ПРИЛОЖЕНИЕ А

Текст программы

АННОТАЦИЯ

Настоящие приложение направлено на описание текста программы информационной системы выполнения анализа и устранения аномальных ситуаций с прайсом товара.

Содержание

[ВВЕДЕНИЕ 3](#_Toc42453830)

[1. ПЕРЕЧЕНЬ МОДУЛЕЙ 4](#_Toc42453831)

[2. ТЕКСТ ПРОГРАММЫ 5](#_Toc42453832)

[2.1 Текст скрипта базы данных 5](#_Toc42453833)

[2.2 . Текст модулей программы 13](#_Toc42453834)

[2.2.1 Текст «AboutBox.cs» 13](#_Toc42453835)

[2.2.2 Текст «ChangePasswordForm.cs» 14](#_Toc42453836)

[2.2.3 Текст «ConectionForm.cs» 15](#_Toc42453837)

[2.2.4 Текст «DataBase\_Configuration.cs» 17](#_Toc42453838)

[2.2.5 Текст «DataBaseConnection.cs» 18](#_Toc42453839)

[2.2.6 Текст «DoljForm.cs» 18](#_Toc42453840)

[2.2.7 Текст «FeedBackForm.cs» 20](#_Toc42453841)

[2.2.8 Текст «GroupForm.cs» 21](#_Toc42453842)

[2.2.9 Текст «MainForm.cs» 23](#_Toc42453843)

[2.2.10 Текст «PdfClass.cs» 34](#_Toc42453844)

[2.2.11 Текст «Program.cs» 34](#_Toc42453845)

[2.2.12 Текст «SignInForm.cs» 35](#_Toc42453846)

[2.2.13 Текст «SignUpForm.cs» 35](#_Toc42453847)

[2.2.14 Текст «TipRabotForm.cs» 39](#_Toc42453848)

[2.2.15 Текст «TTForm.cs» 40](#_Toc42453849)

[2.2.16 Текст «Vip\_no\_RabotForm.cs» 41](#_Toc42453850)

[2.2.17 Текст «ZayavkaForm.cs» 44](#_Toc42453851)

[2.2.18 Текст «ZastavkaForm.cs» 46](#_Toc42453852)

# ВВЕДЕНИЕ

Разрабатываемое программное обеспечение предназначено для использования сотрудниками техничего отдела ООО «Compo»

Программный продукт реализован используя Microsoft Visual Studio 2017 на объектно-ориентированном языке программирования C#. Также использовались дополнительные бесплатные библиотеки «Microsoft.Office.Interop.Excel.dll» и «Microsoft.Office.Interop.Word.dll»

# ПЕРЕЧЕНЬ МОДУЛЕЙ

Таблица 1 – Модули информацтонной системы программного продукта

| Название модуля | Описание модуля |
| --- | --- |
| 1 | 2 |
| AboutBox.cs | Форма сведений о программе |
| ChangePasswordForm.cs | Форма смены пароля сотрудника |
| ConnectionForm.cs | Форма подключения к серверу базы данных |
| DataBase\_Configuration.cs | Класс создания папки в реестре |
| DataBaseConnection.cs | Создание строки подлючения |
| FeedBackForm.cs | Форма обратной связи |
| GroupForm.cs | Форма создания, изменения, удаления группы, назначения, изменения снятия сотрудника с группы |
| DoljForm.cs | Форма добавления, изменения, удаления должности |
| MainForm.cs | Основная форма приложения |
| PdfClass.cs | Класс экспорта данных в Pdf-файл |
| Program.cs | Основной класс программы |
| SignInForm.cs | Форма авторизации |
| SignUpForm | Форма добавления, изменения, удаления сотрудников |
| StatusForm.cs | Форма добавления, изменения, удаления статусов |
| TipRabot.cs | Форма добавление, изменение, удаление типа работ |
| TTForm.cs | Форма добавление, изменения, удаления торговых точек |
| Vip\_noRabotForm.cs | Форма добавление, изменения, удаления типа выполненных работ |
| ZayavkaForm.cs | Форма добавлния, удаления, изменения заявки |
| ZastavkaForm.cs | Форма проверки подключения к серверу базы данных |

# ТЕКСТ ПРОГРАММЫ

## Текст скрипта базы данных

set ansi\_padding on

go

set quoted\_identifier on

go

set ansi\_nulls on

go

create database [Compo]

go

use [Compo]

go

------------------------------\

--Областьб объявления таблиц--|

------------------------------/

Create table [dbo].[Dolj]

(

[ID\_Dolj] [int] not null identity(1,1) Primary key,

[Dolj][varchar](20) not null,

[Oklad] [int] not null,

[Dolj\_Logical\_Delete] [int] not null,

constraint [UQ\_Dolj\_Profile] unique ([Dolj]),

constraint [CH\_Dolj\_Cyrill\_Upper] check ([Dolj] like ('%[А-Я]%')),

constraint [CH\_Dolj\_Cyrill\_Lower] check ([Dolj] like ('%[а-я]%')),

constraint [CH\_Dolj\_Logical\_Delete] check ([Dolj\_Logical\_Delete] in (0,1)),

)

go

Create table [dbo].[Sotr]

(

[ID\_Sotr] [int] not null identity(1,1) Primary key,

[Fam\_Sotr] [varchar](25) not null,

[Name\_Sotr] [varchar](20) not null,

[Otch\_Sotr] [varchar](25) null default ('Нет данных'),

[Ser\_Pas] [int] not null,

[Num\_Pas] [int] not null,

[Date\_Birth] [varchar](10) not null,

[Dolj\_ID] [int] not null,

[Login\_Sotr] [varchar] (16) not null,

[Password\_Sotr] [varchar] (40) not null,

[Sotr\_Logical\_Delete] [int] not null,

constraint [FK\_Dolj] foreign key ([Dolj\_ID])

references [Dolj] ([ID\_Dolj]),

constraint [UQ\_Passport] unique ([Ser\_Pas],[Num\_Pas]),

constraint [UQ\_Login\_Sotr] unique ([Login\_Sotr]),

constraint [CH\_Sotr\_Name\_Cyrill\_Upper] check ([Name\_Sotr] like ('%[А-Я]%')),

constraint [CH\_Sotr\_Name\_Cyrill\_Lower] check ([Name\_Sotr] like ('%[а-я]%')),

constraint [CH\_Sotr\_Fam\_Cyrill\_Upper] check ([Fam\_Sotr] like ('%[А-Я]%')),

constraint [CH\_Sotr\_Fam\_Cyrill\_Lower] check ([Fam\_Sotr] like ('%[а-я]%')),

constraint [CH\_Sotr\_Otch\_Cyrill\_Upper] check ([Otch\_Sotr] like ('%[А-Я]%')),

constraint [CH\_Sotr\_Otch\_Cyrill\_Lower] check ([Otch\_Sotr] like ('%[а-я]%')),

constraint [CH\_Password\_Sotr\_Latin\_Upper] check ([Password\_Sotr] like ('%[A-Z]%')),

constraint [CH\_Password\_Sotr\_Latin\_Lower] check ([Password\_Sotr] like ('%[a-z]%')),

constraint [CH\_Password\_Sotr\_Spec\_Numbers] check ([Password\_Sotr] like ('%[0-9]%')),

constraint [CH\_Num\_Pas\_Legth] check (len([Num\_Pas])=6),

constraint [CH\_Ser\_Pas\_Legth] check (len([Ser\_Pas])=4),

constraint [CH\_Num\_Pas\_Spec\_Numbers] check ([Num\_Pas] like ('%[0-9]%')),

constraint [CH\_Ser\_Pas\_Spec\_Numbers] check ([Ser\_Pas] like ('%[0-9]%')),

constraint [CH\_Login\_Sotr\_Latin\_Upper] check ([Login\_Sotr] not in ('%[А-Я]%')),

constraint [CH\_Login\_Sotr\_Latin\_Lower] check ([Login\_Sotr] not in ('%[а-я]%')),

constraint [CH\_Sotr\_Logical\_Delete] check ([Sotr\_Logical\_Delete] in (0,1))

)

go

create table [dbo].[TT]

(

[ID\_TT] [int] not null identity(1,1) Primary key,

[TT\_Name] [varchar](32) not null,

[Kol\_vo\_Ust\_nix\_Polok] [int] not null,

[Adres][varchar](255) not null,

[TT\_Logical\_Delete] [int] not null,

constraint [CH\_TT\_Logical\_Delete] check ([TT\_Logical\_Delete] in (0,1))

)

go

create table [dbo].[Group]

(

[ID\_Group][int] not null identity(1,1) Primary key,

[Date] [varchar](10) not null,

[Group\_Logical\_Delete] [int] not null,

constraint [CH\_Group\_Logical\_Delete] check ([Group\_Logical\_Delete] in (0,1))

)

go

create table [dbo].[Sostav\_Group]

(

[ID] [int] not null identity(1,1) Primary key,

[Group\_ID] [int] not null,

[Sotr\_ID] [int] not null,

constraint [FK\_Sostav\_Group] foreign key ([Group\_ID])

references [Group] ([ID\_Group]),

constraint [FK\_Sotr] foreign key ([Sotr\_ID])

references [Sotr] ([ID\_Sotr]),

)

go

create table [dbo].[Status]

(

[ID\_Status] [int] not null identity(1,1) Primary key,

[Status\_Name] [varchar](16) not null,

[Status\_Logical\_Delete] [int] not null,

constraint [CH\_Status\_Cyrill\_Upper] check ([Status\_Name] like ('%[А-Я]%')),

constraint [CH\_Status\_Cyrill\_Lower] check ([Status\_Name] like ('%[а-я]%')),

constraint [CH\_Status\_Logical\_Delete] check ([Status\_Logical\_Delete] in (0,1))

)

go

create table[dbo].[Zayavka]

(

[ID\_Zayavka][int] not null identity(1,1) Primary key,

[Group\_ID] [int] not null,

[TT\_ID] [int] not null,

[Status\_ID][int] not null,

[Data\_Podachi] [varchar](16)not null,

[Data\_Nachala\_Rabot] [varchar](16),

[Data\_Okonchania\_Rabot] [varchar](16),

[Data\_Zakritia\_Zayavki] [varchar](16),

[Spisok\_Rapot][varchar](255) not null,

[Spisok\_Materialov][varchar](255) not null,

[Kommentariy\_TT][varchar](255),

[Kommentariy\_Group][varchar](255),

[Zayavka\_Logical\_Delete] [int] not null,

constraint [FK\_TT] foreign key ([TT\_ID])

references [TT] ([ID\_TT]),

constraint [FK\_Status] foreign key ([Status\_ID])

references [Status] ([ID\_Status]),

constraint [FK\_Group] foreign key ([Group\_ID])

references [Group] ([ID\_Group]),

constraint [CH\_Zayavka\_Logical\_Delete] check ([Zayavka\_Logical\_Delete] in (0,1))

)

go

create table [dbo].[Tip\_Rabot]

(

[ID\_Tip\_Rabot] [int] not null identity(1,1) Primary key,

[Name\_Tip\_Rabot] [varchar](63) not null,

[Tip\_Rabot\_Logical\_Delete][int] not null,

constraint [CH\_Name\_Tip\_Rabot\_Cyrill\_Upper] check ([Name\_Tip\_Rabot] like ('%[А-Я]%')),

constraint [CH\_Name\_Tip\_Rabot\_Cyrill\_Lower] check ([Name\_Tip\_Rabot] like ('%[а-я]%')),

constraint [CH\_Tip\_Rabot\_Logical\_Delete] check ([Tip\_Rabot\_Logical\_Delete] in (0,1))

)

go

create table [dbo].[Vip\_no\_Rabot]

(

[ID\_Vip\_no\_Rabot][int] not null identity(1,1) Primary key,

[Zayavka\_ID][int]not null,

[Tip\_Rabot\_ID][int]not null,

[Vip\_no\_Rabot\_Logical\_Delete][int] not null,

constraint [FK\_Zayavka] foreign key ([Zayavka\_ID])

references [Zayavka] ([ID\_Zayavka]),

constraint [FK\_Tip\_Rabot] foreign key ([Tip\_Rabot\_ID])

references [Tip\_Rabot] ([ID\_Tip\_Rabot]),

constraint [CH\_Vip\_no\_Rabot\_Logical\_Delete] check ([Vip\_no\_Rabot\_Logical\_Delete] in (0,1))

)

go

create view [dbo].[View\_Sotr] ("ID","Фамилия","Имя","Отчество","Серия паспорта","Номер паспорта","Дата рождения","Должность","Оклад","Sotr\_Logical\_Delete")

as

SELECT

dbo.Sotr.ID\_Sotr,

dbo.Sotr.Fam\_Sotr,

dbo.Sotr.Name\_Sotr,

dbo.Sotr.Otch\_Sotr,

dbo.Sotr.Ser\_Pas,

dbo.Sotr.Num\_Pas,

dbo.Sotr.Date\_Birth,

dbo.Dolj.Dolj,

dbo.Dolj.Oklad,

dbo.Sotr.Sotr\_Logical\_Delete

FROM dbo.Dolj INNER JOIN

dbo.Sotr ON dbo.Dolj.ID\_Dolj = dbo.Sotr.Dolj\_ID

go

create view [dbo].[View\_Group]("ID","№ Группы","ФИО","Дата")

as

SELECT

dbo.[Sostav\_Group].ID,

dbo.[Group].ID\_Group,

dbo.Sotr.Fam\_Sotr+' '+dbo.Sotr.Name\_Sotr+' '+dbo.Sotr.Otch\_Sotr,

dbo.[Group].Date

FROM dbo.[Group] INNER JOIN

dbo.Sostav\_Group ON dbo.[Group].ID\_Group = dbo.Sostav\_Group.Group\_ID INNER JOIN

dbo.Sotr ON dbo.Sostav\_Group.Sotr\_ID = dbo.Sotr.ID\_Sotr

go

create view [dbo].[View\_Zayavka]

(

"ID",

"№ Группы",

"ТТ",

"Адрес ТТ",

"Статус",

"Дата подачи",

"Дата начала работ",

"Дата окончания работ",

"Дата закрытия заявки",

"Список работ",

"Список материалов",

"Комментарий торговой точки",

"Комментарий группы",

"Zayavka\_Logical\_Delete"

)

as

SELECT

dbo.Zayavka.ID\_Zayavka,

dbo.Zayavka.Group\_ID,

dbo.TT.TT\_Name,

dbo.TT.Adres,

dbo.Status.Status\_Name,

dbo.Zayavka.Data\_Podachi,

dbo.Zayavka.Data\_Nachala\_Rabot,

dbo.Zayavka.Data\_Okonchania\_Rabot,

dbo.Zayavka.Data\_Zakritia\_Zayavki,

dbo.Zayavka.Spisok\_Rapot,

dbo.Zayavka.Spisok\_Materialov,

dbo.Zayavka.Kommentariy\_TT,

dbo.Zayavka.Kommentariy\_Group,

dbo.Zayavka.Zayavka\_Logical\_Delete

FROM dbo.Zayavka INNER JOIN

dbo.[Group] ON dbo.Zayavka.Group\_ID = dbo.[Group].ID\_Group INNER JOIN

dbo.TT ON dbo.Zayavka.TT\_ID = dbo.TT.ID\_TT INNER JOIN

dbo.Status ON dbo.Zayavka.Status\_ID = dbo.Status.ID\_Status

go

create view [dbo].[View\_Vip\_no\_Rabot]

