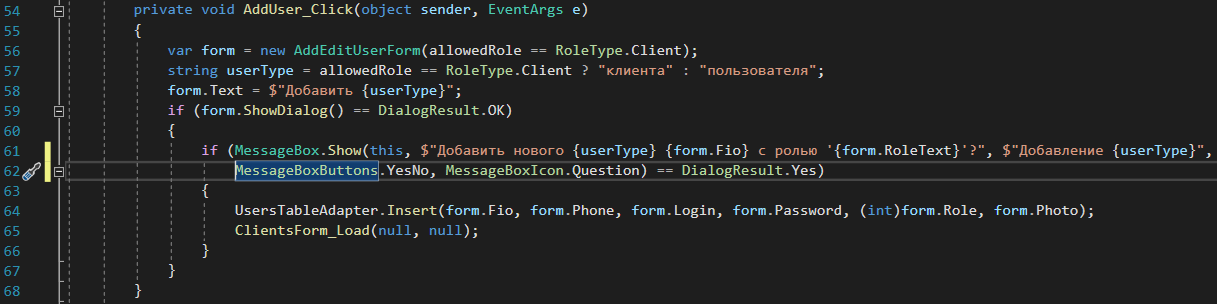


Рисунок 67 – код

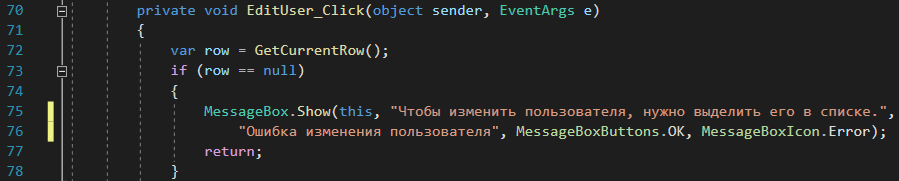


Рисунок 68 – код

Изображение выглядит как текст, программное обеспечение, Мультимедийное программное обеспечение, Графическое программное обеспечение

Автоматически созданное описание

Рисунок 69 – код

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

Рисунок 70 – код

Изображение выглядит как снимок экрана, текст, линия

Автоматически созданное описание

Рисунок 71 – код

Изображение выглядит как текст, снимок экрана, Мультимедийное программное обеспечение, программное обеспечение

Автоматически созданное описание

Рисунок 72 – код

На рисунке 73 изображена форма для редактирования пользователя (не запущена)

Изображение выглядит как текст, снимок экрана, дисплей, программное обеспечение

Автоматически созданное описание

Рисунок 73 – код

На рисунках 74 - 82 изображены код для данной формы

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

Рисунок 74 – код

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 75 – код

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

Рисунок 76 – код

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

Рисунок 77 – код

Изображение выглядит как текст, снимок экрана, программное обеспечение

Автоматически созданное описание

Рисунок 78 – код

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

Рисунок 79 – код

Изображение выглядит как снимок экрана, текст, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

Рисунок 80 – код

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

Рисунок 81 – код

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 82 – код

На рисунке 83 изображена форма техники, которая в ремонте (не запущена)

Изображение выглядит как снимок экрана, текст, программное обеспечение, компьютер

Автоматически созданное описание

Рисунок 83 – форма

На рисунке 84 изображён код для данной формы

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

Рисунок 84 – код

На рисунке 85 изображена форма для истории входа (не запущена)

Изображение выглядит как текст, снимок экрана, дисплей, программное обеспечение

Автоматически созданное описание

Рисунок 85 – форма

На рисунках 86 - 88 изображены код для данной формы

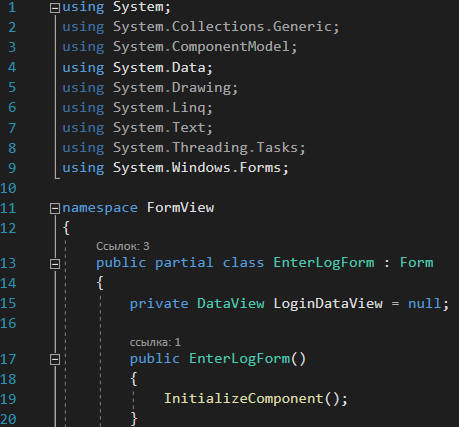


Рисунок 86 – код

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

Рисунок 87 – код

Изображение выглядит как снимок экрана, текст, Мультимедийное программное обеспечение, программное обеспечение

Автоматически созданное описание

Рисунок 88 – код

На рисунке 89 изображена форма для выбора пользователя (не запущена)

Изображение выглядит как текст, снимок экрана, программное обеспечение, Операционная система

Автоматически созданное описание

Рисунок 89 – код

На рисунках 90 - 93 изображены код для данной формы

Изображение выглядит как текст, снимок экрана, программное обеспечение

Автоматически созданное описание

Рисунок 90 – код

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

Рисунок 91 – код

Изображение выглядит как текст, программное обеспечение, Мультимедийное программное обеспечение, снимок экрана

Автоматически созданное описание

Рисунок 92 – код

Изображение выглядит как текст, Шрифт, снимок экрана

Автоматически созданное описание

Рисунок 93 – код

На рисунке 94 изображён код для типов ролей

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

Рисунок 94 – код

На рисунке 95 изображён код для класса Program

Изображение выглядит как текст, снимок экрана, дисплей, программное обеспечение

Автоматически созданное описание

Рисунок 95 – код

Текст программы:

