Оглавление

[Описание предметной области 3](#_Toc70674756)

[Схема бд 9](#_Toc70674757)

[Код создания таблиц 11](#_Toc70674758)

[Запросы 13](#_Toc70674759)

[Представления 13](#_Toc70674760)

[Запросы на добавление 14](#_Toc70674761)

[Запросы на вывод 16](#_Toc70674762)

[Запросы на удаление 16](#_Toc70674763)

[Запросы на изменение 17](#_Toc70674764)

[Сложные запросы 18](#_Toc70674765)

[ХП 21](#_Toc70674766)

[Триггеры 22](#_Toc70674767)

[Приложение: 25](#_Toc70674768)

# Описание предметной области

Группы пользователей разрабатываемой информационной системы (ИС)

|  |  |
| --- | --- |
| № пп | Наименование пользователя |
| 1 | Сотрудник банка |
| 2 | Клиент |
| 3 | Администратор |

Функции групп пользователей

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № пп | Выполняемая функция | Входные данные | Выходные данные | Функции, которые должны быть реализованы в ИС |
| ***Сотрудник банка*** | | | | |
| 1 | Добавление нового клиента | Данные о клиенте | Изменение БД | Добавление нового клиента |
| 2 | Изменение данных о клиенте | id клиента, данные о клиенте | Изменение БД | Изменение данных о клиенте |
| 3 | Просмотр данных о клиенте и о его кредитах / получение полной выписки о счете клиента | id клиента | Данные о клиенте и о его кредитах | Вывод данных о клиенте и о его кредитах(выборочно) или вывод полной информации о клиенте |
| 4 | Добавление запроса на кредит | id клиента, сумма кредита | Изменение БД | Добавление кредита в список |
| 5 | Удаление кредита клиента | id клиента, номер счета | Изменение БД | Удаление кредита |
| ***Клиент*** | | | | |
| 1 | Авторизация | Логин, пароль | Успешная авторизация / Сообщение об ошибке | Авторизация и разрешение на определенные запросы |
| 2 | Вывод информации о кредите | id, запрос | Информация по запросы | Вывод информации по запросу пользователя |
| 3 | Запрос кредита | id, сумма кредита | Изменение БД | Добавить запрос на кредит |
| 4 | Получение выписки по счету клиента | id | Полная информация о клиенте | Вывод всей информации о клиенте |
| 5 | Вывод / пополнение счета | id, номер счета, сумма пополнения / вывода | Изменение БД | Изменение баланса при выводе / пополнении |
| 6 | Удаление аккаунта | id | Изменение БД | Удаление всех данных о клиенте |
| ***Администратор*** | | | | |
| 1 | Удаление аккаунта клиента | id клиента | Изменение БД | Удаление всех данных о клиенте |
| 2 | Удаление аккаунта сотрудника банка | id сотрудника | Изменение БД | Удаление всех данных о сотруднике |
| 3 | Добавление сотрудника банка | id сотрудника, данные о сотруднике | Изменение БД | Добавление нового сотрудника |
| 4 | Подтверждение кредита | Номер кредита | Изменение БД | Добавление денег на баланс клиента |
| 5 | Просмотр информации о клиенте / сотруднике | id клиента / сотрудника | Информация о клиенте / сотруднике | Вывод данных о клиенте / сотруднике |
| 6 | Изменение данных сотрудника | id сотрудника, новые данные | Изменение БД | Изменение данных о сотруднике |
| 7 | Подсчет прибыли банка |  | Вывод данных | Подсчет прибыли |