(

"ID",

"Тип работы",

"ТТ",

"Vip\_no\_Rabot\_Logical\_Delete"

)

as

SELECT

dbo.Vip\_no\_Rabot.ID\_Vip\_no\_Rabot,

dbo.Tip\_Rabot.Name\_Tip\_Rabot,

dbo.TT.TT\_Name,

dbo.Vip\_no\_Rabot.Vip\_no\_Rabot\_Logical\_Delete

FROM dbo.Vip\_no\_Rabot INNER JOIN

dbo.Zayavka ON dbo.Vip\_no\_Rabot.Zayavka\_ID = dbo.Zayavka.ID\_Zayavka INNER JOIN

dbo.Tip\_Rabot ON dbo.Vip\_no\_Rabot.Tip\_Rabot\_ID = dbo.Tip\_Rabot.ID\_Tip\_Rabot INNER JOIN

dbo.TT ON dbo.Zayavka.TT\_ID = dbo.TT.ID\_TT

go

----------------------------------------\

--Область объявления хранимых процедур--|

----------------------------------------/

create procedure [dbo].[Dolj\_Insert]

@Dolj [varchar](20),

@Oklad [int]

as

insert into [dbo].[Dolj]([Dolj],[Oklad],[Dolj\_Logical\_Delete])

values

(

@Dolj,

@Oklad,

0

)

go

create procedure [dbo].[Dolj\_Update]

@ID\_Dolj [int],

@Dolj [varchar](20),

@Oklad [int]

as

update [dbo].[Dolj] set

[Dolj]=@Dolj,

[Oklad]=@Oklad

where

[ID\_Dolj]=@ID\_Dolj

go

Create procedure [dbo].[Dolj\_Delete]

@ID\_Dolj [int]

as

delete from [dbo].[Dolj]

where [ID\_Dolj]=@ID\_Dolj

go

Create procedure [dbo].[Dolj\_Logical\_Delete]

@ID\_Dolj [int]

as

update [dbo].[Dolj] set

Dolj\_Logical\_Delete = 1

where

ID\_Dolj=@ID\_Dolj

go

Create procedure [dbo].[Sotr\_Insert]

@Fam\_Sotr [varchar](25),

@Name\_Sotr [varchar](20),

@Otch\_Sotr [varchar](25),

@Ser\_Pas [int],

@Num\_Pas [int],

@Date\_Birth [varchar](10),

@Dolj\_ID [int],

@Login\_Sotr [varchar](16),

@Password\_Sotr [varchar](40)

as

insert into [dbo].[Sotr]

(

[Fam\_Sotr],

[Name\_Sotr],

[Otch\_Sotr],

[Ser\_Pas],

[Num\_Pas],

[Date\_Birth],

[Dolj\_ID],

[Login\_Sotr],

[Password\_Sotr],

[Sotr\_Logical\_Delete]

)

values

(

@Fam\_Sotr,

@Name\_Sotr,

@Otch\_Sotr,

@Ser\_Pas ,

@Num\_Pas ,

@Date\_Birth,

@Dolj\_ID ,

@Login\_Sotr ,

@Password\_Sotr,

0

)

go

create procedure [dbo].[Sotr\_Update]

@ID\_Sotr [int],

@Fam\_Sotr [varchar](25),

@Name\_Sotr [varchar](20),

@Otch\_Sotr [varchar](25),

@Ser\_Pas [int],

@Num\_Pas [int],

@Date\_Birth [varchar](10),

@Dolj\_ID [int]

as

update [dbo].[Sotr] set

[Fam\_Sotr] = @Fam\_Sotr,

[Name\_Sotr]= @Name\_Sotr,

[Otch\_Sotr]= @Otch\_Sotr,

[Ser\_Pas] = @Ser\_Pas,

[Num\_Pas] = @Num\_Pas,

[Date\_Birth] = @Date\_Birth,

[Dolj\_ID] = @Dolj\_ID

where

ID\_Sotr = @ID\_Sotr

go

Create procedure [dbo].[Sotr\_Delete]

@ID\_Sotr [int]

as

delete from [dbo].[Sotr]

where [ID\_Sotr]=@ID\_Sotr

go

Create procedure [dbo].[Sotr\_Logical\_Delete]

@ID\_Sotr [int]

as

update [dbo].[Sotr] set

[Sotr\_Logical\_Delete]=1

where

[ID\_Sotr]= @ID\_Sotr

go

create procedure [dbo].[Sotr\_Password\_Update]

@Login\_Sotr [varchar](16),

@Password\_Sotr [varchar](40)

as

update [dbo].[Sotr] set

[Password\_Sotr] = @Password\_Sotr

where [Login\_Sotr]=@Login\_Sotr

go

Create procedure [dbo].[TT\_Insert]

@TT\_Name [varchar](32),

@Kol\_vo\_Ust\_nix\_Polok [int],

@Adres[varchar](255),

@TT\_Logical\_Delete [int]

as

insert into [dbo].[TT]

(

[TT\_Name],

[Kol\_vo\_Ust\_nix\_Polok],

[Adres],

[TT\_Logical\_Delete]

)

values

(

@TT\_Name ,

@Kol\_vo\_Ust\_nix\_Polok,

@Adres,

0

)

go

create procedure [dbo].[TT\_Update]

@ID\_TT [int],

@TT\_Name [varchar](32),

@Kol\_vo\_Ust\_nix\_Polok [int],

@Adres[varchar](255)

as

update [dbo].[TT] set

[TT\_Name]=@TT\_Name,

[Kol\_vo\_Ust\_nix\_Polok]=@Kol\_vo\_Ust\_nix\_Polok,

[Adres]=@Adres

where

ID\_TT = @ID\_TT

go

Create procedure [dbo].[TT\_Delete]

@ID\_TT [int]

as

delete from [dbo].[TT]

where [ID\_TT]=@ID\_TT

go

Create procedure [dbo].[TT\_Logical\_Delete]

@ID\_TT [int]

as

update [dbo].[TT] set

[TT\_Logical\_Delete]=1

where

[ID\_TT]= @ID\_TT

go

Create procedure [dbo].[Group\_Insert]

@Date [varchar](10),

@Group\_Logical\_Delete [int]

as

insert into [dbo].[Group]

(

[Date],

[Group\_Logical\_Delete]

)

values

(

@Date,

0

)

go

create procedure [dbo].[Group\_Update]

@ID\_Group[int],

@Date [varchar](10)

as

update [dbo].[Group] set

[Date]=@Date

where

ID\_Group = @ID\_Group

go

Create procedure [dbo].[Group\_Delete]

@ID\_Group [int]

as

delete from [dbo].[Group]

where [ID\_Group]=@ID\_Group

go

Create procedure [dbo].[Group\_Logical\_Delete]

@ID\_Group [int]

as

update [dbo].[Group] set

[Group\_Logical\_Delete]=1

where

[ID\_Group]= @ID\_Group

go

Create procedure [dbo].[Sostav\_Group\_Insert]

@Group\_ID[int],

@Sotr\_ID [int]

as

insert into [dbo].[Sostav\_Group]

(

[Group\_ID],

[Sotr\_ID]

)

values

(

@Group\_ID,

@Sotr\_ID

)

go

create procedure [dbo].[Sostav\_Group\_Update]

@ID[int],

@Group\_ID[int],

@Sotr\_ID [int]

as

update [dbo].[Sostav\_Group] set

[Group\_ID]=@Group\_ID,

[Sotr\_ID]=@Sotr\_ID

where

[ID]= @ID

go

Create procedure [dbo].[Sostav\_Group\_Delete]

@ID [int]

as

delete from [dbo].[Sostav\_Group]

where [ID]=@ID

go

Create procedure [dbo].[Status\_Insert]

@Status\_Name[varchar](16),

@Status\_Logical\_Delete [int]

as

insert into [dbo].[Status]

(

[Status\_Name],

[Status\_Logical\_Delete]

)

values

(

@Status\_Name,

0

)

go

create procedure [dbo].[Status\_Update]

@ID\_Status[int],

@Status\_Name[varchar](16)

as

update [dbo].[Status] set

[Status\_Name]=@Status\_Name

where

ID\_Status = @ID\_Status

go

Create procedure [dbo].[Status\_Delete]

@ID\_Status [int]

as

delete from [dbo].[Status]

where [ID\_Status]=@ID\_Status

go

Create procedure [dbo].[Status\_Logical\_Delete]

@ID\_Status [int]

as

update [dbo].[Status] set

[Status\_Logical\_Delete]=1

where

[ID\_Status]= @ID\_Status

go

Create procedure [dbo].[Zayavka\_Insert]

@Group\_ID [int],

@TT\_ID [int],

@Status\_ID[int],

@Data\_Podachi [varchar](16),

@Data\_Nachala\_Rabot [varchar](16),

@Data\_Okonchania\_Rabot [varchar](16),

@Data\_Zakritia\_Zayavki [varchar](16),

@Spisok\_Rapot[varchar](255),

@Spisok\_Materialov[varchar](255),

@Kommentariy\_TT[varchar](255),

@Kommentariy\_Group[varchar](255),

@Zayavka\_Logical\_Delete [int]

as

insert into [dbo].[Zayavka]

(

[Group\_ID],

[TT\_ID],

[Status\_ID],

[Data\_Podachi],

[Data\_Nachala\_Rabot],

[Data\_Okonchania\_Rabot],

[Data\_Zakritia\_Zayavki],

[Spisok\_Rapot],

[Spisok\_Materialov],

[Kommentariy\_TT],

[Kommentariy\_Group],

[Zayavka\_Logical\_Delete]

)

values

(

@Group\_ID,

@TT\_ID,

@Status\_ID,

@Data\_Podachi ,

@Data\_Nachala\_Rabot ,

@Data\_Okonchania\_Rabot,

@Data\_Zakritia\_Zayavki,

@Spisok\_Rapot,

@Spisok\_Materialov,

@Kommentariy\_TT,

@Kommentariy\_Group,

0

)

go

create procedure [dbo].[Zayavka\_Update]

@ID\_Zayavka [int],

@Group\_ID [int],

@TT\_ID [int],

@Status\_ID[int],

@Data\_Podachi [varchar],

@Data\_Nachala\_Rabot [varchar],

@Data\_Okonchania\_Rabot [varchar],

@Data\_Zakritia\_Zayavki [varchar],

@Spisok\_Rapot[varchar](255),

@Spisok\_Materialov[varchar](255),

@Kommentariy\_TT[varchar](255),

@Kommentariy\_Group[varchar](255),

@Zayavka\_Logical\_Delete [int]

as

update [dbo].[Zayavka] set

[Group\_ID]=@Group\_ID,

[TT\_ID]=@TT\_ID,

[Status\_ID]=@Status\_ID,

[Data\_Podachi]=@Data\_Podachi,

[Data\_Nachala\_Rabot]=@Data\_Nachala\_Rabot,

[Data\_Okonchania\_Rabot]=Data\_Okonchania\_Rabot,

[Data\_Zakritia\_Zayavki]=Data\_Zakritia\_Zayavki,

[Spisok\_Rapot]=@Spisok\_Rapot,

[Spisok\_Materialov]=@Spisok\_Materialov,

[Kommentariy\_TT]=@Kommentariy\_TT,

[Kommentariy\_Group]=@Kommentariy\_Group

where

ID\_Zayavka = @ID\_Zayavka

go

Create procedure [dbo].[Zayavka\_Delete]

@ID\_Zayavka [int]

as

delete from [dbo].[Zayavka]

where [ID\_Zayavka]=@ID\_Zayavka

go

Create procedure [dbo].[Zayavka\_Logical\_Delete]

@ID\_Zayavka [int]

as

update [dbo].[Zayavka] set

[Zayavka\_Logical\_Delete]=1

where

[ID\_Zayavka]= @ID\_Zayavka

go

Create procedure [dbo].[Tip\_Rabot\_Insert]

@Name\_Tip\_Rabot[varchar](16),

@Tip\_Rabot\_Logical\_Delete [int]

as

insert into [dbo].[Tip\_Rabot]

(

[Name\_Tip\_Rabot],

[Tip\_Rabot\_Logical\_Delete]

)

values

(

@Name\_Tip\_Rabot,

0

)

go

create procedure [dbo].[Tip\_Rabot\_Update]

@ID\_Tip\_Rabot[int],

@Name\_Tip\_Rabot[varchar](16)

as

update [dbo].[Tip\_Rabot] set

[Name\_Tip\_Rabot]=@Name\_Tip\_Rabot

where

ID\_Tip\_Rabot = @ID\_Tip\_Rabot

go

Create procedure [dbo].[Tip\_Rabot\_Delete]

@ID\_Tip\_Rabot [int]

as

delete from [dbo].[Tip\_Rabot]

where [ID\_Tip\_Rabot]=@ID\_Tip\_Rabot

go

Create procedure [dbo].[Tip\_Rabot\_Logical\_Delete]

@ID\_Tip\_Rabot [int]

as

update [dbo].[Tip\_Rabot] set

[Tip\_Rabot\_Logical\_Delete]=1

where

[ID\_Tip\_Rabot]= @ID\_Tip\_Rabot

go

Create procedure [dbo].[Vip\_no\_Rabot\_Insert]

@Zayavka\_ID [int],

@Tip\_Rabot\_ID[int],

@Vip\_no\_Rabot\_Logical\_Delete [int]

as

insert into [dbo].[Vip\_no\_Rabot]

(

[Zayavka\_ID],

[Tip\_Rabot\_ID],

[Vip\_no\_Rabot\_Logical\_Delete]

)

values

(

@Zayavka\_ID,

@Tip\_Rabot\_ID,

0

)

go

create procedure [dbo].[Vip\_no\_Rabot\_Update]

@ID\_Vip\_no\_Rabot[int],

@Zayavka\_ID [int],

@Tip\_Rabot\_ID[int]

as

update [dbo].[Vip\_no\_Rabot] set

[Zayavka\_ID]=@Zayavka\_ID,

[Tip\_Rabot\_ID]=@Tip\_Rabot\_ID

where

[ID\_Vip\_no\_Rabot] = @ID\_Vip\_no\_Rabot

go

Create procedure [dbo].[Vip\_no\_Rabot\_Delete]

@ID\_Vip\_no\_Rabot [int]

as

delete from [dbo].[Vip\_no\_Rabot]

where [ID\_Vip\_no\_Rabot]=@ID\_Vip\_no\_Rabot

go

Create procedure [dbo].[Vip\_no\_Rabot\_Logical\_Delete]

@ID\_Vip\_no\_Rabot [int]

as

update [dbo].[Vip\_no\_Rabot] set

[Vip\_no\_Rabot\_Logical\_Delete]=1

where

[ID\_Vip\_no\_Rabot]= @ID\_Vip\_no\_Rabot

go

insert into [dbo].[Dolj] ([Dolj],[Oklad],[Dolj\_Logical\_Delete])

values

('Администратор',90000,0),

('Старший инженер',60000,0),

('Инженер',50000,0)

go

insert into [dbo].[Sotr] ([Fam\_Sotr],[Name\_Sotr],[Otch\_Sotr],[Ser\_Pas],[Num\_Pas],[Date\_Birth],[Dolj\_ID],[Login\_Sotr],[Password\_Sotr],[Sotr\_Logical\_Delete])

values

('Иванов','Андрей','Дмитриевич',4514,551866,'19.05.2000',1,'Root','PMMc0kYUmuxoB5JB5x6Y9g==',0),