MainForm:

using FormView.PracticeDataSetTableAdapters;

using FormView.Properties;

using PracticeTools;

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Data.SqlClient;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace FormView

{

public partial class MainForm : Form

{

private readonly LoginForm loginForm = new LoginForm();

private PracticeDataSet.USERSRow currentUser = null;

private readonly Dictionary<RoleType, List<Control>> controls;

private readonly ImageLoader photo;

private readonly DatabaseWatcher completeWatcher;

private readonly DatabaseWatcher statusWatcher;

public MainForm()

{

InitializeComponent();

photo = new ImageLoader(Photo, PhotoLoadingError);

completeWatcher = new DatabaseWatcher(

() => RequestsTableAdapter.GetDataForCompleteWatching(currentUser.userID),

(table, row) => ((PracticeDataSet.RequestsDataTable)table).Rows.Find(((PracticeDataSet.RequestsRow)row).requestID),

2000);

completeWatcher.OnDataNew += ClientWatcher\_OnRequestCompleted;

statusWatcher = new DatabaseWatcher(

() => RequestsTableAdapter.GetDataForStatusWatching(),

(table, row) => ((PracticeDataSet.RequestsDataTable)table).Rows.Find(((PracticeDataSet.RequestsRow)row).requestID),

2000);

statusWatcher.OnDataChanged += ClientWatcher\_OnRequestStatusChanged;

statusWatcher.OnDataNew += ClientWatcher\_OnNewRequest;

statusWatcher.OnDataAbsent += ClientWatcher\_OnRequestAbsent;

controls = new Dictionary<RoleType, List<Control>>()

{

{

RoleType.None,

new List<Control>()

{ }

},

{

RoleType.Manager,

new List<Control>()

{

Requests,

NewRequest,

Clients,

AllUsers,

OngoingTech,

EnterLog,

}

},

{

RoleType.Master,

new List<Control>()

{

Requests,

}

},

{

RoleType.Operator,

new List<Control>()

{

Requests,

Clients,

}

},

{

RoleType.Client,

new List<Control>()

{

Requests,

NewRequest,

}

}

};

}

private void MainForm\_Load(object sender, EventArgs e)

{

TypeTableAdapter.Fill(PracticeDataSet.TYPE);

UsersTableAdapter.Fill(PracticeDataSet.USERS);

Login();

if (currentUser == null)

{

Close();

}

}

private void Login()

{

loginForm.Users = PracticeDataSet.USERS;

if (loginForm.ShowDialog(this) == DialogResult.OK)

{

currentUser = loginForm.User;

CurrentUser.Text = currentUser.fio;

Role.Text = PracticeDataSet.TYPE.First(t => t.typeID == currentUser.type).role;

LoginLogout.Text = "Выйти";

photo.LoadImage(currentUser.IsphotoNull() ? null : currentUser.photo);

var role = (RoleType)currentUser.type;

if (role == RoleType.Client)

{

completeWatcher.StartWatching();

}

else if (role == RoleType.Operator)

{

statusWatcher.StartWatching();

}

UpdateRoleAccess((RoleType)currentUser.type);

}

}

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

{

if (currentUser != null)

{

currentUser = null;

CurrentUser.Text = "";

Role.Text = "";

photo.ResetImage();

LoginLogout.Text = "Войти";

completeWatcher.StopWatching();

UpdateRoleAccess(RoleType.None);

}

else

{

Login();

}

}

private void UpdateRoleAccess(RoleType roleType)

{

var c = controls.Values.SelectMany(l => l).Distinct().ToList();

EnableDisableControls(c, false);

c = controls[roleType];

EnableDisableControls(c, true);

}

private void EnableDisableControls(List<Control> controls, bool enable)

{

controls.ForEach(c => c.Enabled = enable);

}

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

{

var form = new RequestsForm(currentUser);

form.ShowDialog(this);

}

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

{

var form = new ClientsForm(RoleType.Client, currentUser.userID);

form.ShowDialog(this);

}

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

{

var form = new ClientsForm(RoleType.None, currentUser.userID);

form.ShowDialog(this);

}

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

{

if ((RoleType)currentUser.type != RoleType.Client && !UsersTableAdapter.GetDataByType((int)RoleType.Client).Any())

{

MessageBox.Show(this, "Невозможно добавить заявку - в системе отсутствуют клиенты. Необходимо добавить хотя бы одного клиента.",

"Ошибка добавления заявки", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

var form = new NewOrderForm(currentUser);

while (true)

{

if (form.ShowDialog(this) == DialogResult.OK)

{

if (string.IsNullOrWhiteSpace(form.Description))

if (MessageBox.Show(this, "Необходимо описать неисправность. Продолжить?", "Ошибка формирования заявки",

MessageBoxButtons.YesNo, MessageBoxIcon.Question) == DialogResult.Yes)

continue;

else

return;

if (string.IsNullOrWhiteSpace(form.TechName))

if (MessageBox.Show(this, "Необходимо указать название техники. Продолжить?", "Ошибка формирования заявки",

MessageBoxButtons.YesNo, MessageBoxIcon.Question) == DialogResult.Yes)

continue;

else

return;

var techTable = OrgTechModelTableAdapter.GetData();

var newRow = (PracticeDataSet.orgTechModelRow)techTable.NewRow();

newRow.TechTypeID = form.TechType;

newRow.Model = form.TechName;

techTable.Rows.Add(newRow);

OrgTechModelTableAdapter.Update(techTable);

RequestsTableAdapter.Insert(DateTime.Now.Date, form.Description, "Ноавя заявка", null, "", newRow.orgTechModelID, null, form.Client);

MessageBox.Show(this, $"Новая заявка успешно добавлена пользователем {PracticeDataSet.USERS.FindByuserID(form.Client).fio}",

"Заявка добавлена", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

break;

}

}

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

{

var form = new OngoingTechForm();

form.ShowDialog(this);

}

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

{

var форма = new EnterLogForm();

форма.ShowDialog(this);

}

private void ClientWatcher\_OnRequestCompleted(DatabaseWatcher sender, DataRow changedRow)

{

var requestId = ((PracticeDataSet.RequestsRow)changedRow).requestID;

var message = $"Появились новости по Вашему заказу номер {requestId}.\nЗаказ готов к выдаче.";

ShowNotification(message);

}

private void ClientWatcher\_OnRequestStatusChanged(DatabaseWatcher sender, DataRow changedRow)

{

var requestId = ((PracticeDataSet.RequestsRow)changedRow).requestID;

var requestRow = (PracticeDataSet.RequestsRow)RequestsTableAdapter.GetRequestById(requestId).Rows[0];

var masterRow = requestRow.IsmasterIDNull() ? null : (PracticeDataSet.USERSRow)PracticeDataSet.USERS.Rows.Find(requestRow.masterID);

var masterFio = masterRow == null ? "<не установлен>" : masterRow.fio;

var message = $"Появились новости по заказу номер {requestRow.requestID}.\nТекущий статус заказа: {requestRow.requestStatus}.\nМастер: {masterFio}";

if (!requestRow.IscomplectionDateNull())

message += "\nЗаказ готов к выдаче.";

ShowNotification(message);

}

private void ClientWatcher\_OnNewRequest(DatabaseWatcher sender, DataRow newRow)

{

var requestId = ((PracticeDataSet.RequestsRow)newRow).requestID;

var requestRow = (PracticeDataSet.RequestsRow)RequestsTableAdapter.GetRequestById(requestId).Rows[0];

var clientFio = PracticeDataSet.USERS.FindByuserID(requestRow.clientID).fio;

var message = $"Клиент {clientFio} добавил в систему новую заявку # {requestId}";

ShowNotification(message);

}

private void ClientWatcher\_OnRequestAbsent(DatabaseWatcher sender, DataRow newRow)

{

var requestId = ((PracticeDataSet.RequestsRow)newRow).requestID;

var requestDB = RequestsTableAdapter.GetRequestById(requestId);

if (requestDB == null || requestDB.Rows.Count == 0)

{

var message = $"Заявка # {requestId} удалена из системы.";

ShowNotification(message);

}

else

{

var requestRow = (PracticeDataSet.RequestsRow)requestDB.Rows[0];

if (!requestRow.IscomplectionDateNull())

{

var message = $"Появились новости по заказу номер {requestId}.\nЗаказ готов к выдаче.";

ShowNotification(message);

}

}

}

private void ShowNotification(string message)

{

if (InvokeRequired)

{

Invoke(new Action<string>((msg) => MessageBox.Show(this, msg, "Обновление по заказу", MessageBoxButtons.OK, MessageBoxIcon.Information)), message);

}

else

{

MessageBox.Show(this, message, "Обновление по заказу", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

}

}

}

AddEditUserForm:

using PracticeTools;

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 FormView

{

public partial class AddEditUserForm : Form

{

private readonly ImageLoader photo;

private int editedUserId = 0;

RoleType initialRole;

public AddEditUserForm(bool clientOnly)

{

InitializeComponent();

RoleComboBox.Visible = RoleLabel.Visible = !clientOnly;

photo = new ImageLoader(PhotoBox, PhotoLoadingError);

}

public int EditedUserId

{

set => editedUserId = value;

}

public string Fio

{

get => FullName.Text;

set => FullName.Text = value;

}

public string RoleText

{ get => RoleComboBox.Text; }

public RoleType Role

{

get { return (RoleType)((int?)RoleComboBox.SelectedValue ?? 0); }

set

{

if (RoleComboBox.Visible)

RoleComboBox.SelectedValue = (int)value;

else

initialRole = value;

}

}

public string Phone

{

get { return PhoneNumber.Text; }

set { PhoneNumber.Text = value; }

}

public string Login

{

get { return LoginTextBox.Text; }

set { LoginTextBox.Text = value; }

}

public string Password

{

get { return Password1.Text; }

set

{

Password1.Text = Password2.Text = value;

ShowPassword.Checked = false;

ShowPassword.Enabled = false;

}

}

public string Photo

{

get { return PhotoPath.Text; }

set

{

PhotoPath.Text = value;

RedrawPhoto();

}

}

private void AddEditUserForm\_Load(object sender, EventArgs e)

{

TypeTableAdapter.Fill(this.PracticeDataSet.TYPE);

if (!RoleComboBox.Visible)

RoleComboBox.SelectedValue = (int)RoleType.Client;

else

RoleComboBox.SelectedValue = (int)initialRole;

}

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

{

Password1.PasswordChar = Password2.PasswordChar = ShowPassword.Checked ? '\0' : '\*';

}

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

{

if (OpenFhotoDialog.ShowDialog() == DialogResult.OK)

{

PhotoPath.Text = OpenFhotoDialog.FileName;

RedrawPhoto();

}

}

private void PhotoPath\_Leave(object sender, EventArgs e)

{

RedrawPhoto();