('Антонов','Евгений','Владимирович',4514,545456,'23.07.2000',3,'@nton0v','PMMc0kYUmuxoB5JB5x6Y9g==',0),

('Козленков','Михаил','Андреевич',4519,435556,'31.10.1999',3,'Koz1enk0v','PMMc0kYUmuxoB5JB5x6Y9g==',0),

('Литвина','Екатерина','Игоревна',4385,565656,'20.04.1985',2,'Letv1n@','PMMc0kYUmuxoB5JB5x6Y9g==',0),

('Бофт','Александр','Нет данных',4180,434553,'03.02.1980',2,'@lexB0ft','PMMc0kYUmuxoB5JB5x6Y9g==',0),

('Смирнова','Ксения','Нет данных',4514,556876,'21.11.2000',3,'Ksen1@','PMMc0kYUmuxoB5JB5x6Y9g==',0),

('Иванов','Алексей','Петрович',4514,545458,'13.11.2000',3,'Iv@nov','PMMc0kYUmuxoB5JB5x6Y9g==',0),

('Иванов','Алексей','Петрович',4514,785458,'25.02.2000',3,'Iv@n0v','PMMc0kYUmuxoB5JB5x6Y9g==',0)

go

insert into [dbo].[TT]([TT\_Name],[Kol\_vo\_Ust\_nix\_Polok], [Adres], [TT\_Logical\_Delete])

values

('L212 Аэропорт Домодедово',9,'аэропорт Домодедово имени М.В. Ломоносова, 1',0),

('L003 Аэропорт Домодедово2',12,'аэропорт Домодедово имени М.В. Ломоносова, 1',0),

('L153 ТЦ Бум',9,'ул. Перерва, 43, корп. 1, Москва',0),

('L131 ТРЦ Афимолл',12,'Пресненская наб., 2, Москва',0),

('L135 ТЦ Европейский',9,'площадь Киевского Вокзала, 2, Москва',0),

('L145 ТРЦ Фестиваль',12,'ул. Мичуринский Проспект, Олимпийская Деревня, 3, корп. 1, Москва',0),

('MV78 ТЦ Европейский 2',42,'площадь Киевского Вокзала, 2, Москва',0),

('L446 ТРЦ Вегас Каширский',9,'МКАД, 24-й километр, 1, Москва',0),

('L515 Мега Белая Дача 5',36,'1-й Покровский пр., 5, Котельники',0),

('L504 Ашан Белая Дача',33,'1-й Покровский пр., 5, Котельники',0),

('L157 ТЦ Домодедовский',27,'Ореховый бул., 14, корп. 3, Москва',0)

go

insert into [dbo].[Group] ([Date], [Group\_Logical\_Delete])

values

('20.05.2020',0),

('20.05.2020',0)

go

insert into [dbo].[Status] ([Status\_Name], [Status\_Logical\_Delete])

values

('Создана',0),

('Обсуждена',0),

('Утверждена',0),

('Выполняется',0),

('Выполнена',0),

('Возвращена',0),

('Закрыта',0)

go

insert into [dbo].[Sostav\_Group]([Group\_ID], [Sotr\_ID])

values

(1,2),

(1,3),

(1,4),

(2,5),

(2,6),

(2,7)

go

insert into [dbo].[Tip\_Rabot] ([Name\_Tip\_Rabot],[Tip\_Rabot\_Logical\_Delete])

values

('Замена полки',0),

('Замена донгла',0),

('Замена кабеля питания',0),

('Обновление прошивки',0),

('Замена крепежа',0),

('Замена батареек',0),

('Сбор информации о прейскуранте',0),

('Линковка ценников',0),

('Подсчет ценников',0)

go

insert into [dbo].[Zayavka] ([Group\_ID],[TT\_ID],[Status\_ID],[Data\_Podachi],[Data\_Nachala\_Rabot],[Data\_Okonchania\_Rabot],[Data\_Zakritia\_Zayavki],[Spisok\_Rapot],[Spisok\_Materialov],[Kommentariy\_Group],[Kommentariy\_TT],[Zayavka\_Logical\_Delete])

values

(2,7,4,'19.05.2020 22:45', '20.05.2020 09:00', ' . . : ', ' . . : ', 'Не работает полка', 'Новая полка и ТСД', 'ТСД на точке', '',0),

(1,10,5,'20.05.2020 23:00', '21.05.2020 09:00', '27.05.2020 18:00', ' . . : ', 'Неработает полка', 'Новая полка',' ','',0),

(2,8,1,' . . : ', '. . : ', '. . : ','. . : ','','','','',0)

go

insert into [dbo].[Vip\_no\_Rabot] ([Zayavka\_ID],[Tip\_Rabot\_ID],[Vip\_no\_Rabot\_Logical\_Delete])

values

(1,1,0),

(1,8,0)

go

## . Текст модулей программы

### Текст «AboutBox.cs»

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Drawing;

using System.Linq;

using System.Reflection;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Compo

{

partial class AboutBox : Form

{

public AboutBox()

{

InitializeComponent();

Icon iconForm = new Icon(Application.StartupPath + "\\img\\compo2.ico");

Icon = iconForm;

this.Text = String.Format("О программе {0}", AssemblyTitle);

this.labelProductName.Text = AssemblyProduct;

this.labelVersion.Text = String.Format("Версия {0}", AssemblyVersion);

this.labelCopyright.Text = AssemblyCopyright;

this.labelZayavkaName.Text = AssemblyZayavka;

this.textBoxDescription.Text = AssemblyDescription;

}

#region Acessório de Atributos do Assembly

public string AssemblyTitle

{

get

{

object[] attributes = Assembly.GetExecutingAssembly().GetCustomAttributes(typeof(AssemblyTitleAttribute), false);

if (attributes.Length > 0)

{

AssemblyTitleAttribute titleAttribute = (AssemblyTitleAttribute)attributes[0];

if (titleAttribute.Title != "")

{

return titleAttribute.Title;

}

}

return System.IO.Path.GetFileNameWithoutExtension(Assembly.GetExecutingAssembly().CodeBase);

}

}

public string AssemblyVersion

{

get

{

return Assembly.GetExecutingAssembly().GetName().Version.ToString();

}

}

public string AssemblyDescription

{

get

{

object[] attributes = Assembly.GetExecutingAssembly().GetCustomAttributes(typeof(AssemblyDescriptionAttribute), false);

if (attributes.Length == 0)

{

return "";

}

return ((AssemblyDescriptionAttribute)attributes[0]).Description;

}

}

public string AssemblyProduct

{

get

{

object[] attributes = Assembly.GetExecutingAssembly().GetCustomAttributes(typeof(AssemblyProductAttribute), false);

if (attributes.Length == 0)

{

return "";

}

return ((AssemblyProductAttribute)attributes[0]).Product;

}

}

public string AssemblyCopyright

{

get

{

object[] attributes = Assembly.GetExecutingAssembly().GetCustomAttributes(typeof(AssemblyCopyrightAttribute), false);

if (attributes.Length == 0)

{

return "";

}

return ((AssemblyCopyrightAttribute)attributes[0]).Copyright;

}

}

public string AssemblyZayavka

{

get

{

object[] attributes = Assembly.GetExecutingAssembly().GetCustomAttributes(typeof(AssemblyCompanyAttribute), false);

if (attributes.Length == 0)

{

return "";

}

return ((AssemblyCompanyAttribute)attributes[0]).Company;

}

}

#endregion

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

{

textBoxDescription.Text = "== Compo ==" +

"\r\n\r\nЭта программа представляет собой информационную систему для предприятия \"Compo\"" +

"\r\n\r\nСоздано Андреем Ивановым. (C)2020";

}

private void button1\_Click(object sender, EventArgs e)

{

FeedBackForm feedback = new FeedBackForm();

feedback.ShowDialog();

}

}

}

### Текст «ChangePasswordForm.cs»

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Data.SqlClient;

using System.Drawing;

using System.Linq;

using System.Security.Cryptography;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Compo

{

public partial class ChangePasswordForm : Form

{

DatabaseConnection database = new DatabaseConnection();

public ChangePasswordForm()

{

InitializeComponent();

Icon iconForm = new Icon(Application.StartupPath + "\\img\\compo2.ico");

Icon = iconForm;

lblNewPassword.Text = "Введите новый пароль";

lblRepNewPassword.Text = "Повторите новый пароль";

btnChangePassword.Text = "Сменить пароль";

tbNewPassword.PasswordChar = '\*';

tbRepNewPassword.PasswordChar = '\*';

}

private string GetHash(string input) //Хэширование пароля в БД

{

var md5 = MD5.Create();

var hash = md5.ComputeHash(Encoding.UTF8.GetBytes(input));

return Convert.ToBase64String(hash);

}

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

{

database.DatabaseSQL().Open();

try

{

if (tbNewPassword.Text == tbRepNewPassword.Text)

{

string passwordHash = GetHash(tbNewPassword.Text);

{

var sqlCommand = new SqlCommand("Sotr\_Password\_Update", database.DatabaseSQL());

sqlCommand.CommandType = CommandType.StoredProcedure;

sqlCommand.Parameters.AddWithValue("@Password\_Sotr", passwordHash);

sqlCommand.Parameters.AddWithValue("@Login\_Sotr", Program.Autoriz\_Sotr);

sqlCommand.ExecuteNonQuery();

}

MessageBox.Show("Пароль изменен");

this.Hide();

}

else

{

MessageBox.Show("Пароли не совпадают");

}

}

catch (Exception ex)

{

MessageBox.Show("Парооль не изменен " + ex.Message);

}

database.DatabaseSQL().Close();

}

}

}

### Текст «ConectionForm.cs»

using System;

using System.Data;

using System.Drawing;

using System.Threading;

using System.Windows.Forms;

namespace Compo

{

public partial class ConectionForm : Form

{

private DataBase\_Configuration configuration = new DataBase\_Configuration();

private Int32 status;

public ConectionForm()

{

this.TopMost = true;

InitializeComponent();

Icon iconForm = new Icon(Application.StartupPath + "\\img\\compo2.ico");

Icon = iconForm;

}

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

{

switch (DataBase\_Configuration.logCon)

{

case (true):

cbDataSource.Text = DatabaseConnection.DS;

cbInitialCatalog.Text = DatabaseConnection.IC;

cbDataSource.Enabled = true;

cbInitialCatalog.Enabled = true;

tbUserID.Enabled = true;

tbPassword.Enabled = true;

btStatus.Enabled = true;

break;

case (false):

status = 1;

Thread threadMessage = new Thread(tsslMessage);

threadMessage.Start();

configuration.dtServers += dtservers;

Thread thread = new Thread(configuration.Servers\_get);

thread.Start();

break;

}

}

private void dtservers(DataTable table)

{

Action action = () =>

{

foreach (DataRow r in table.Rows)

{

cbDataSource.Items.Add(r[0] + @"\" + r[1]);

}

status = 0;

cbDataSource.Enabled = true;

tbUserID.Enabled = true;

tbPassword.Enabled = true;

btStatus.Enabled = true;

};

Invoke(action);

}

private void tsslMessage()

{

tsslStatus.Visible = true;

switch (status)

{

case (1):

tsslStatus.Text = "Поиск серверов";

break;

case (2):

tsslStatus.Text = "Поиск источников данных";

break;

}

do

{

Thread.Sleep(300);

switch (status)

{

case (1):

if (tsslStatus.Text == "Поиск серверов...")

tsslStatus.Text = "Поиск серверов";

else tsslStatus.Text = tsslStatus.Text + ".";

break;

case (2):

if (tsslStatus.Text == "Поиск источников данных...")

tsslStatus.Text = "Поиск источников данных";

else tsslStatus.Text = tsslStatus.Text + ".";

break;

}

} while (status != 0);

tsslStatus.Text = "-";

tsslStatus.Visible = false;

Thread.CurrentThread.Abort();

}

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

{

configuration.cds = cbDataSource.Text;

configuration.cui = tbUserID.Text;

configuration.cpw = tbPassword.Text;

configuration.dtDatabases += databases;

status = 2;

Thread threadMessage = new Thread(tsslMessage);

Thread thread = new Thread(configuration.Databases\_get);

threadMessage.Start();

thread.Start();

}

private void databases(DataTable table)

{

Action action = () =>

{

foreach (DataRow r in table.Rows)

{

cbInitialCatalog.Items.Add(r[0]);

}

status = 0;

cbInitialCatalog.Enabled = true;

btConect.Enabled = true;

};

Invoke(action);

}

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

{

DatabaseConnection registry = new DatabaseConnection();

registry.Registry\_Set(cbDataSource.Text,

cbInitialCatalog.Text, tbUserID.Text, tbPassword.Text);

DataBase\_Configuration.logCon = true;

Zastavka zastavka = new Zastavka();

zastavka.Show();

this.Close();

}

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

{

Close();

}

private void ConectionForm\_FormClosing(object sender, FormClosingEventArgs e)

{

switch (DataBase\_Configuration.logCon)

{

case (true):

e.Cancel = false;

break;

case (false):

switch (MessageBox.Show("Подключение не было уствновлено! " +

"\n Завершить работу приложения?", "Химчистка",

MessageBoxButtons.YesNo, MessageBoxIcon.Warning))

{

case DialogResult.Yes:

e.Cancel = false;

Application.Exit();

break;

case DialogResult.No:

e.Cancel = true;

break;

}

break;

}

}

}

}

### Текст «DataBase\_Configuration.cs»

using System;

using System.Data;

using System.Data.SqlClient;

using System.Data.Sql;

namespace Compo

{

class DataBase\_Configuration

{

public event Action<DataTable> dtServers;

public event Action<DataTable> dtDatabases;

public event Action<bool> conState;

private DatabaseConnection registry = new DatabaseConnection();

public string cds, cui, cpw;

public static bool logCon;

public void Servers\_get()

{

SqlDataSourceEnumerator sourceEnumerator

= SqlDataSourceEnumerator.Instance;

dtServers(sourceEnumerator.GetDataSources());

}

public void Databases\_get()

{

SqlConnection sql = new SqlConnection("Data Source = " + cds +

"; Initial Catalog = master; Persist Security Info = true; " +

" User ID = " + cui + "; Password = \"" + cpw + "\"");

try

{

SqlCommand command = new SqlCommand("select name from sys.databases " +

"where name not in ('master','tempdb','model','msdb')", sql);

DataTable table = new DataTable();

sql.Open();

table.Load(command.ExecuteReader());

dtDatabases(table);

}

catch (SqlException ex)

{

DatabaseConnection.error\_message += "\n" + DateTime.Now.ToLongDateString()

+ ex.Message;

}

finally

{

sql.Close();

}

}

public void Connection\_Check()

{

try

{

registry.Registry\_Get();

DatabaseConnection.sqlConnection.Open();

conState(true);

logCon = true;

}

catch (Exception ex)

{

DatabaseConnection.error\_message += "\n" + DateTime.Now.ToLongDateString()

+ ex.Message;

conState(false);

logCon = false;

}

finally

{

DatabaseConnection.sqlConnection.Close();

}

}

}

}

### Текст «DataBaseConnection.cs»

using System.Data;

using System;

using System.Data.SqlClient;

using System.Windows.Forms;

using Microsoft.Win32;

public class DatabaseConnection

{

public static string DS = "Empty", IC = "Empty", UI = "Empty", PW = "Empty";

public static string error\_message = "App:start:" + DateTime.Now.ToLongDateString();

public static SqlConnection sqlConnection = new SqlConnection();

public void Registry\_Get()