}

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

{

DialogResult = DialogResult.OK;

if (string.IsNullOrWhiteSpace(Password1.Text))

{

if (MessageBox.Show(this, "Пароль не может быть пустым. Повторите ввод.", "Ошибка назначения пароля",

MessageBoxButtons.OKCancel, MessageBoxIcon.Error) == DialogResult.OK)

{

DialogResult = DialogResult.None;

return;

}

DialogResult = DialogResult.Cancel;

}

else if (Password1.Text != Password2.Text)

{

if (MessageBox.Show(this, "Пароли не совпадают. Повторите ввод.", "Ошибка назначения пароля",

MessageBoxButtons.OKCancel, MessageBoxIcon.Error) == DialogResult.OK)

{

DialogResult = DialogResult.None;

return;

}

DialogResult = DialogResult.Cancel;

}

else if (string.IsNullOrWhiteSpace(FullName.Text))

{

if (MessageBox.Show(this, "Ф. И. О. не может быть пустым. Повторите ввод.", "Ошибка ввода Ф. И. О.", MessageBoxButtons.OKCancel,

MessageBoxIcon.Error) == DialogResult.OK)

{

DialogResult = DialogResult.None;

return;

}

DialogResult = DialogResult.Cancel;

}

else if (string.IsNullOrWhiteSpace(LoginTextBox.Text))

{

if (MessageBox.Show(this, "Логин не может быть пустым. Повторите ввод.", "Ошибка назначения логина",

MessageBoxButtons.OKCancel, MessageBoxIcon.Error) == DialogResult.OK)

{

DialogResult = DialogResult.None;

return;

}

DialogResult = DialogResult.Cancel;

}

else if (UsersTableAdapter.GetData().Any(u => u.login.ToLower() == LoginTextBox.Text.ToLower() && u.userID != editedUserId))

{

if (MessageBox.Show(this, $"Логин {LoginTextBox.Text} уже занят. Повторите ввод.",

"Ошибка назначения логина", MessageBoxButtons.OKCancel, MessageBoxIcon.Error) == DialogResult.OK)

{

DialogResult = DialogResult.None;

return;

}

DialogResult = DialogResult.Cancel;

}

Close();

}

private void RedrawPhoto()

{

if (string.IsNullOrWhiteSpace(PhotoPath.Text))

photo.ResetImage();

else

photo.LoadImage(PhotoPath.Text);

}

}

}

ClientsForm:

using PracticeTools;

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Data.SqlClient;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace FormView

{

public partial class ClientsForm : Form

{

private readonly int currentUser;

private readonly RoleType allowedRole;

private readonly ImageLoader photo;

public ClientsForm(RoleType allowedRole, int currentUser)

{

InitializeComponent();

photo = new ImageLoader(Photo, PhotoLoadingError);

this.allowedRole = allowedRole;

this.currentUser = currentUser;

RoleColumn.Visible = allowedRole == RoleType.None;

}

private void ClientsForm\_Load(object sender, EventArgs e)

{

if (allowedRole == RoleType.None)

UsersTableAdapter.FillExt(PracticeDataSet.USERS);

else

UsersTableAdapter.FillExtByType(PracticeDataSet.USERS, (int)allowedRole);

}

private PracticeDataSet.USERSRow GetCurrentRow()

{

var curRowView = (DataRowView)UsersBindingSource.Current;

return (PracticeDataSet.USERSRow)curRowView?.Row;

}

private void UserGridView\_SelectionChanged(object sender, EventArgs e)

{

var row = GetCurrentRow();

if (row == null)

photo.ResetImage();

else

photo.LoadImage(row.IsphotoNull() ? null : row.photo);

}

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

{

var form = new AddEditUserForm(allowedRole == RoleType.Client);

string userType = allowedRole == RoleType.Client ? "клиента" : "пользователя";

form.Text = $"Добавить {userType}";

if (form.ShowDialog() == DialogResult.OK)

{

if (MessageBox.Show(this, $"Добавить нового {userType} {form.Fio} с ролью '{form.RoleText}'?", $"Добавление {userType}",

MessageBoxButtons.YesNo, MessageBoxIcon.Question) == DialogResult.Yes)

{

UsersTableAdapter.Insert(form.Fio, form.Phone, form.Login, form.Password, (int)form.Role, form.Photo);

ClientsForm\_Load(null, null);

}

}

}

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

{

var row = GetCurrentRow();

if (row == null)

{

MessageBox.Show(this, "Чтобы изменить пользователя, нужно выделить его в списке.",

"Ошибка изменения пользователя", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

string userType = allowedRole == RoleType.Client ? "клиента" : "пользователя";

var form = new AddEditUserForm(allowedRole == RoleType.Client)

{

Text = $"Изменить {userType}",

Fio = row.fio,

Role = (RoleType)row.type,

Phone = row.phone,

Login = row.login,

Password = row.password,

EditedUserId = row.userID,

};

if (!row.IsphotoNull())

form.Photo = row.photo;

if (form.ShowDialog() == DialogResult.OK)

{

if (MessageBox.Show(this, $"Изменить свойства {userType} {form.Fio}?", $"Изменение {userType}",

MessageBoxButtons.YesNo, MessageBoxIcon.Question) == DialogResult.Yes)

{

row.fio = form.Fio;

row.phone = form.Phone;

row.login = form.Login;

row.password = form.Password;

if (form.Photo.Length == 0)

row.SetphotoNull();

else

row.photo = form.Photo;

row.type = (int)form.Role;

row["role"] = TypeTableAdapter.GetData().FindBytypeID((int)form.Role).role;

UsersTableAdapter.Update(row);

UserGridView.Refresh();

UserGridView\_SelectionChanged(null, null);

}

}

}

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

{

var row = GetCurrentRow();

if (row == null)

{

MessageBox.Show(this, "Чтобы удалить пользователя, нужно выделить его в списке.",

"Ошибка удаления пользователя", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

if (row.userID == currentUser)

{

MessageBox.Show(this, "Нельзя удалить текущего пользователя.", "Ошибка удаления пользователя", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

if (MessageBox.Show(this, $"Вы действительно хотите безвозвратно удалить пользователя {row.fio}?",

"Удаление пользователя", MessageBoxButtons.YesNo, MessageBoxIcon.Question) != DialogResult.Yes)

{

return;

}

try

{

var table = UsersTableAdapter.GetData();

var deletingRow = table.First(x => x.userID == row.userID);

deletingRow.Delete();

UsersTableAdapter.Update(table);

PracticeDataSet.USERS.RemoveUSERSRow(row);

UserGridView.Refresh();

}

catch(SqlException)

{

MessageBox.Show(this, $"Невозможно удалить пользователя {row.fio}, так как он фигурирует в заказах.",

"Ошибка удаления пользователя", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

}

}

EnterLogForm:

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 FormView

{

public partial class EnterLogForm : Form

{

private DataView LoginDataView = null;

public EnterLogForm()

{

InitializeComponent();

}

private void EnterLogForm\_Load(object sender, EventArgs e)

{

EnterLogTableAdapter.FillWithUserName(PracticeDataSet.EnterLog);

LoginDataView = new DataView(PracticeDataSet.EnterLog);

EnterLogGridView.DataSource = LoginDataView;

}

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

{

if (LoginDataView.RowFilter.Length == 0)

{

var form = new FilterLoginForm();

if (form.ShowDialog(this) == DialogResult.OK)

{

LoginDataView.RowFilter = "UserLogin " + form.Filter;

Filter.Text = "Убрать фильтр";

if (LoginDataView.Count == 0)

{

MessageBox.Show(this, "Поиск по заданным критериям не дал результатов. Попробуйте изменить критерии поиска.",

"Пустой результат поиска", MessageBoxButtons.OK, MessageBoxIcon.Warning);

}

}

}

else

{

LoginDataView.RowFilter = string.Empty;

Filter.Text = "Отфильтровать по логину";

}

}

}

}

FilterLoginForm:

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 FormView

{

public partial class FilterLoginForm : Form

{

public FilterLoginForm()

{

InitializeComponent();

UserSelect.Checked = true;

}

public string Filter

{

get

{

if (UserSelect.Checked)

{

return $" = '{(string)UserComboBox.SelectedValue}'";

}

else

{

return $" LIKE '\*{LoginTextBox.Text}\*'";

}

}

}

private void SelectUserForm\_Load(object sender, EventArgs e)

{

UsersTableAdapter.Fill(PracticeDataSet.USERS);

}

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

{

if (LoginSelect.Checked && LoginTextBox.Text.Length == 0)

{

var yesno = MessageBox.Show(this, "Нужно указать искомый логин или его фрагмент, или выбрать зарегистрированного пользователя из списка. Продолжить?",

"Ошибка ввода", MessageBoxButtons.YesNo, MessageBoxIcon.Exclamation);

if (yesno == DialogResult.Yes)

{

return;

}

DialogResult = DialogResult.Cancel;

}

else

{

DialogResult = DialogResult.OK;

}

Close();

}

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

{

bool user = sender == UserSelect;

UserComboBox.Enabled = user;

LoginTextBox.Enabled = !user;

}

}

}

FindRequestForm:

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 FormView

{

public partial class FindRequestForm : Form

{

private readonly RoleType filteredRole;

private DateTime parsedStartDate;

public FindRequestForm(bool filterByMaster)

{

InitializeComponent();

filteredRole = filterByMaster ? RoleType.Master : RoleType.Client;

if (filteredRole == RoleType.Master)

{

FilterByUser.Text = "Мастер:";

}

}

public string GetFilter()

{

List<string> where = new List<string>();

int id;

if (int.TryParse(RequestID.Text, out id))

where.Add($"requestID = {id}");

if (CompletedOnlyCheckBox.Checked)

where.Add("complectionDate IS NOT NULL");

if (FilterByUser.Checked)

{

var rowView = (DataRowView)UserComboBox.SelectedValue;

var selectedRow = (PracticeDataSet.USERSRow)rowView.Row;

var columnName = filteredRole == RoleType.Master ? "masterID" : "clientID";

where.Add($"{columnName} = {selectedRow.userID}");

}

if (Description.Text.Length > 0)

where.Add($"LowerDescription LIKE '\*{Description.Text.ToLower()}\*'");

if (Status.Text.Length > 0)

where.Add($"LowerStatus LIKE '\*{Status.Text.ToLower()}\*'");

if (StartDate.Text.Length > 0)

where.Add($"startDate = '{parsedStartDate}'");

string res = "";

if (where.Count > 0)

{

res = where[0];

for (int i = 1; i < where.Count; i++)

{

res += $" AND {where[i]}";

}

}

return res;

}

public string StatusFilter

{

get { return Status.Text; }

}

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

{

if (StartDate.Text.Length > 0 && !DateTime.TryParse(StartDate.Text, out parsedStartDate))