{

RegistryKey registry = Registry.CurrentUser;

RegistryKey key = registry.CreateSubKey("Compo");

try

{

DS = key.GetValue("DS").ToString();

IC = key.GetValue("IC").ToString();

UI = key.GetValue("UI").ToString();

PW = key.GetValue("PW").ToString();

}

catch

{

key.SetValue("DS", "Empty");

key.SetValue("IC", "Empty");

key.SetValue("UI", "Empty");

key.SetValue("PW", "Empty");

}

finally

{

sqlConnection.ConnectionString = "Data Source = " + DS +

"; Initial Catalog = " + IC + "; Persist Security Info = true; " +

"User ID = " + UI + "; Password = \"" + PW + "\"";

}

}

public void Registry\_Set(string ds, string ic, string ui, string pw)

{

RegistryKey registry = Registry.CurrentUser;

RegistryKey key = registry.CreateSubKey("Compo");

try

{

key.SetValue("DS", ds);

key.SetValue("IC", ic);

key.SetValue("UI", ui);

key.SetValue("PW", pw);

Registry\_Get();

}

catch (Exception ex)

{

error\_message += "\n" + DateTime.Now.ToLongDateString() + ex.Message;

}

}

public SqlConnection DatabaseSQL()

{

return sqlConnection;

}

public DataSet TableFill(string query, SqlConnection sql)

{

SqlDataAdapter adapter = new SqlDataAdapter(query, sql);

var dataSet = new DataSet();

dataSet.Clear();

adapter.Fill(dataSet);

return dataSet;

}

}

### Текст «DoljForm.cs»

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 Compo

{

public partial class DoljForm : Form

{

DatabaseConnection database = new DatabaseConnection();

public DoljForm()

{

this.TopMost = true; //Вызов окна поверх других

InitializeComponent();

this.StartPosition = FormStartPosition.CenterScreen; //Расположение окна по центру монитора

this.FormBorderStyle = FormBorderStyle.FixedToolWindow;//Добавление иконки приложения

Icon iconForm = new Icon(Application.StartupPath + "\\img\\compo2.ico");

Icon = iconForm;

lblName.Text = "Название должности";

lblOklad.Text = "Оклад";

}

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

{

database.DatabaseSQL().Open();

try

{

SqlCommand sqlCommand = new SqlCommand("Dolj\_Insert", database.DatabaseSQL());

sqlCommand.CommandType = CommandType.StoredProcedure;

sqlCommand.Parameters.AddWithValue("@Dolj", tbName.Text);

sqlCommand.Parameters.AddWithValue("@Oklad", tbOklad.Text);

sqlCommand.ExecuteNonQuery();

MessageBox.Show("Должность добавлена");

this.Hide();

}

catch (Exception ex)

{

MessageBox.Show("Должность не добавлена " + ex.Message);

}

database.DatabaseSQL().Close();

}

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

{

database.DatabaseSQL().Open();

try

{

SqlCommand sqlCommand = new SqlCommand("Dolj\_Update", database.DatabaseSQL());

sqlCommand.CommandType = CommandType.StoredProcedure;

sqlCommand.Parameters.AddWithValue("@ID\_Dolj", Program.ID\_Position);

sqlCommand.Parameters.AddWithValue("@Dolj", tbName.Text);

sqlCommand.Parameters.AddWithValue("@Oklad", tbOklad.Text);

sqlCommand.ExecuteNonQuery();

MessageBox.Show("Должность изменена");

this.Hide();

}

catch (Exception ex)

{

MessageBox.Show("Должность не изменена " + ex.Message);

}

database.DatabaseSQL().Close();

}

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

{

database.DatabaseSQL().Open();

try

{

if (Program.ID\_Dolj ==1)

{

SqlCommand sqlCommand = new SqlCommand("Dolj\_Delete", database.DatabaseSQL());

sqlCommand.CommandType = CommandType.StoredProcedure;

sqlCommand.Parameters.AddWithValue("@ID\_Dolj", Program.ID\_Position);

sqlCommand.ExecuteNonQuery();

}

else

{

SqlCommand sqlCommand = new SqlCommand("Dolj\_Logical\_Delete", database.DatabaseSQL());

sqlCommand.CommandType = CommandType.StoredProcedure;

sqlCommand.Parameters.AddWithValue("@ID\_Dolj", Program.ID\_Position);

sqlCommand.ExecuteNonQuery();

}

MessageBox.Show("Должность удалена");

this.Hide();

}

catch (Exception ex)

{

MessageBox.Show("Должность не удалена " + ex.Message);

}

database.DatabaseSQL().Close();

}

}

}

### Текст «FeedBackForm.cs»

using System;

using System.Windows.Forms;

using System.Net;

using System.Net.Mail;

using System.Drawing;

namespace Compo

{

public partial class FeedBackForm : Form

{

public FeedBackForm()

{

InitializeComponent();

Icon iconForm = new Icon(Application.StartupPath + "\\img\\compo2.ico");

Icon = iconForm;

openFileDialog1.Filter = "Text files(\*.txt)|\*.txt|All files(\*.\*)|\*.\*";

openFileDialog1.FileName = "";

}

public void MySendMail(string bodyMail, string nameAuthor, string subject)

{

string body = "<html><head> " +

"</head><body><center>Уведомление<br/> " +

"<img src=\"https://img.freepik.com/free-vector/abstract-dynamic-pattern-wallpaper-vector\_53876-59131.jpg?size=338&ext=jpg\"/>" +

"</center></br></body></html>";

try

{

// Отправитель

var from = new MailAddress(tbFrom.Text.ToString(), nameAuthor);

// Кому отправляем

var to = new MailAddress("andrey5062@yandex.ru");

// адрес smtp-сервера и порт, с которого будем отправлять письмо

SmtpClient smtp = new SmtpClient("smtp.yandex.ru", 587);

// логин и пароль

smtp.Credentials = new NetworkCredential(from.Address, "Pa$$w0rd");

smtp.EnableSsl = true;

// Создаем объект сообщения

MailMessage mail = new MailMessage(from, to);

// Указываем, что письмо с кодом html

mail.IsBodyHtml = true;

// Записываем тему письма

mail.Subject = subject;

// Содержимое пиьсма

mail.Body = body + bodyMail;

smtp.Send(mail);

MessageBox.Show("Сообщение доставлено!", "Информационное письмо", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

catch (Exception ex)

{

MessageBox.Show(ex.ToString(), "Ошибка отправки", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private void button1\_Click(object sender, EventArgs e)

{

//file = openFileDialog1.FileName;

// Перадаем значения в класс MySendMail

MySendMail(tbBody.Text, tbTema.Text, tbFrom.Text);

}

private void Button2\_Click(object sender, EventArgs e)

{

openFileDialog1.ShowDialog();

}

}

}

### Текст «GroupForm.cs»

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.Data.SqlClient;

namespace Compo

{

public partial class GroupForm : Form

{

DatabaseConnection database = new DatabaseConnection();

public GroupForm()

{

InitializeComponent();

this.StartPosition = FormStartPosition.CenterScreen; //Расположение окна по центру монитора

this.FormBorderStyle = FormBorderStyle.FixedToolWindow;//Добавление иконки приложения

ShowSotr();

Icon iconForm = new Icon(Application.StartupPath + "\\img\\compo2.ico");

Icon = iconForm;

}

private void ShowSotr() //Вывод должности в comboBox

{

database.DatabaseSQL().Open();

{

cbSotr.DataSource = database.TableFill("select [Fam\_Sotr]+' '+[Name\_Sotr]+' '+[Otch\_Sotr]+' '+[Date\_Birth] as Sotr from [dbo].[Sotr]", database.DatabaseSQL()).Tables[0];

cbSotr.DisplayMember = "Sotr";

cbNomerGpoupi.DataSource = database.TableFill("select [ID\_Group] as OldGroup from [dbo].[Group]", database.DatabaseSQL()).Tables[0];

cbNomerGpoupi.DisplayMember = "OldGroup";

}

database.DatabaseSQL().Close();

}

private void CbGroup\_SelectedIndexChanged(object sender, EventArgs e)

{

try

{

database.DatabaseSQL().Open();

{

SqlCommand cmd = new SqlCommand($"select ID\_Sotr from [dbo].[Sotr] where [Fam\_Sotr]+' '+[Name\_Sotr]+' '+[Otch\_Sotr]+' '+[Date\_Birth] = '{cbSotr.Text}'", database.DatabaseSQL());

Program.ID\_Sotr = (int)cmd.ExecuteScalar();

}

}

catch { }

database.DatabaseSQL().Close();

}

private void cbNomerGpoupi\_SelectedIndexChanged(object sender, EventArgs e)

{

try

{

database.DatabaseSQL().Open();

{

SqlCommand cmd = new SqlCommand($"select ID\_Group from [dbo].[Group] where [ID\_Group] = {cbNomerGpoupi.Text}", database.DatabaseSQL());

Program.ID\_Group = (int)cmd.ExecuteScalar();

}

}

catch { }

database.DatabaseSQL().Close();

}

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

{

database.DatabaseSQL().Open();

try

{

{

SqlCommand sqlCommand = new SqlCommand("Group\_Insert", database.DatabaseSQL());

sqlCommand.CommandType = CommandType.StoredProcedure;

sqlCommand.Parameters.AddWithValue("@Date",mtbDate.Text);

sqlCommand.Parameters.AddWithValue("@Group\_Logical\_Delete",0);

sqlCommand.ExecuteNonQuery();

}

MessageBox.Show("Группа добавлена");

this.Hide();

}

catch (Exception ex)

{

MessageBox.Show("Группа не добавлена" + ex.Message);

}

database.DatabaseSQL().Close();

}

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

{

database.DatabaseSQL().Open();

try

{

{

SqlCommand sqlCommand = new SqlCommand("Group\_Update", database.DatabaseSQL());

sqlCommand.CommandType = CommandType.StoredProcedure;

sqlCommand.Parameters.AddWithValue("@ID\_Group", Program.ID\_Group);

sqlCommand.Parameters.AddWithValue("@Date", mtbDate.Text);

sqlCommand.ExecuteNonQuery();

}

MessageBox.Show("Группа изменена");

this.Hide();

}

catch (Exception ex)

{

MessageBox.Show("Группа не изменена" + ex.Message);

}

database.DatabaseSQL().Close();

}

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

{

database.DatabaseSQL().Open();

try

{

{

SqlCommand sqlCommand = new SqlCommand("Group\_Delete", database.DatabaseSQL());

sqlCommand.CommandType = CommandType.StoredProcedure;

sqlCommand.Parameters.AddWithValue("@ID\_Group", Program.ID\_Group);

sqlCommand.ExecuteNonQuery();

}

MessageBox.Show("Группа удалена");

this.Hide();

}

catch(Exception ex)

{

MessageBox.Show("Группа не удалена " + ex.Message);

}

database.DatabaseSQL().Close();

}

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

{

database.DatabaseSQL().Open();

try

{

{

SqlCommand sqlCommand = new SqlCommand("Sostav\_Group\_Insert", database.DatabaseSQL());

sqlCommand.CommandType = CommandType.StoredProcedure;

sqlCommand.Parameters.AddWithValue("@Group\_ID", Program.ID\_Group);

sqlCommand.Parameters.AddWithValue("@Sotr\_ID", Program.ID\_Sotr);

sqlCommand.ExecuteNonQuery();

}

MessageBox.Show("Сотрудник добавлен в группу");

this.Hide();

}

catch (Exception ex)

{

MessageBox.Show("Сотрудник не добавлен в группу" + ex.Message);

}

database.DatabaseSQL().Close();

}

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

{

database.DatabaseSQL().Open();

try

{

{

SqlCommand sqlCommand = new SqlCommand("Sostav\_Group\_Update", database.DatabaseSQL());

sqlCommand.CommandType = CommandType.StoredProcedure;

sqlCommand.Parameters.AddWithValue("@ID", Program.ID);

sqlCommand.Parameters.AddWithValue("@Group\_ID", Program.ID\_Group);

sqlCommand.Parameters.AddWithValue("@Sotr\_ID", Program.ID\_Sotr);

sqlCommand.ExecuteNonQuery();

}

MessageBox.Show("Состав группы изменен");

this.Hide();

}

catch (Exception ex)

{

MessageBox.Show("Состав группы не изменен" + ex.Message);

}

database.DatabaseSQL().Close();

}

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

{

database.DatabaseSQL().Open();

try

{

{

SqlCommand sqlCommand = new SqlCommand("Sostav\_Group\_Delete", database.DatabaseSQL());

sqlCommand.CommandType = CommandType.StoredProcedure;

sqlCommand.Parameters.AddWithValue("@ID", Program.ID);

sqlCommand.ExecuteNonQuery();

}

MessageBox.Show("Сотрудник удален из группы");

this.Hide();

}

catch (Exception ex)

{

MessageBox.Show("Сотрудник не удален из группы " + ex.Message);

}

database.DatabaseSQL().Close();

}

}

}

### Текст «MainForm.cs»

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Data.SqlClient;

using System.Drawing;

using System.IO;

using System.Linq;

using System.Text;

using System.Threading;

using System.Threading.Tasks;

using System.Windows.Forms;

using static System.Net.WebRequestMethods;

using Excel = Microsoft.Office.Interop.Excel;

using Word = Microsoft.Office.Interop.Word;

namespace Compo

{

public partial class MainForm : Form

{

DataBase\_Configuration data = new DataBase\_Configuration();

DatabaseConnection database = new DatabaseConnection();

public MainForm()

{

InitializeComponent();

Icon iconForm = new Icon(Application.StartupPath + "\\img\\compo2.ico");//Добавление иконки приложения

Icon = iconForm;

this.Text = "Compo";

DataGridLocation();

Dolj();

lblSearch.Text = "Поиск";

dataGridView1.RowHeadersVisible = false;

LichniyKabinet();

DataGridLocation();

}

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

{

DataGridLocation();

}

private void DataGridLocation()

{

dataGridView1.Top = 50;

dataGridView1.Left = 12;

dataGridView1.Height = ClientSize.Height - 60;

}

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

{

Application.Exit();

}

private void MainForm\_FormClosed(object sender, FormClosedEventArgs e)

{

Application.Exit();

}

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

{

this.Hide();

SignInForm signInForm = new SignInForm();

signInForm.Show();

}

private void LichniyKabinet()

{

database.DatabaseSQL().Close();

database.DatabaseSQL().Open();

string IF\_Sotr = "";

SqlCommand sqlCommand = new SqlCommand($"select [Name\_Sotr]+' '+[Fam\_Sotr] from [dbo].[Sotr] where [Login\_Sotr] = '{Program.Autoriz\_Sotr}'", database.DatabaseSQL());

IF\_Sotr = sqlCommand.ExecuteScalar().ToString();

здравствуйтеToolStripMenuItem.Text = ("Здравствуйте " + IF\_Sotr);

database.DatabaseSQL().Close();

}

private void Dolj()

{

if (Program.ID\_Dolj == 1) //Администратор

{

Restriction();

}

if (Program.ID\_Dolj == 2) //Cтарший инженер

{

Restriction();

doljToolStripMenuItem.Visible = false;

employeeListToolStripMenuItem.Visible = false;

StatusToolStripMenuItem.Visible = false;