{

MessageBox.Show(this, $"Данные, введенные в поле '{StartDateLabel.Text}', не могут быть распознаны как дата.\nПовторите ввод.",

"Ошибка ввода", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

DialogResult = DialogResult.OK;

Close();

}

private void FindRequestForm\_Load(object sender, EventArgs e)

{

if (filteredRole == RoleType.None)

{

UsersTableAdapter.Fill(PracticeDataSet.USERS);

}

else

{

UsersTableAdapter.FillByType(PracticeDataSet.USERS, (int)filteredRole);

}

}

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

{

UserComboBox.Enabled = FilterByUser.Checked;

}

}

}

InputTextForm:

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 FormView

{

public partial class InputTextForm : Form

{

public InputTextForm()

{

InitializeComponent();

}

public string Caption

{

set { Text = value; }

}

public string Label

{

set { LabelControl.Text = value; }

}

public string InputText

{

set { TextControl.Text = value; }

get { return TextControl.Text; }

}

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

{

DialogResult = DialogResult.OK;

Close();

}

}

}

Login:

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 FormView

{

public partial class LoginForm : Form

{

private Random random = new Random();

private const int captchaLength = 6;

private int tryCount = 0;

private PracticeDataSet.USERSRow loggedIn = null;

private PracticeDataSet.USERSDataTable users = null;

public LoginForm()

{

InitializeComponent();

ReEnableOk.Interval = 3 \* 60 \* 1000;

}

public PracticeDataSet.USERSDataTable Users

{

set { users = value; }

}

public PracticeDataSet.USERSRow User

{

get { return loggedIn; }

}

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

{

if (!CheckInput())

{

InitializeCapcha();

if (tryCount++ > 1)

{

Ok.Enabled = false;

if(tryCount == 3)

{

ReEnableOk.Start();

}

}

}

}

private bool CheckInput()

{

var expectedUser = users.FirstOrDefault(u => u.login == Login.Text);

bool result;

if ((!Capcha.Visible || CheckCapcha()) && CheckPassword())

{

DialogResult = DialogResult.OK;

tryCount = 0;

Close();

result = true;

}

else

{

MessageBox.Show(this, tryCount >= 3 ? "Вход запрещен" : "Повторите вход", "Ошибка входа", MessageBoxButtons.OK, MessageBoxIcon.Error);

result = false;

}

EnterLogTableAdapter.InsertEnter(expectedUser?.userID, Login.Text, result);

return result;

}

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

{

Password.PasswordChar = ShowPassword.Checked ? '\0' : '\*';

}

private void InitializeCapcha()

{

Capcha.Visible = CapchaLabel.Visible = CapchaText.Visible = RegenerateCapcha.Visible = true;

CapchaText.Text = string.Empty;

var text = GenerateRandomText();

Capcha.Tag = text;

var bitmap = new Bitmap(Capcha.Width, Capcha.Height);

using (Graphics g = Graphics.FromImage(bitmap))

{

g.Clear(Color.White);

using (Font font = new Font("Arial", 24, FontStyle.Bold))

{

var dx = Capcha.Width / captchaLength;

int x = 0;

foreach (char c in text)

{

g.DrawString($"{c}", font, Brushes.Black, x, random.Next(0, Capcha.Height / 2));

x += dx;

}

}

Pen pen = new Pen(Color.LightGray, 3);

for (int i = 0; i < 5; i++)

{

g.DrawLine(pen, random.Next(0, Capcha.Width), random.Next(0, Capcha.Height), random.Next(0, Capcha.Width), random.Next(0, Capcha.Height));

}

}

Capcha.Image = bitmap;

}

private string GenerateRandomText()

{

const string chars = "ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789";

char[] text = new char[captchaLength];

for (int i = 0; i < captchaLength; i++)

{

text[i] = chars[random.Next(chars.Length)];

}

return new string(text);

}

private bool CheckCapcha()

{

if (!Capcha.Visible)

{

return true;

}

return (string)Capcha.Tag == CapchaText.Text.ToUpper();

}

private bool CheckPassword()

{

loggedIn = users == null ? null : users.FirstOrDefault(u => u.login == Login.Text && u.password == Password.Text);

return loggedIn != null;

}

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

{

InitializeCapcha();

}

private void ReEnableOk\_Tick(object sender, EventArgs e)

{

ReEnableOk.Stop();

Ok.Enabled = true;

}

private void LoginForm\_Load(object sender, EventArgs e)

{

if (tryCount == 0)

{

Capcha.Visible = CapchaLabel.Visible = CapchaText.Visible = RegenerateCapcha.Visible = false;

}

else

{

InitializeCapcha();

}

Login.Text = "";

Password.Text = "";

CapchaText.Text = "";

ShowPassword.Checked = false;

}

}

}

NewOrderForm:

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 FormView

{

public partial class NewOrderForm : Form

{

private readonly int currentClientId;

public NewOrderForm(PracticeDataSet.USERSRow currentUser)

{

InitializeComponent();

currentClientId = (RoleType)currentUser.type == RoleType.Client ? currentUser.userID : 0;

ClientList.Enabled = currentClientId == 0;

}

public string Description

{ get => ProblemDescr.Text; }

public int TechType

{ get => (int)ModelType.SelectedValue; }

public string TechName

{ get => ModelName.Text; }

public int Client

{ get => (int)ClientList.SelectedValue; }

private void NewOrderForm\_Load(object sender, EventArgs e)

{

UsersTableAdapter.FillByType(PracticeDataSet.USERS, (int)RoleType.Client);

if (currentClientId != 0)

{

ClientList.SelectedValue = currentClientId;

}

TechTypeTableAdapter.Fill(PracticeDataSet.TechType);

}

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

{

if (Description.Length == 0)

{

MessageBox.Show(this, "Необходимо ввести описание проблемы.", "Ошибка ввода", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

if (TechName.Length == 0)

{

MessageBox.Show(this, "Необходимо указать модель устройства.", "Ошибка ввода", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

DialogResult = DialogResult.OK;

}

}

}

OngoingTechForm:

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 FormView

{

public partial class OngoingTechForm : Form

{

public OngoingTechForm()

{

InitializeComponent();

}

private void OngoingTechForm\_Load(object sender, EventArgs e)

{

OrgTechModelTableAdapter.FillOngoing(PracticeDataSet.orgTechModel);

}

}

}

Program:

using System;

using System.Collections.Generic;

using System.Data.SqlClient;

using System.Linq;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace FormView

{

internal static class Program

{

/// <summary>

/// The main entry point for the application.

/// </summary>

[STAThread]

static void Main()

{

Application.EnableVisualStyles();

Application.SetCompatibleTextRenderingDefault(false);

Application.Run(new MainForm());

}

}

}

RequestsForm:

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Reflection;

using System.Runtime.CompilerServices;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace FormView

{

public partial class RequestsForm : Form

{

private readonly PracticeDataSet.USERSRow CurrentUser;

private readonly bool showExecutableTime;

private DataView requestDataView;

public RequestsForm(PracticeDataSet.USERSRow currentUser)

{

CurrentUser = currentUser;

InitializeComponent();

var role = (RoleType)currentUser.type;

ChangeDescription.Enabled = role == RoleType.Client || role == RoleType.Manager;

SetMaster.Enabled = role == RoleType.Operator || role == RoleType.Manager;

FindRequest.Enabled = role != RoleType.Client;

Parts.Enabled = role == RoleType.Manager || role == RoleType.Master;

showExecutableTime = role != RoleType.Client;

ExecutionTimeColumn.Visible = AverageTime.Visible = AverageTimeLabel.Visible = showExecutableTime;

Comment.ReadOnly = role != RoleType.Manager && role != RoleType.Master;

ChangeStatus.Enabled = role == RoleType.Manager || role == RoleType.Master;

DeleteRequest.Enabled = role == RoleType.Manager;

PracticeDataSet.Requests.Columns.Add("LowerDescription", typeof(string));

PracticeDataSet.Requests.Columns.Add("LowerStatus", typeof(string));

UserName.Text = currentUser.fio;

}

private void RequestsForm\_Load(object sender, EventArgs e)

{

RequestGrid.DataSource = null;

CommentsTableAdapter.Fill(PracticeDataSet.Comments);

switch ((RoleType)CurrentUser.type)

{

case RoleType.Client:

ClientColumn.Visible = false;

RequestsTableAdapter.FillByClient(PracticeDataSet.Requests, CurrentUser.userID);

break;

case RoleType.Master:

MasterColumn.Visible = false;

RequestsTableAdapter.FillByMaster(PracticeDataSet.Requests, CurrentUser.userID);

break;

default:

RequestsTableAdapter.FillExt(PracticeDataSet.Requests);

break;

}

foreach(var rawData in PracticeDataSet.Requests)

{

rawData["LowerDescription"] = rawData.problemDescryption.ToLower();

rawData["LowerStatus"] = rawData.requestStatus.ToLower();

}

requestDataView = new DataView(PracticeDataSet.Requests);

RequestGrid.DataSource = requestDataView;

RefreshView();

RoleLabel.Text = TypeTableAdapter.GetData().FindBytypeID(CurrentUser.type).role;

}

private void RefreshView()

{

if (showExecutableTime)

{

double summDays = 0;

int completedCount = 0;

foreach (DataGridViewRow row in RequestGrid.Rows)

{

var duration = row.Cells["ExecutionTimeColumn"].Value.ToString();

if (duration.Length > 0)

{

summDays += int.Parse(duration);

completedCount++;

}

}

AverageTime.Text = completedCount > 0

? (summDays / completedCount).ToString("F0")

: "<нет>";

}

Comment.Enabled = requestDataView.Count > 0;

TotalRequest.Text = RequestsTableAdapter.GetTotalCount().ToString();

RequestCount.Text = requestDataView.Count.ToString();

RequestGrid.Refresh();

}

private PracticeDataSet.RequestsRow GetCurrentRow()

{

var sel = RequestGrid.CurrentRow;

if (sel == null)

return null;

var index = sel.Index;

return (PracticeDataSet.RequestsRow)requestDataView[index].Row;

}

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

{

var row = GetCurrentRow();

if(row == null)

return;

var inputText = new InputTextForm();

inputText.Caption = "Изменить описание проблемы";

inputText.Label = "Описание проблемы:";

inputText.InputText = row.problemDescryption;

if (inputText.ShowDialog() == DialogResult.OK)

{

row.problemDescryption = inputText.InputText;

row["LowerDescription"] = inputText.InputText.ToLower();

RequestsTableAdapter.Update(row);

RefreshView();

}

}

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

{

if (string.IsNullOrEmpty(requestDataView.RowFilter))

{

var findForm = new FindRequestForm((RoleType)CurrentUser.type == RoleType.Client);

if (findForm.ShowDialog() != DialogResult.OK)

return;

requestDataView.RowFilter = findForm.GetFilter();