TipRabotoolStripMenuItem.Visible = false;

}

if (Program.ID\_Dolj == 3) //Инженер

{

Restriction();

doljToolStripMenuItem.Visible = false;

employeeListToolStripMenuItem.Visible = false;

TTToolStripMenuItem.Visible = false;

StatusToolStripMenuItem.Visible = false;

TipRabotoolStripMenuItem.Visible = false;

}

}

private void Restriction()

{

doljToolStripMenuItem.Visible = true;

employeeListToolStripMenuItem.Visible = true;

GroupToolStripMenuItem.Visible = true;

TTToolStripMenuItem.Visible = true;

StatusToolStripMenuItem.Visible = true;

ZayavkaToolStripMenuItem.Visible = true;

TipRabotoolStripMenuItem.Visible = true;

VipolnennieRabotiToolStripMenuItem.Visible = true;

}

private void DataGridView1\_CellMouseDown(object sender, DataGridViewCellMouseEventArgs e)

{

if (this.Text == "Compo")

{

AddToolStripMenuItem.Visible = false;

}

else

{

AddToolStripMenuItem.Visible = true;

}

dataGridView1.ContextMenuStrip = contextMenuStrip1;

if (!e.RowIndex.Equals(-1) && !e.ColumnIndex.Equals(-1) && e.Button.Equals(MouseButtons.Right)) //Выделение всей строки ПКМ

{

dataGridView1.CurrentCell = dataGridView1[e.ColumnIndex, e.RowIndex];

dataGridView1.CurrentRow.Selected = true;

}

dataGridView1.SelectionMode = DataGridViewSelectionMode.FullRowSelect;

}

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

{

ChangePasswordForm changePassword = new ChangePasswordForm();

changePassword.ShowDialog();

}

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

{

AboutBox aboutbox = new AboutBox();

aboutbox.ShowDialog();

}

private void Cheack\_MS\_Office\_Click(object sender, EventArgs e)

{

string sVersion = string.Empty;

Microsoft.Office.Interop.Word.Application appVersion = new Microsoft.Office.Interop.Word.Application();

appVersion.Visible = false;

switch (appVersion.Version.ToString())

{

case "7.0":

sVersion = "95";

break;

case "8.0":

sVersion = "97";

break;

case "9.0":

sVersion = "2000";

break;

case "10.0":

sVersion = "2002";

break;

case "11.0":

sVersion = "2003";

break;

case "12.0":

sVersion = "2007";

break;

case "14.0":

sVersion = "2010";

break;

case "15.0":

sVersion = "2013";

break;

case "16.0":

sVersion = "2016";

break;

default:

sVersion = "Too Old!";

break;

}

MessageBox.Show("На компьюторе обнаружена версия MS Office : " + sVersion);

}

private void Status\_Word\_Click(object sender, EventArgs e)

{

using (var regWord = Microsoft.Win32.Registry.ClassesRoot.OpenSubKey("Word.Application"))

{

if (regWord == null)

{

MessageBox.Show("Microsoft Word не установлен");

}

else

{

MessageBox.Show("Microsoft Word установлен");

}

}

}

private void Status\_Excel\_Click(object sender, EventArgs e)

{

Type officeType = Type.GetTypeFromProgID("Excel.Application");

if (officeType == null)

{

MessageBox.Show("Microsoft Excel не установлен");

}

else

{

MessageBox.Show("Microsoft Excel установлен");

}

}

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

{

ShowDolj();

}

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

{

ShowFillEmployeeList();

}

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

{

ShowTT();

}

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

{

ShowGroup();

}

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

{

ShowStatus();

}

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

{

ShowZayavka();

}

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

{

ShowTipRabot();

}

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

{

ShowVipolnennieraboti();

}

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

{

ShowOtchet();

}

private void ShowDolj()

{

this.Text = "Compo | Должности";

database.DatabaseSQL().Open();

{

if (Program.ID\_Dolj == 1)

dataGridView1.DataSource = database.TableFill("select [ID\_Dolj] as 'ID' ,[Dolj] as 'Название дожности', [Oklad] as 'Оклад',[Dolj\_Logical\_Delete] from [dbo].[Dolj]", database.DatabaseSQL()).Tables[0];

else

dataGridView1.DataSource = database.TableFill("select [ID\_Dolj] as 'ID' ,[Dolj] as 'Название дожности', [Oklad] as 'Оклад' from [dbo].[Dolj] where [Dolj\_Logical\_Delete] = 0", database.DatabaseSQL()).Tables[0];

}

database.DatabaseSQL().Close();

}

private void ShowFillEmployeeList()

{

this.Text = "Compo | Сотрудники";

database.DatabaseSQL().Open();

{

if (Program.ID\_Dolj == 1)

dataGridView1.DataSource = database.TableFill("select \* from View\_Sotr", database.DatabaseSQL()).Tables[0];

else

dataGridView1.DataSource = database.TableFill("select [ID],[Фамилия],[Имя],[Отчество],[Серия паспорта],[Номер паспорта],[Дата рождения],[Должность],[Оклад], from View\_Sotr where [Sotr\_Logical\_Delete] = 0", database.DatabaseSQL()).Tables[0];

}

database.DatabaseSQL().Close();

}

private void ShowTT()

{

this.Text = "Compo | Торговые точки";

database.DatabaseSQL().Open();

{

if (Program.ID\_Dolj == 1)

dataGridView1.DataSource = database.TableFill("select [ID\_TT] as 'ID', [TT\_Name] as 'Имя торговой точки', [Kol\_vo\_Ust\_nix\_Polok] as 'Кол-во установленных полок', [Adres] as 'Адрес', [TT\_Logical\_Delete] from [dbo].[TT]", database.DatabaseSQL()).Tables[0];

else

dataGridView1.DataSource = database.TableFill("select [ID\_TT] as 'ID', [TT\_Name] as 'Имя торговой точки', [Kol\_vo\_Ust\_nix\_Polok] as 'Кол-во установленных полок', [Adres] as 'Адрес' from [dbo].[TT] where [TT\_Logical\_Delete] = 0", database.DatabaseSQL()).Tables[0];

}

database.DatabaseSQL().Close();

}

private void ShowGroup()

{

this.Text = "Compo | Дежурные группы";

database.DatabaseSQL().Open();

{

if (Program.ID\_Dolj == 1)

dataGridView1.DataSource = database.TableFill("select \* from View\_Group", database.DatabaseSQL()).Tables[0];

else

dataGridView1.DataSource = database.TableFill("select \* from View\_Group", database.DatabaseSQL()).Tables[0];

}

database.DatabaseSQL().Close();

}

private void ShowStatus()

{

this.Text = "Compo | Статусы";

database.DatabaseSQL().Open();

{

if (Program.ID\_Dolj == 1)

dataGridView1.DataSource = database.TableFill("Select [ID\_Status] as 'ID', [Status\_Name] as 'Статус', Status\_Logical\_Delete from [dbo].[Status]", database.DatabaseSQL()).Tables[0];

else

dataGridView1.DataSource = database.TableFill("Select [ID\_Status] as 'ID', [Status\_Name] as 'Статус' from [dbo].[Status] where [Status\_Logical\_Delete] = 0", database.DatabaseSQL()).Tables[0];

}

database.DatabaseSQL().Close();

}

private void ShowZayavka()

{

this.Text = "Compo | Заявки";

database.DatabaseSQL().Open();

{

if (Program.ID\_Dolj == 1)

dataGridView1.DataSource = database.TableFill("select \* from View\_Zayavka", database.DatabaseSQL()).Tables[0];

else

dataGridView1.DataSource = database.TableFill("Select [ID], [№ Группы], [ТТ], [Адрес ТТ], [Статус], [Дата подачи], [Дата начала работ],[Дата окончания работ],[Дата закрытия заявки],[Список работ],[Список материалов], [Комментарий торговой точки],[Комментарий группы] from View\_Zayavka where [Zayavka\_Logical\_Delete] = 0 ", database.DatabaseSQL()).Tables[0];

}

database.DatabaseSQL().Close();

}

private void ShowTipRabot()

{

this.Text = "Compo | Типы работ";

database.DatabaseSQL().Open();

{

if (Program.ID\_Dolj == 1)

dataGridView1.DataSource = database.TableFill("select [ID\_Tip\_Rabot] as 'ID', [Name\_Tip\_Rabot] as 'Тип работы', [Tip\_Rabot\_Logical\_Delete] from [dbo].[Tip\_Rabot]", database.DatabaseSQL()).Tables[0];

else

dataGridView1.DataSource = database.TableFill("select [ID\_Tip\_Rabot] as 'ID', [Name\_Tip\_Rabot] as 'Тип работы' from [dbo].[Tip\_Rabot] where [TipRabot\_Logical\_Delete] = 0", database.DatabaseSQL()).Tables[0];

}

database.DatabaseSQL().Close();

}

private void ShowVipolnennieraboti()

{

this.Text = "Compo | Выполненные работы";

database.DatabaseSQL().Open();

{

if (Program.ID\_Dolj == 1)

dataGridView1.DataSource = database.TableFill("select [ID],[Тип работы],[ТТ],[Vip\_no\_Rabot\_Logical\_Delete] from View\_Vip\_no\_Rabot", database.DatabaseSQL()).Tables[0];

else

dataGridView1.DataSource = database.TableFill("select [ID],[Тип работы],[ТТ] from View\_Vip\_no\_Rabot where [Vip\_no\_Rabot\_Logical\_Delete] = 0 ", database.DatabaseSQL()).Tables[0];

}

database.DatabaseSQL().Close();

}

private void ShowOtchet()

{

this.Text = "Compo | Популярные проблемы";

database.DatabaseSQL().Open();

{

dataGridView1.DataSource = database.TableFill("select count ([Тип работы])as [Количество],[Тип работы] from [dbo].[View\_Vip\_no\_Rabot] group by [Тип работы] order by [Тип работы]", database.DatabaseSQL()).Tables[0];

}

database.DatabaseSQL().Close();

}

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

{

database.DatabaseSQL().Close();

AddConMenuStripButton();

}

private void AddConMenuStripButton()

{

if (this.Text == "Compo | Должности")

{

DoljForm Dolj = new DoljForm();

Dolj.btnUpdate.Enabled = false;

Dolj.btnDelete.Enabled = false;

Dolj.ShowDialog();

}

if (this.Text == "Compo | Сотрудники")

{

SigUpForm Employee = new SigUpForm();

Employee.btnUpdate.Enabled = false;

Employee.btnDelete.Enabled = false;

Employee.ShowDialog();

}

if (this.Text == "Compo | Торговые точки")

{

TTForm TT = new TTForm();

TT.btnUpdate.Enabled = false;

TT.btnDelete.Enabled = false;

TT.ShowDialog();

}

if (this.Text == "Compo | Дежурные группы")

{

GroupForm Group = new GroupForm();

Group.btnUpdate.Enabled = false;

Group.btnDelete.Enabled = false;

Group.btnIzmenit.Enabled = false;

Group.btnSnyat.Enabled = false;

Group.ShowDialog();

}

if (this.Text == "Compo | Статусы")

{

StatusForm Status = new StatusForm();

Status.btnUpdate.Enabled = false;

Status.btnDelete.Enabled = false;

Status.ShowDialog();

}

if (this.Text == "Compo | Заявки")

{

ZayavkaForm Zayavka = new ZayavkaForm();

Zayavka.btnUpdate.Enabled = false;

Zayavka.btnDelete.Enabled = false;

Zayavka.ShowDialog();

}

if (this.Text == "Compo | Типы работ")

{

TipRabotForm TipRabot = new TipRabotForm();

TipRabot.btnUpdate.Enabled = false;

TipRabot.btnDelete.Enabled = false;

TipRabot.ShowDialog();

}

if (this.Text == "Compo | Выполненные работы")

{

Vip\_no\_RabotForm Vip\_no\_Rabot = new Vip\_no\_RabotForm();

Vip\_no\_Rabot.btnUpdate.Enabled = false;

Vip\_no\_Rabot.btnDelete.Enabled = false;

Vip\_no\_Rabot.ShowDialog();

}

}

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

{

database.DatabaseSQL().Close();

ChangeConMenuStripButton();

}

private void ChangeConMenuStripButton()

{

if (this.Text == "Compo | Должности")

{

DoljForm position = new DoljForm();

Program.ID\_Position = (int)dataGridView1.CurrentRow.Cells[0].Value;

position.tbName.Text = dataGridView1.CurrentRow.Cells[1].Value.ToString();

position.tbOklad.Text = dataGridView1.CurrentRow.Cells[2].Value.ToString();

position.btnAdd.Enabled = false;

position.ShowDialog();

}

if (this.Text == "Compo | Сотрудники")

{

SigUpForm Employee = new SigUpForm();

Program.ID\_Sotr = (int)dataGridView1.CurrentRow.Cells[0].Value;

Employee.tbFam.Text = dataGridView1.CurrentRow.Cells[1].Value.ToString();

Employee.tbName.Text = dataGridView1.CurrentRow.Cells[2].Value.ToString();

Employee.tbOtch.Text = dataGridView1.CurrentRow.Cells[3].Value.ToString();

Employee.mtbSer.Text = dataGridView1.CurrentRow.Cells[4].Value.ToString();

Employee.mtbNum.Text = dataGridView1.CurrentRow.Cells[5].Value.ToString();

Employee.mtbDateBirth.Text = dataGridView1.CurrentRow.Cells[6].Value.ToString();

Employee.cbDolj.Text = dataGridView1.CurrentRow.Cells[7].Value.ToString();

Employee.btnAdd.Enabled = false;

Employee.ShowDialog();

}

if (this.Text == "Compo | Торговые точки")

{

TTForm TT = new TTForm();

Program.ID\_TT = (int)dataGridView1.CurrentRow.Cells[0].Value;

TT.tbNameTT.Text = dataGridView1.CurrentRow.Cells[1].Value.ToString();

TT.tbPrice.Text = dataGridView1.CurrentRow.Cells[2].Value.ToString();

TT.tbAdres.Text = dataGridView1.CurrentRow.Cells[3].Value.ToString();

TT.btnAdd.Enabled = false;

TT.ShowDialog();

}

if (this.Text == "Compo | Дежурные группы")

{

GroupForm Group = new GroupForm();

Program.ID\_Group = (int)dataGridView1.CurrentRow.Cells[0].Value;

Program.ID = (int)dataGridView1.CurrentRow.Cells[0].Value;

Group.mtbDate.Text = dataGridView1.CurrentRow.Cells[2].Value.ToString();

Group.btnAdd.Enabled = false;

Group.btnNaznachit.Enabled = false;

Group.ShowDialog();

}

if (this.Text == "Compo | Статусы")

{

StatusForm Status = new StatusForm();

Program.ID\_Status = (int)dataGridView1.CurrentRow.Cells[0].Value;

Status.tbStatus.Text = dataGridView1.CurrentRow.Cells[1].Value.ToString();

Status.btnAdd.Enabled = false;

Status.ShowDialog();

}

if (this.Text == "Compo | Заявки")

{

ZayavkaForm Zayavka = new ZayavkaForm();

Program.ID\_Zayavka = (int)dataGridView1.CurrentRow.Cells[0].Value;

Zayavka.btnAdd.Enabled = false;

Zayavka.Show();

}

if (this.Text == "Compo | Типы работ")

{

TipRabotForm TipRabot = new TipRabotForm();