FindRequest.Text = "Сбросить фильтр";

if (requestDataView.Count == 0)

{

MessageBox.Show(this, "Поиск по заданным критериям не дал результатов. Попробуйте изменить критерии поиска.",

"Пустой результат поиска", MessageBoxButtons.OK, MessageBoxIcon.Warning);

}

}

else

{

requestDataView.RowFilter = "";

FindRequest.Text = "Найти заявку";

}

RefreshView();

}

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

{

var row = GetCurrentRow();

if (row == null)

return;

if (row.IscomplectionDateNull())

{

var form = new SelectUserForm(RoleType.Master, row.IsmasterIDNull() ? 0 : row.masterID);

if (form.ShowDialog() == DialogResult.OK)

{

row.masterID = form.SelectedUserID;

row["MasterFio"] = UsersTableAdapter.GetData().FindByuserID(form.SelectedUserID).fio;

RequestsTableAdapter.Update(row);

RefreshView();

}

}

else

{

MessageBox.Show(this, "Заявка уже закрыта, назначить мастера нельзя.", "Ошибка назначения мастера", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

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

{

var row = GetCurrentRow();

if (row == null)

return;

var inputText = new InputTextForm();

inputText.Caption = "Требуемые запчасти";

inputText.Label = "Укажите сведения о требуемых запчастях:";

inputText.InputText = row.repairParts;

if (inputText.ShowDialog() == DialogResult.OK)

{

row.repairParts = inputText.InputText;

RequestsTableAdapter.Update(row);

RefreshView();

}

}

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

{

var row = GetCurrentRow();

if (row == null)

return;

if (!row.IscomplectionDateNull())

{

MessageBox.Show(this, "Изменить статус завершенной заявки нельзя", "Ошибка изменения статуса", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

else

{

var form = new StatusForm();

form.Status = row.requestStatus;

if (form.ShowDialog() == DialogResult.OK)

{

row.requestStatus = form.Status;

row["LowerStatus"] = form.Status.ToLower();

if (form.IsFinished)

{

row.complectionDate = DateTime.Now.Date;

row["Duration"] = (row.complectionDate - row.startDate).TotalDays.ToString();

}

RequestsTableAdapter.Update(row);

RefreshView();

}

}

}

private void RequestGrid\_SelectionChanged(object sender, EventArgs e)

{

var row = GetCurrentRow();

if (row == null)

{

Comment.Text = "";

}

else

{

var comment = PracticeDataSet.Comments.FirstOrDefault(c => c.requestID == row.requestID);

if (comment == null)

Comment.Text = "";

else

Comment.Text = comment.message;

}

}

private void Comment\_Leave(object sender, EventArgs e)

{

var requestRow = GetCurrentRow();

var commentRow = PracticeDataSet.Comments.FirstOrDefault(c => c.requestID == requestRow.requestID);

if (commentRow == null)

{

if (Comment.Text.Length > 0)

{

CommentsTableAdapter.Insert(Comment.Text, requestRow.requestID);

CommentsTableAdapter.Fill(PracticeDataSet.Comments);

}

}

else

{

commentRow.message = Comment.Text;

CommentsTableAdapter.Update(commentRow);

}

}

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

{

var requestRow = GetCurrentRow();

if (requestRow == null)

{

MessageBox.Show(this, "Необходимо выделить в списке заявку, которую следует удалить.",

"Ошибка удаления заявки", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

if (MessageBox.Show(this, $"Вы действительно хотите удалить заявку #{requestRow.requestID}, заведенную клентом '{requestRow["ClientFio"]}'" +

$" {requestRow.startDate.ToString("dd-MM-yyyy")}? Восстановить заявку после удаления будет невозможно!",

"Удалить заявку", MessageBoxButtons.YesNo, MessageBoxIcon.Question) != DialogResult.Yes)

{

return;

}

RequestsTableAdapter.DeleteById(requestRow.requestID);

PracticeDataSet.Requests.RemoveRequestsRow(requestRow);

RefreshView();

}

}

}

RoleType:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace FormView

{

public enum RoleType

{

None = 0,

Manager = 1,

Master = 2,

Operator = 3,

Client = 4,

}

}

SelectUserForm:

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 FormView

{

public partial class SelectUserForm : Form

{

private readonly RoleType roleType;

private readonly int initialSelectedUser;

public SelectUserForm(RoleType roleType, int initialSelectedUser)

{

InitializeComponent();

this.roleType = roleType;

this.initialSelectedUser = initialSelectedUser;

}

public int SelectedUserID

{

get { return (int?)UserComboBox.SelectedValue ?? 0; }

}

private void SelectUserForm\_Load(object sender, EventArgs e)

{

if (roleType == RoleType.None)

{

UsersTableAdapter.Fill(PracticeDataSet.USERS);

}

else

{

UsersTableAdapter.FillByType(PracticeDataSet.USERS, (int)roleType);

}

if (initialSelectedUser != 0)

{

UserComboBox.SelectedValue = initialSelectedUser;

}

}

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

{

DialogResult = DialogResult.OK;

Close();

}

}

}

StatusForm:

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 FormView

{

public partial class StatusForm : Form

{

public StatusForm()

{

InitializeComponent();

}

public string Status

{

get { return StatusTextBox.Text; }

set { StatusTextBox.Text = value; }

}

public bool IsFinished

{

get { return FinishRequest.Checked; }

}

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

{

DialogResult = DialogResult.OK;

Close();

}

}

}