Program.ID\_TipRabot = (int)dataGridView1.CurrentRow.Cells[0].Value;

TipRabot.tbTipRabot.Text = dataGridView1.CurrentRow.Cells[1].Value.ToString();

TipRabot.btnAdd.Enabled = false;

TipRabot.ShowDialog();

}

if (this.Text == "Compo | Выполненные работы")

{

Vip\_no\_RabotForm Vip\_no\_Rabot = new Vip\_no\_RabotForm();

Program.ID\_Vip\_no\_Rabot = (int)dataGridView1.CurrentRow.Cells[0].Value;

Vip\_no\_Rabot.btnAdd.Enabled = false;

Vip\_no\_Rabot.ShowDialog();

}

}

private void tbSearch\_TextChanged(object sender, EventArgs e)

{

Search();

}

private void Search()

{

database.DatabaseSQL().Open();

{

switch (this.Text)

{

case "Compo | Должности":

dataGridView1.DataSource = database.TableFill($"select [ID\_Dolj], [Dolj], [Oklad] from [dbo].[Dolj] where [Dolj] like '%{tbSearch.Text}%'or [Oklad] like '%{tbSearch.Text}%'", database.DatabaseSQL()).Tables[0];

break;

case "Compo | Сотрудники":

dataGridView1.DataSource = database.TableFill($"select [ID],[Фамилия],[Имя],[Отчество],[Серия паспорта],[Номер паспорта],[Дата рождения],[Должность],[Оклад] from View\_Sotr where [Фамилия]+' '+[Имя]+' '+[Отчество] like '%{tbSearch.Text}%' or [Дата рождения] like '%{tbSearch.Text}%'", database.DatabaseSQL()).Tables[0];

break;

case "Compo | Торговые точки":

dataGridView1.DataSource = database.TableFill($"select [ID\_TT] as 'ID', [TT\_Name] as 'Имя торговой точки', [Kol\_vo\_Ust\_nix\_Polok] as 'Кол-во установленных полок', [Adres] as 'Адрес' from [dbo].[TT] where [TT\_Name] like '%{tbSearch.Text}%' or [Kol\_vo\_Ust\_nix\_Polok] like '%{tbSearch.Text}%' or [Adres] like '%{tbSearch.Text}%'", database.DatabaseSQL()).Tables[0];

break;

case "Compo | Дежурные группы":

dataGridView1.DataSource = database.TableFill($"select \* from View\_Group where [№ Группы] like '%{tbSearch.Text}%' or [ФИО] like '%{tbSearch.Text}%' or [Дата]like '%{tbSearch.Text}%'", database.DatabaseSQL()).Tables[0];

break;

case "Compo | Статусы":

dataGridView1.DataSource = database.TableFill($"Select [ID\_Status] as 'ID', [Status\_Name] as 'Статус' from [dbo].[Status] where [Status\_Name] like '{tbSearch.Text}'", database.DatabaseSQL()).Tables[0];

break;

case "Compo | Заявки":

dataGridView1.DataSource = database.TableFill($"Select [ID], [№ Группы], [ТТ], [Адрес ТТ], [Статус], [Дата подачи], [Дата начала работ],[Дата окончания работ],[Дата закрытия заявки],[Список работ],[Список материалов], [Комментарий торговой точки],[Комментарий группы] from View\_Zayavka where [№ Группы] like '%{tbSearch.Text}%' or [ТТ] like '%{tbSearch.Text}%' or [Адрес ТТ] like '%{tbSearch.Text}%' or [Статус] like '%{tbSearch.Text}%' or [Дата подачи] like '%{tbSearch.Text}%' or [Дата начала работ] like '%{tbSearch.Text}%' or [Дата окончания работ] like '%{tbSearch.Text}%' or [Дата закрытия заявки] like '%{tbSearch.Text}%' or [Список работ] like '%{tbSearch.Text}%' or [Список материалов] like '%{tbSearch.Text}%' or [Комментарий торговой точки] like '%{tbSearch.Text}%' or [Комментарий группы] like '%{tbSearch.Text}%' ", database.DatabaseSQL()).Tables[0];

break;

case "Compo | Типы работ":

dataGridView1.DataSource = database.TableFill($"select [ID\_Tip\_Rabot] as 'ID', [Name\_Tip\_Rabot] as 'Тип работы' from [dbo].[Tip\_Rabot] where [Name\_Tip\_Rabot] like '%{tbSearch.Text}%' ", database.DatabaseSQL()).Tables[0];

break;

case "Compo | Выполненные работы":

dataGridView1.DataSource = database.TableFill($"select [ID],[Тип работы],[ТТ] from View\_Vip\_no\_Rabot where [Тип работы] like '%{tbSearch.Text}%' or [ТТ] like '%{tbSearch.Text}%'", database.DatabaseSQL()).Tables[0];

break;

}

}

database.DatabaseSQL().Close();

}

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

{

try

{

Microsoft.Office.Interop.Excel.Application excel = new Microsoft.Office.Interop.Excel.Application();

excel.Visible = true;

Microsoft.Office.Interop.Excel.Workbook workbook = excel.Workbooks.Add(System.Reflection.Missing.Value);

Microsoft.Office.Interop.Excel.Worksheet sheet1 = (Microsoft.Office.Interop.Excel.Worksheet)workbook.Sheets[1];

int StartCol = 1;

int StartRow = 1;

int j = 0, i = 0;

//Write Headers

for (j = 0; j < dataGridView1.Columns.Count; j++)

{

Microsoft.Office.Interop.Excel.Range myRange = (Microsoft.Office.Interop.Excel.Range)sheet1.Cells[StartRow, StartCol + j];

myRange.Value2 = dataGridView1.Columns[j].HeaderText;

}

StartRow++;

//Write datagridview content

for (i = 0; i < dataGridView1.Rows.Count; i++)

{

for (j = 0; j < dataGridView1.Columns.Count; j++)

{

try

{

Microsoft.Office.Interop.Excel.Range myRange = (Microsoft.Office.Interop.Excel.Range)sheet1.Cells[StartRow + i, StartCol + j];

myRange.Value2 = dataGridView1[j, i].Value == null ? "" : dataGridView1[j, i].Value;

}

catch

{

;

}

}

}

}

catch (Exception ex)

{

MessageBox.Show(ex.ToString());

}

}

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

{

switch (this.Text)

{

case "Compo | Должности":

ShowDolj();

break;

case "Compo | Сотрудники":

ShowFillEmployeeList();

break;

case "Compo | Торговые точки":

ShowTT();

break;

case "Compo | Дежурные группы":

ShowGroup();

break;

case "Compo | Статусы":

ShowStatus();

break;

case "Compo | Заявки":

ShowZayavka();

break;

case "Compo | Типы работ":

ShowTipRabot();

break;

case "Compo | Выполненные работы":

ShowVipolnennieraboti();

break;

case "Compo | Популярные проблемы":

ShowOtchet();

break;

}

}

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

{

SaveFileDialog sfd = new SaveFileDialog();

sfd.Filter = "Word Documents (\*.docx)|\*.docx";

sfd.FileName = "Отчет " + System.DateTime.Now.ToShortDateString();

switch (this.Text)

{

case "Compo | Сотрудники":

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

{

Export\_Data\_To\_Word(dataGridView1, sfd.FileName, "Список сотрудников");

}

break;

case "Compo | Статусы":

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

{

Export\_Data\_To\_Word(dataGridView1, sfd.FileName, "Список статусов");

}

break;

case "Compo | Должности":

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

{

Export\_Data\_To\_Word(dataGridView1, sfd.FileName, "Список должностей");

}

break;

case "Compo | Торговые точки":

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

{

Export\_Data\_To\_Word(dataGridView1, sfd.FileName, "Список TT");

}

break;

case "Compo | Заявки":

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

{

Export\_Data\_To\_Word(dataGridView1, sfd.FileName, "Список заявок");

}

break;

case "Compo | Дежурные группы":

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

{

Export\_Data\_To\_Word(dataGridView1, sfd.FileName, "Список групп");

}

break;

case "Compo | Типы работ":

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

{

Export\_Data\_To\_Word(dataGridView1, sfd.FileName, "Список типов работ");

}

break;

case "Compo | Выполненные работы":

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

{

Export\_Data\_To\_Word(dataGridView1, sfd.FileName, "Отчет о выполненных работах " + System.DateTime.Now.ToString());

}

break;

case "Compo | Популярные проблемы":

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

{

Export\_Data\_To\_Word(dataGridView1, sfd.FileName, "Отчет популярных проблем " + System.DateTime.Now.ToString());

}

break;

}

}

private void Export\_Data\_To\_Word(DataGridView DGV, string filename, string headerTable)

{

if (DGV.Rows.Count != 0)

{

int RowCount = DGV.Rows.Count;

int ColumnCount = DGV.Columns.Count;

Object[,] DataArray = new object[RowCount + 1, ColumnCount + 1];

//add rows

int r = 0;

for (int c = 0; c <= ColumnCount - 1; c++)

{

for (r = 0; r <= RowCount - 1; r++)

{

DataArray[r, c] = DGV.Rows[r].Cells[c].Value;

} //end row loop

} //end column loop

Word.Document oDoc = new Word.Document();

oDoc.Application.Visible = true;

//page orintation

oDoc.PageSetup.Orientation = Word.WdOrientation.wdOrientLandscape;

dynamic oRange = oDoc.Content.Application.Selection.Range;

string oTemp = "";

for (r = 0; r <= RowCount - 1; r++)

{

for (int c = 0; c <= ColumnCount - 1; c++)

{

oTemp = oTemp + DataArray[r, c] + "\t";

}

}

//table format

oRange.Text = oTemp;

object Separator = Word.WdTableFieldSeparator.wdSeparateByTabs;

object ApplyBorders = true;

object AutoFit = true;

object AutoFitBehavior = Word.WdAutoFitBehavior.wdAutoFitContent;

oRange.ConvertToTable(ref Separator, ref RowCount, ref ColumnCount,

Type.Missing, Type.Missing, ref ApplyBorders,

Type.Missing, Type.Missing, Type.Missing,

Type.Missing, Type.Missing, Type.Missing,

Type.Missing, ref AutoFit, ref AutoFitBehavior, Type.Missing);

oDoc.Application.Selection.Tables[1].Select();

oDoc.Application.Selection.Tables[1].Rows.AllowBreakAcrossPages = 0;

oDoc.Application.Selection.Tables[1].Rows.Alignment = 0;

oDoc.Application.Selection.Tables[1].Rows[1].Select();

oDoc.Application.Selection.InsertRowsAbove(1);

oDoc.Application.Selection.Tables[1].Rows[1].Select();

//header row style

oDoc.Application.Selection.Tables[1].Rows[1].Range.Bold = 1;

oDoc.Application.Selection.Tables[1].Rows[1].Range.Font.Name = "Times New Roman";

oDoc.Application.Selection.Tables[1].Rows[1].Range.Font.Size = 14;

//add header row manually

for (int c = 0; c <= ColumnCount - 1; c++)

{

oDoc.Application.Selection.Tables[1].Cell(1, c + 1).Range.Text = DGV.Columns[c].HeaderText;

}

//table style

oDoc.Application.Selection.Tables[1].set\_Style("Классическая таблица 1");

oDoc.Application.Selection.Tables[1].Rows[1].Select();

oDoc.Application.Selection.Cells.VerticalAlignment = Word.WdCellVerticalAlignment.wdCellAlignVerticalCenter;

//header text

foreach (Word.Section section in oDoc.Application.ActiveDocument.Sections)

{

Word.Range headerRange = section.Headers[Word.WdHeaderFooterIndex.wdHeaderFooterPrimary].Range;

headerRange.Fields.Add(headerRange, Word.WdFieldType.wdFieldPage);

headerRange.Text = headerTable;

headerRange.Font.Size = 16;

headerRange.ParagraphFormat.Alignment = Word.WdParagraphAlignment.wdAlignParagraphCenter;

}

oDoc.SaveAs2(filename);

}

}

private async void PDFToolStripMenuItem\_Click(object sender, EventArgs e)

{

OpenFileDialog fileDialog = new OpenFileDialog();

fileDialog.ShowDialog();

PdfClass pdfClass = new PdfClass();

await Task.Run(() => pdfClass.ExportWordToPdf(fileDialog.FileName));

}

}

}

### Текст «PdfClass.cs»

using Microsoft.Office.Interop.Word;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Compo

{

class PdfClass

{

public void ExportWordToPdf(string path)

{

try

{

Application word = new Application();

Document file = word.Documents.Open(path);

file.ExportAsFixedFormat(path + "info\_pdf.pdf", WdExportFormat.wdExportFormatPDF);

}

catch

{

}

}

}

}

### Текст «Program.cs»

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Compo

{

static class Program

{

public static string Autoriz\_Sotr;

public static int ID\_Position;

public static int ID\_Sotr;

public static int ID\_Dolj;

public static int ID\_Zayavka;

public static int ID\_TT;

public static int ID\_Group;

public static int ID\_Status;

public static int ID;

public static int ID\_TipRabot;

public static int ID\_Vip\_no\_Rabot;

/// <summary>

/// Главная точка входа для приложения.

/// </summary>

[STAThread]

static void Main()

{

Application.EnableVisualStyles();

Application.SetCompatibleTextRenderingDefault(false);

Application.Run(new Zastavka());

Application.Run(new SignInForm());

}

}

}

### Текст «SignInForm.cs»

using System;

using System.Data.SqlClient;

using System.Drawing;

using System.Security.Cryptography;

using System.Text;

using System.Windows.Forms;

namespace Compo

{

public partial class SignInForm : Form

{

DatabaseConnection DrySql = new DatabaseConnection();

public SignInForm()

{

InitializeComponent();

this.TopMost = true; //Вызов окна поверх других

this.StartPosition = FormStartPosition.CenterScreen; //Расположение окна по центру монитора

this.FormBorderStyle = FormBorderStyle.FixedToolWindow;//Добавление иконки приложения

Icon iconForm = new Icon(Application.StartupPath + "\\img\\compo2.ico");

Icon = iconForm;

tbPassword.PasswordChar = '\*';

this.Text = "Окно Авторизации";

lblAutorization.Text = "Авторизация";

lblAutorization.Left = (ClientSize.Width - lblAutorization.Width) / 2; //Расположение в центре экрана

lblLogin.Text = "Логин";

LblPassword.Text = "Пароль";

btnOK.Text = "OK";

btnClose.Text = "Закрыть";

}

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

{

StatusUsers();

}

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

{

Application.Exit();

}

private void StatusUsers()

{

DrySql.DatabaseSQL().Open();

string loginStatus = tbLogin.Text;

string passwordStatus = GetHash(tbPassword.Text);

string StatusCmd = $"Select [Login\_Sotr], [Password\_Sotr] from [dbo].[Sotr] where [Login\_Sotr] = '{loginStatus}' and [Password\_Sotr] = '{passwordStatus}'";

string StatusRole = $"Select [Dolj\_ID] from [dbo].[Sotr] where [Login\_Sotr] ='{loginStatus}'";

var command = new SqlCommand(StatusCmd, DrySql.DatabaseSQL());

var commandRole = new SqlCommand(StatusRole, DrySql.DatabaseSQL());

command.Prepare();

command.ExecuteNonQuery();

if (loginStatus == (string)command.ExecuteScalar())

{

Program.ID\_Dolj = (int)commandRole.ExecuteScalar();

Program.Autoriz\_Sotr = (tbLogin.Text);

MessageBox.Show("Авторизация успешна");

this.Hide();

MainForm mainForm = new MainForm();

mainForm.Show();

}

else

{

MessageBox.Show("Логин или пароль ведены не верно");

}

DrySql.DatabaseSQL().Close();

}

private string GetHash(string input) //Хэширование пароля в БД

{

var md5 = MD5.Create();

var hash = md5.ComputeHash(Encoding.UTF8.GetBytes(input));

return Convert.ToBase64String(hash);

}

}

}

### Текст «SignUpForm.cs»

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Data.SqlClient;

using System.Drawing;

using System.Linq;

using System.Security.Cryptography;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using Word = Microsoft.Office.Interop.Word;

namespace Compo

{

public partial class SigUpForm : Form

{

DatabaseConnection database = new DatabaseConnection();

public SigUpForm()

{

this.TopMost = true; //Вызов окна поверх других

InitializeComponent();

this.StartPosition = FormStartPosition.CenterScreen; //Расположение окна по центру монитора

this.FormBorderStyle = FormBorderStyle.FixedToolWindow;//Добавление иконки приложения

this.StartPosition = FormStartPosition.CenterScreen; //Расположение окна по центру монитора

this.FormBorderStyle = FormBorderStyle.FixedToolWindow;//Добавление иконки приложения

Icon iconForm = new Icon(Application.StartupPath + "\\img\\compo2.ico");

Icon = iconForm;

this.Text = "Cотрудник";

lblFam.Text = "Фамилия";

lblName.Text = "Имя";

lblOtch.Text = "Отчество";

lblSer.Text = "Серия паспорта";

lblNum.Text = "Номер паспорта";

lblLogin.Text = "Логин";

lblPassword.Text = "Пароль";

lblRepPas.Text = "Повторите пароль";

lblDolj.Text = "Должность";

btnAdd.Text = "Добавить";

btnDelete.Text = "Удалить";

tbPas.PasswordChar = '\*';

tbRepPas.PasswordChar = '\*';

ShowDolj();

}

private void ShowDolj() //Вывод должности в comboBox

{

database.DatabaseSQL().Open();

{

cbDolj.DataSource = database.TableFill("select Dolj from [dbo].[Dolj]", database.DatabaseSQL()).Tables[0];

cbDolj.DisplayMember = "Dolj";

}

database.DatabaseSQL().Close();

}

private void ComboBox1\_SelectedIndexChanged(object sender, EventArgs e)

{

try

{

database.DatabaseSQL().Open();

{

SqlCommand cmd = new SqlCommand($"select ID\_Dolj from [dbo].[Dolj] where Dolj = '{cbDolj.Text}'", database.DatabaseSQL());

Program.ID\_Position = (int)cmd.ExecuteScalar();

}

database.DatabaseSQL().Close();

}

catch { }

}

private string GetHash(string input) //Хэширование пароля в БД

{

var md5 = MD5.Create();

var hash = md5.ComputeHash(Encoding.UTF8.GetBytes(input));

return Convert.ToBase64String(hash);

}

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

{

database.DatabaseSQL().Open();

try

{

if (tbPas.Text == tbRepPas.Text)

{

string passwordHash = GetHash(tbPas.Text);

{

var sqlCommand = new SqlCommand("Sotr\_Insert", database.DatabaseSQL());

sqlCommand.CommandType = CommandType.StoredProcedure;

sqlCommand.Parameters.AddWithValue("@Fam\_Sotr", tbFam.Text);

sqlCommand.Parameters.AddWithValue("@Name\_Sotr", tbName.Text);

sqlCommand.Parameters.AddWithValue("@Otch\_Sotr", tbOtch.Text);

sqlCommand.Parameters.AddWithValue("@Ser\_Pas", mtbSer.Text);

sqlCommand.Parameters.AddWithValue("@Num\_Pas", mtbNum.Text);

sqlCommand.Parameters.AddWithValue("@Date\_Birth", mtbDateBirth.Text);

sqlCommand.Parameters.AddWithValue("@Login\_Sotr", tbLogin.Text);

sqlCommand.Parameters.AddWithValue("@Password\_Sotr", passwordHash);

sqlCommand.Parameters.AddWithValue("@Dolj\_ID", Program.ID\_Position);

sqlCommand.ExecuteNonQuery();

}

MessageBox.Show("Сотрудник добавлен");

this.Hide();

}

else

{

MessageBox.Show("Пароли не совпадают");

}

}

catch (Exception ex)

{

MessageBox.Show("Сотрудник не добавлен" + ex.Message);

}

database.DatabaseSQL().Close();

}

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

{

database.DatabaseSQL().Open();

try

{

var sqlCommand = new SqlCommand("Sotr\_Update", database.DatabaseSQL());

sqlCommand.CommandType = CommandType.StoredProcedure;

sqlCommand.Parameters.AddWithValue("@ID\_Sotr", Program.ID\_Sotr);

sqlCommand.Parameters.AddWithValue("@Fam\_Sotr", tbFam.Text);

sqlCommand.Parameters.AddWithValue("@Name\_Sotr", tbName.Text);

sqlCommand.Parameters.AddWithValue("@Otch\_Sotr", tbOtch.Text);

sqlCommand.Parameters.AddWithValue("@Ser\_Pas", mtbSer.Text);

sqlCommand.Parameters.AddWithValue("@Num\_Pas", mtbNum.Text);

sqlCommand.Parameters.AddWithValue("@Dolj\_ID", Program.ID\_Position);

sqlCommand.Parameters.AddWithValue("@Date\_Birth", mtbDateBirth.Text);

sqlCommand.ExecuteNonQuery();

MessageBox.Show("Сотрудник изменен");

this.Hide();

}

catch (Exception ex)

{

MessageBox.Show("Сотрудник не изменен " + ex.Message);

}

database.DatabaseSQL().Close();

}

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

{

database.DatabaseSQL().Open();

try

{

if (Program.ID\_Dolj == 1)

{

SqlCommand sqlCommand = new SqlCommand("Sotr\_Delete", database.DatabaseSQL());

sqlCommand.CommandType = CommandType.StoredProcedure;

sqlCommand.Parameters.AddWithValue("@ID\_Sotr", Program.ID\_Sotr);

sqlCommand.ExecuteNonQuery();

}

else

{

SqlCommand sqlCommand = new SqlCommand("Sotr\_Logical\_Delete", database.DatabaseSQL());

sqlCommand.CommandType = CommandType.StoredProcedure;

sqlCommand.Parameters.AddWithValue("@ID\_Sotr", Program.ID\_Sotr);

sqlCommand.ExecuteNonQuery();

}

MessageBox.Show("Сотрудник удален");

this.Hide();

}

catch (Exception ex)

{

MessageBox.Show("Сотрудник не удален " + ex.Message);

}

database.DatabaseSQL().Close();

}

}

}

Текст «StatusForm.cs»

using System;

using System.Data;

using System.Drawing;

using System.Windows.Forms;

using System.Data.SqlClient;

namespace Compo

{

public partial class StatusForm : Form

{

DatabaseConnection database = new DatabaseConnection();

public StatusForm()

{

InitializeComponent();

this.StartPosition = FormStartPosition.CenterScreen; //Расположение окна по центру монитора

this.FormBorderStyle = FormBorderStyle.FixedToolWindow;//Добавление иконки приложения

Icon iconForm = new Icon(Application.StartupPath + "\\img\\compo2.ico");

Icon = iconForm;

}

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

{

database.DatabaseSQL().Open();

{

try

{

SqlCommand sqlCommand = new SqlCommand("Status\_Insert", database.DatabaseSQL());

sqlCommand.CommandType = CommandType.StoredProcedure;

sqlCommand.Parameters.AddWithValue("@Status\_Name", tbStatus.Text);

sqlCommand.Parameters.AddWithValue("@Status\_Logical\_Delete",0);

sqlCommand.ExecuteNonQuery();

MessageBox.Show("Статус добавлен");

this.Hide();

}

catch (Exception ex)

{

MessageBox.Show("Статус не добавлен " + ex.Message);

}

}

database.DatabaseSQL().Close();

}

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

{

database.DatabaseSQL().Open();

{

try

{

SqlCommand sqlCommand = new SqlCommand("Status\_Update", database.DatabaseSQL());

sqlCommand.CommandType = CommandType.StoredProcedure;

sqlCommand.Parameters.AddWithValue("@ID\_Status", Program.ID\_Status);

sqlCommand.Parameters.AddWithValue("@Status\_Name",tbStatus.Text);

sqlCommand.ExecuteNonQuery();

MessageBox.Show("Статус изменен");

this.Hide();

}

catch (Exception ex)

{

MessageBox.Show("Статус не изменен " + ex.Message);

}

database.DatabaseSQL().Close();

}

}

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

{

database.DatabaseSQL().Open();

{

try

{

if (Program.ID\_Dolj == 1)

{

SqlCommand sqlCommand = new SqlCommand("Status\_Delete", database.DatabaseSQL());

sqlCommand.CommandType = CommandType.StoredProcedure;

sqlCommand.Parameters.AddWithValue("@ID\_Status", Program.ID\_Status);

sqlCommand.ExecuteNonQuery();

}

else

{

SqlCommand sqlCommand = new SqlCommand("Status\_Logical\_Delete", database.DatabaseSQL());

sqlCommand.CommandType = CommandType.StoredProcedure;

sqlCommand.Parameters.AddWithValue("@ID\_Status", Program.ID\_Status);

sqlCommand.ExecuteNonQuery();

}

MessageBox.Show("Статус удален");

this.Hide();

}

catch (Exception ex)

{

MessageBox.Show("Статус не удален" + ex.Message);

}

database.DatabaseSQL().Close();

}

}

}

}

### Текст «TipRabotForm.cs»

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 Compo

{

public partial class TipRabotForm : Form

{

DatabaseConnection database = new DatabaseConnection();

public TipRabotForm()

{

this.TopMost = true; //Вызов окна поверх других

InitializeComponent();

this.StartPosition = FormStartPosition.CenterScreen; //Расположение окна по центру монитора

this.FormBorderStyle = FormBorderStyle.FixedToolWindow;//Добавление иконки приложения

Icon iconForm = new Icon(Application.StartupPath + "\\img\\compo2.ico");

Icon = iconForm;

UpdateForm();

}

private void UpdateForm()

{

}

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

{

database.DatabaseSQL().Open();

try

{

SqlCommand sqlCommand = new SqlCommand("Tip\_Rabot\_Insert", database.DatabaseSQL());

sqlCommand.CommandType = CommandType.StoredProcedure;

sqlCommand.Parameters.AddWithValue("@Name\_Tip\_Rabot", tbTipRabot.Text);

sqlCommand.Parameters.AddWithValue("@Tip\_Rabot\_Logical\_Delete", 0);

sqlCommand.ExecuteNonQuery();

MessageBox.Show("Тип работ добавлен");

this.Hide();

}

catch (Exception ex)

{

MessageBox.Show("Тип работ добавлен не добавлен " + ex.Message);

}

database.DatabaseSQL().Close();

}

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

{

database.DatabaseSQL().Open();

try

{

SqlCommand sqlCommand = new SqlCommand("Tip\_Rabot\_Update", database.DatabaseSQL());

sqlCommand.CommandType = CommandType.StoredProcedure;

sqlCommand.Parameters.AddWithValue("@ID\_Tip\_Rabot", Program.ID\_TipRabot);

sqlCommand.Parameters.AddWithValue("@Name\_Tip\_Rabot", tbTipRabot.Text);

sqlCommand.Parameters.AddWithValue("@Tip\_Rabot\_Logical\_Delete", 0);

sqlCommand.ExecuteNonQuery();

MessageBox.Show("Тип работ добавлен изменен");

this.Hide();

}

catch (Exception ex)

{

MessageBox.Show("Тип работ добавлен не изменен " + ex.Message);

}

database.DatabaseSQL().Close();

}

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

{

database.DatabaseSQL().Open();

try

{

if (Program.ID\_Dolj == 1)

{

SqlCommand sqlCommand = new SqlCommand("Tip\_Rabot\_Delete", database.DatabaseSQL());

sqlCommand.CommandType = CommandType.StoredProcedure;

sqlCommand.Parameters.AddWithValue("@ID\_Tip\_Rabot", Program.ID\_TipRabot);

sqlCommand.ExecuteNonQuery();

}

else

{

SqlCommand sqlCommand = new SqlCommand("Tip\_Rabot\_Logical\_Delete", database.DatabaseSQL());

sqlCommand.CommandType = CommandType.StoredProcedure;

sqlCommand.Parameters.AddWithValue("@ID\_Tip\_Rabot", Program.ID\_TipRabot);

sqlCommand.ExecuteNonQuery();

}

MessageBox.Show("Тип работ добавлен удален");

this.Hide();

}

catch (Exception ex)

{

MessageBox.Show("Тип работ добавлент не удален " + ex.Message);

}

database.DatabaseSQL().Close();

}

}

}

### Текст «TTForm.cs»

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 Compo

{

public partial class TTForm : Form

{

DatabaseConnection database = new DatabaseConnection();

public TTForm()

{

this.TopMost = true; //Вызов окна поверх других

InitializeComponent();

this.StartPosition = FormStartPosition.CenterScreen; //Расположение окна по центру монитора

this.FormBorderStyle = FormBorderStyle.FixedToolWindow;//Добавление иконки приложения

//Icon iconForm = new Icon(Application.StartupPath + "\\img\\Compo.ico");

//Icon = iconForm;

}

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

{

database.DatabaseSQL().Open();

try

{

SqlCommand sqlCommand = new SqlCommand("TT\_Insert", database.DatabaseSQL());

sqlCommand.CommandType = CommandType.StoredProcedure;

sqlCommand.Parameters.AddWithValue("@TT\_Name", tbNameTT.Text);

sqlCommand.Parameters.AddWithValue("@Kol\_vo\_Ust\_nix\_Polok", tbPrice.Text);

sqlCommand.Parameters.AddWithValue("@Adres", tbAdres.Text);

sqlCommand.Parameters.AddWithValue("@TT\_Logical\_Delete",0);

sqlCommand.ExecuteNonQuery();

MessageBox.Show("Торговая точка добавлена");

this.Hide();

}

catch (Exception ex)

{

MessageBox.Show("Торговая точка не добавлена " + ex.Message);

}

database.DatabaseSQL().Close();

}

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

{

database.DatabaseSQL().Open();

try

{

SqlCommand sqlCommand = new SqlCommand("TT\_Update", database.DatabaseSQL());

sqlCommand.CommandType = CommandType.StoredProcedure;

sqlCommand.Parameters.AddWithValue("@ID\_TT", Program.ID\_TT);

sqlCommand.Parameters.AddWithValue("@TT\_Name", tbNameTT.Text);

sqlCommand.Parameters.AddWithValue("@Kol\_vo\_Ust\_nix\_Polok", tbPrice.Text);

sqlCommand.Parameters.AddWithValue("@Adres", tbAdres.Text);

sqlCommand.ExecuteNonQuery();

MessageBox.Show("Торговая точка изменена");

this.Hide();

}

catch (Exception ex)

{

MessageBox.Show("Торговая точка изменена " + ex.Message);

}

database.DatabaseSQL().Close();

}

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

{

database.DatabaseSQL().Open();

try

{

if (Program.ID\_Dolj == 1)

{

SqlCommand sqlCommand = new SqlCommand("TT\_Delete", database.DatabaseSQL());

sqlCommand.CommandType = CommandType.StoredProcedure;

sqlCommand.Parameters.AddWithValue("@ID\_TT", Program.ID\_TT);

sqlCommand.ExecuteNonQuery();

}

else

{

SqlCommand sqlCommand = new SqlCommand("TT\_Logical\_Delete", database.DatabaseSQL());

sqlCommand.CommandType = CommandType.StoredProcedure;

sqlCommand.Parameters.AddWithValue("@ID\_TT", Program.ID\_TT);

sqlCommand.ExecuteNonQuery();

}

MessageBox.Show("Торговая точка удалена");

this.Hide();

}

catch (Exception ex)

{

MessageBox.Show("Торговая точка не удалена " + ex.Message);

}

database.DatabaseSQL().Close();

}

}

}

### Текст «Vip\_no\_RabotForm.cs»

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;

using Word = Microsoft.Office.Interop.Word;

namespace Compo

{

public partial class Vip\_no\_RabotForm : Form

{

DatabaseConnection database = new DatabaseConnection();

public Vip\_no\_RabotForm()

{

this.TopMost = true; //Вызов окна поверх других

InitializeComponent();

Icon iconForm = new Icon(Application.StartupPath + "\\img\\compo2.ico");

Icon = iconForm;

this.StartPosition = FormStartPosition.CenterScreen; //Расположение окна по центру монитора

this.FormBorderStyle = FormBorderStyle.FixedToolWindow;//Добавление иконки приложения

UpdateForm();

ShowComboBox();

}

private void UpdateForm()

{

database.DatabaseSQL().Close();

}

private void ShowComboBox()

{

database.DatabaseSQL().Open();

{

cbZayavka.DataSource = database.TableFill("select ТТ from [dbo].[View\_Zayavka]", database.DatabaseSQL()).Tables[0];

cbZayavka.DisplayMember = "ТТ";

cbTip\_Rabot.DataSource = database.TableFill("select [Name\_Tip\_Rabot] as Name from [dbo].[Tip\_Rabot]", database.DatabaseSQL()).Tables[0];

cbTip\_Rabot.DisplayMember = "Name";

}

database.DatabaseSQL().Close();

}

private void cbZayavka\_SelectedIndexChanged(object sender, EventArgs e)

{

try

{

database.DatabaseSQL().Open();

{

SqlCommand com = new SqlCommand($"select [ID] from [dbo].[View\_Zayavka] where [ТТ] = '{cbZayavka.Text}'", database.DatabaseSQL());

Program.ID\_Zayavka = (int)com.ExecuteScalar();

}

database.DatabaseSQL().Close();

}

catch { }

}

private void cbTip\_Rabot\_SelectedIndexChanged(object sender, EventArgs e)

{

try

{

database.DatabaseSQL().Open();

{

SqlCommand cmd = new SqlCommand($"select ID\_Tip\_Rabot from [dbo].[Tip\_Rabot] where Name\_Tip\_Rabot = '{cbTip\_Rabot.Text}'", database.DatabaseSQL());

Program.ID\_TipRabot = (int)cmd.ExecuteScalar();

}

database.DatabaseSQL().Close();

}

catch { }

}

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

{

database.DatabaseSQL().Open();

try

{

SqlCommand sqlCommand = new SqlCommand("Vip\_no\_Rabot\_Insert", database.DatabaseSQL());

sqlCommand.CommandType = CommandType.StoredProcedure;

sqlCommand.Parameters.AddWithValue("@Zayavka\_ID", Program.ID\_Zayavka);

sqlCommand.Parameters.AddWithValue("@Tip\_Rabot\_ID", Program.ID\_TipRabot);

sqlCommand.Parameters.AddWithValue("Vip\_no\_Rabot\_Logical\_Delete", 0);

sqlCommand.ExecuteNonQuery();

MessageBox.Show("Тип выполненной работы добавлен");

this.Hide();

}

catch (Exception ex)

{

MessageBox.Show("Тип выполненной работы не добавлен " + ex.Message);

}

database.DatabaseSQL().Close();

}

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

{

database.DatabaseSQL().Open();

try

{

SqlCommand sqlCommand = new SqlCommand("Vip\_no\_Rabot\_Update", database.DatabaseSQL());

sqlCommand.CommandType = CommandType.StoredProcedure;

sqlCommand.Parameters.AddWithValue("@ID\_Vip\_no\_Rabot", Program.ID\_Vip\_no\_Rabot);

sqlCommand.Parameters.AddWithValue("@Zayavka\_ID", Program.ID\_Zayavka);

sqlCommand.Parameters.AddWithValue("@Tip\_Rabot\_ID", Program.ID\_TipRabot);

sqlCommand.ExecuteNonQuery();

MessageBox.Show("Тип выполненной работы обновлен");

this.Hide();

}

catch (Exception ex)

{

MessageBox.Show("Тип выполненной работы не обновлен " + ex.Message);

}

database.DatabaseSQL().Close();

}

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

{

database.DatabaseSQL().Open();

try

{

if (Program.ID\_Dolj == 1)

{

SqlCommand sqlCommand = new SqlCommand("Vip\_no\_Rabot\_Delete", database.DatabaseSQL());

sqlCommand.CommandType = CommandType.StoredProcedure;

sqlCommand.Parameters.AddWithValue("@ID\_Vip\_no\_Rabot", Program.ID\_Vip\_no\_Rabot);

sqlCommand.ExecuteNonQuery();

}

else

{

SqlCommand sqlCommand = new SqlCommand("Vip\_no\_Rabot\_Logical\_Delete", database.DatabaseSQL());

sqlCommand.CommandType = CommandType.StoredProcedure;

sqlCommand.Parameters.AddWithValue("@ID\_Vip\_no\_Rabot", Program.ID\_Vip\_no\_Rabot);

sqlCommand.ExecuteNonQuery();

}

MessageBox.Show("Тип выполненной работы удален");

this.Hide();

}

catch (Exception ex)

{

MessageBox.Show("Тип выполненной работы не удален " + ex.Message);

}

database.DatabaseSQL().Close();

}

}

}

Текст «Zastavka.cs»

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Compo

{

public partial class Zastavka : Form

{

DataBase\_Configuration data = new DataBase\_Configuration();

public Zastavka()

{

this.TopMost = true; //Вызов окна поверх других

InitializeComponent();

Icon iconForm = new Icon(Application.StartupPath + "\\img\\compo2.ico");

Icon = iconForm;

this.StartPosition = FormStartPosition.CenterScreen; //Расположение окна по центру монитора

}

private void constate(bool value)

{

Action action = () =>

{

switch (value)

{

case (true):

SignInForm sign = new SignInForm();

Close();

sign.Show();

break;

case (false):

ConectionForm conection = new ConectionForm();

conection.Show();

timer1.Enabled = false;

this.Hide();

break;

}

};

Invoke(action);

}

public void Zastavka\_Load(object sender, EventArgs e)

{

data.conState += constate;

Thread thread = new Thread(data.Connection\_Check);

thread.Start();

}

private void Timer1\_Tick(object sender, EventArgs e)

{

if (progressBar1.Value != 17)

progressBar1.Value++;

else

this.Close();

}

}

}

### Текст «ZayavkaForm.cs»

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 Compo

{

public partial class ZayavkaForm : Form

{

DatabaseConnection database = new DatabaseConnection();

public ZayavkaForm()

{

this.TopMost = true; //Вызов окна поверх других

InitializeComponent();

this.StartPosition = FormStartPosition.CenterScreen; //Расположение окна по центру монитора

this.FormBorderStyle = FormBorderStyle.FixedToolWindow;//Добавление иконки приложения

Icon iconForm = new Icon(Application.StartupPath + "\\img\\compo2.ico");

Icon = iconForm;

UpdateForm();

ShowComboBox();

}

private void UpdateForm()

{

database.DatabaseSQL().Close();

}

private void ShowComboBox() //Вывод должности в comboBox

{

database.DatabaseSQL().Open();

{

cbGroup.DataSource = database.TableFill("select ID\_Group from [dbo].[Group]", database.DatabaseSQL()).Tables[0];

cbGroup.DisplayMember = "ID\_Group";

cbTT.DataSource = database.TableFill("select TT\_Name from [dbo].[TT]", database.DatabaseSQL()).Tables[0];

cbTT.DisplayMember = "TT\_Name";

cbStatus.DataSource = database.TableFill("select Status\_Name from [dbo].[Status]", database.DatabaseSQL()).Tables[0];

cbStatus.DisplayMember = "Status\_Name";

}

database.DatabaseSQL().Close();

}

private void cbGroup\_SelectedIndexChanged(object sender, EventArgs e)

{

try

{

database.DatabaseSQL().Open();

{

SqlCommand cmd = new SqlCommand($"select ID\_Group from [dbo].[Group] where ID\_Group = '{cbGroup.Text}'", database.DatabaseSQL());

Program.ID\_Group = (int)cmd.ExecuteScalar();

}

database.DatabaseSQL().Close();

}

catch { }

}

private void cbTT\_SelectedIndexChanged(object sender, EventArgs e)

{

try

{

database.DatabaseSQL().Open();

{

SqlCommand cmd = new SqlCommand($"select ID\_TT from [dbo].[TT] where TT\_Name = '{cbTT.Text}'", database.DatabaseSQL());

Program.ID\_TT = (int)cmd.ExecuteScalar();

}

database.DatabaseSQL().Close();

}

catch { }

}

private void cbStatus\_SelectedIndexChanged(object sender, EventArgs e)

{

try

{

database.DatabaseSQL().Open();

{

SqlCommand cmd = new SqlCommand($"select ID\_Status from [dbo].[Status] where Status\_Name = '{cbStatus.Text}'", database.DatabaseSQL());

Program.ID\_Status = (int)cmd.ExecuteScalar();

}

database.DatabaseSQL().Close();

}

catch { }

}

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

{

database.DatabaseSQL().Open();

try

{

SqlCommand sqlCommand = new SqlCommand("Zayavka\_Insert", database.DatabaseSQL());

sqlCommand.CommandType = CommandType.StoredProcedure;

sqlCommand.Parameters.AddWithValue("@Group\_ID", Program.ID\_Group);

sqlCommand.Parameters.AddWithValue("@TT\_ID", Program.ID\_TT);

sqlCommand.Parameters.AddWithValue("@Status\_ID", Program.ID\_Status);

sqlCommand.Parameters.AddWithValue("@Data\_Podachi", mtbDataPodachi.Text);

sqlCommand.Parameters.AddWithValue("@Data\_Nachala\_Rabot", mtbDataNachala.Text);

sqlCommand.Parameters.AddWithValue("@Data\_Okonchania\_Rabot", mtbOkonchaniya.Text);

sqlCommand.Parameters.AddWithValue("@Data\_Zakritia\_Zayavki", mtbZakritiya.Text);

sqlCommand.Parameters.AddWithValue("@Spisok\_Rapot", tbSpisokRabot.Text);

sqlCommand.Parameters.AddWithValue("@Spisok\_Materialov", tbSpisokMaterialov.Text);

sqlCommand.Parameters.AddWithValue("@Kommentariy\_TT", tbKomentariiTT.Text);

sqlCommand.Parameters.AddWithValue("@Kommentariy\_Group", tbKimentariiGroup.Text);

sqlCommand.Parameters.AddWithValue("Zayavka\_Logical\_Delete", 0);

sqlCommand.ExecuteNonQuery();

MessageBox.Show("Заявка добавлена");

this.Hide();

}

catch (Exception ex)

{

MessageBox.Show("Заявка не добавлена " + ex.Message);

}

database.DatabaseSQL().Close();

}

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

{

database.DatabaseSQL().Open();

try

{

SqlCommand sqlCommand = new SqlCommand("Zayavka\_Update", database.DatabaseSQL());

sqlCommand.CommandType = CommandType.StoredProcedure;

sqlCommand.Parameters.AddWithValue("@ID\_Zayavka", Program.ID\_Zayavka);

sqlCommand.Parameters.AddWithValue("@Group\_ID", Program.ID\_Group);

sqlCommand.Parameters.AddWithValue("@TT\_ID", Program.ID\_TT);

sqlCommand.Parameters.AddWithValue("@Status\_ID", Program.ID\_Status);

sqlCommand.Parameters.AddWithValue("@Data\_Podachi", mtbDataPodachi.Text);

sqlCommand.Parameters.AddWithValue("@Data\_Nachala\_Rabot", mtbDataNachala.Text);

sqlCommand.Parameters.AddWithValue("@Data\_Okonchania\_Rabot", mtbOkonchaniya.Text);

sqlCommand.Parameters.AddWithValue("@Data\_Zakritia\_Zayavki", mtbZakritiya.Text);

sqlCommand.Parameters.AddWithValue("@Spisok\_Rapot", tbSpisokRabot.Text);

sqlCommand.Parameters.AddWithValue("@Spisok\_Materialov", tbSpisokMaterialov);

sqlCommand.Parameters.AddWithValue("@Kommentariy\_TT", tbKomentariiTT.Text);

sqlCommand.ExecuteNonQuery();

MessageBox.Show("Заявка изменена");

this.Hide();

}

catch (Exception ex)

{

MessageBox.Show("Заявка не изменена " + ex.Message);

}

database.DatabaseSQL().Close();

}

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

{

database.DatabaseSQL().Open();

try

{

if (Program.ID\_Dolj == 1)

{

SqlCommand sqlCommand = new SqlCommand("Zayavka\_Delete", database.DatabaseSQL());

sqlCommand.CommandType = CommandType.StoredProcedure;

sqlCommand.Parameters.AddWithValue("@ID\_Zayavka", Program.ID\_Zayavka);

sqlCommand.ExecuteNonQuery();

}

else

{

SqlCommand sqlCommand = new SqlCommand("Zayavka\_Logical\_Delete", database.DatabaseSQL());

sqlCommand.CommandType = CommandType.StoredProcedure;

sqlCommand.Parameters.AddWithValue("@ID\_Zayavka", Program.ID\_Zayavka);

sqlCommand.ExecuteNonQuery();

}

MessageBox.Show("Заявка удалена");

this.Hide();

}

catch (Exception ex)

{

MessageBox.Show("Заявка не удалена " + ex.Message);

}

database.DatabaseSQL().Close();

}

}

}

### Текст «ZastavkaForm.cs»

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Compo

{

public partial class Zastavka : Form

{

DataBase\_Configuration data = new DataBase\_Configuration();

public Zastavka()

{

this.TopMost = true; //Вызов окна поверх других

InitializeComponent();

Icon iconForm = new Icon(Application.StartupPath + "\\img\\compo2.ico");

Icon = iconForm;

this.StartPosition = FormStartPosition.CenterScreen; //Расположение окна по центру монитора

}

private void constate(bool value)

{

Action action = () =>

{

switch (value)

{

case (true):

SignInForm sign = new SignInForm();

Close();

sign.Show();

break;

case (false):

ConectionForm conection = new ConectionForm();

conection.Show();

timer1.Enabled = false;

this.Hide();

break;

}

};

Invoke(action);

}

public void Zastavka\_Load(object sender, EventArgs e)

{

data.conState += constate;

Thread thread = new Thread(data.Connection\_Check);

thread.Start();

}

private void Timer1\_Tick(object sender, EventArgs e)

{

if (progressBar1.Value != 17)

progressBar1.Value++;

else

this.Close();

}

}

}