**Хранимые данные**

|  |  |  |  |
| --- | --- | --- | --- |
| № пп | Хранимые данные | Пользователи, которым разрешен доступ | Ограничения по типу и значению |
| **Клиент (Client)** | | | |
| 1 | Id клиента (ClientID) | Все | int, not null, PRIMARY KEY |
| 2 | ФИО клиента (FullName) | nvarchar(50), not null |
| 3 | Дата рождения (Born) | date, not null |
| 4 | Адрес (Address) | nvarchar(50), not null |
| 5 | Страна (Country) | nvarchar(50), not null |
| 6 | Город (City) | nvarchar(50), not null |
| 7 | серия и номер паспорта(passport) | Int(10), not null |
| 8 | ИНН (INN) | Int(12), not null |
| 9 | СНИЛС (SNILLS) | Int(9), not null |
| 10 | Номер телефона (Phone) | nvarchar(20), not null |
| 11 | Дата присоединения (joinDate) | datetime |
| 12 | UserID |  | int, not null |
| **Работник (Employee)** | | | |
| 1 | Id работника (EmployeeID) | Админ, сотрудники банка | int, not null, PRIMARY KEY |
| 2 | ФИО работника (FullName) | nvarchar(50), not null |
| 3 | Дата рождения (Born) | date, not null |
| 4 | Адрес (Address) | nvarchar(50), not null |
| 5 | Страна (Country) | nvarchar(50), not null |
| 6 | Город (City) | nvarchar(50), not null |
| 7 | серия и номер паспорта(passport) | Int(10), not null |
| 8 | ИНН (INN) | Int(12), not null |
| 9 | СНИЛС (SNILLS) | Int(9), not null |
| 10 | Номер телефона (Phone) | nvarchar(20), not null |
| 11 | Зарплата (wage) | float |
| 12 | UserID | Int, not null |
| **Аккаунт (Account)** | | | |
| 1 | ID пользователя (IdUser) | Все | Int, not null, PRIMARY KEY |
| 2 | Имя пользователя (username) | nvarchar(20), not null, UNIQUE |
| 3 | Пароль (password) | nvarchar(20), not null |
| **Кредит (Credit)** | | | |
| 1 | ID клиента (ClientID) | Все | Int, not null |
| 2 | ID Кредита (CreditID) | Int, not null, PRIMARY KEY |
| 3 | Дата открытия (Date\_Begin) | Datetime, not null |
| 4 | Дата закрытия (Date\_close) | Datetime |
| 5 | Основной долг (Main\_debt) | Float, not null |
| 6 | Проценты к выплате (interest\_payable)) | Float, not null |
| 7 | Процентная ставка (Interest\_rate) | Float, not null |
| 8 | Подтвержден (Approved) | nvarchar(3), not null |
| 9 | Подтвержден кем (Approved\_by) | Int |
| **Банковский счет (BankAccount)** | | | |
| 1 | idAccount | Все | Int, not null, PRIMARY KEY |
| 2 | Текущий баланс (CurrentBalance) | Float, not null |
| 3 | ClientID | int |
| **История операций (OperationsHistory)** | | | |
| 1 | idAccount | Все | Int, not null |
| 2 | OperationID | Bigint, not null, PRIMARY KEY |
| 3 | Доход/Расход (IncomeExpenese) | Float, not null |
| 4 | Дата операции (Date) | Datetime, not null |

# Схема бд

В предлагаемой базе данных существуют следующие сущности:

* Клиент (Client)
* Аккаунт (Account)
* Кредит (Credit)
* Сотрудник (Employee)
* Банковский счет (BankAccount)
* История операций (OperationsHistory)

Клиент – имеет уникальный идентификатор и информацию о таких данных, как ФИО, адрес, страна, город, дату рождения, id пользователя, номер телефона и дату присоединения.

Аккаунт – имеет уникальный идентификатор, а также информацию о логине и пароле аккаунта.

Кредит – имеет уникальный идентификатор и хранит информацию о кредите (id клиента, id кредита, дату открытия и закрытия, сумму основного долга, сумму процентов к погашению, процентную ставку, статус подтверждения и информацию о сотруднике, который подтвердил кредит)

Сотрудник – имеет уникальный идентификатор, а также информацию о сотруднике (ФИО, дата рождения, адрес, страна, город, телефон, заработная плата, id пользователя)

Банковский счет – имеет уникальный идентификатор, а также информацию о текущем балансе и id клиента, которому принадлежит счет.

История операций – имеет уникальный идентификатор, а также информацию о доходе и расходе и id клиента, которому эта операция принадлежит.



# Код создания таблиц

CREATE TABLE [dbo].[Credit](  
 [id] [int] IDENTITY(1,1) NOT NULL,  
 [BankAccount\_id] [int] NOT NULL,  
 [Date\_Begin] [datetime] NOT NULL,  
 [Date\_Close] [datetime] NOT NULL,  
 [Main\_debt] [money] NOT NULL,  
 [interest\_payable] [money] NOT NULL,  
 [Interest\_rate] [int] NOT NULL,  
 [Approved] [int] NOT NULL,  
 [Employee\_id] [int] NULL,  
 CONSTRAINT [PK\_Credit] PRIMARY KEY CLUSTERED   
(  
 [id] ASC  
)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]  
) ON [PRIMARY]

CREATE TABLE [dbo].[OperationsHistory](  
 [id] [int] IDENTITY(1,1) NOT NULL,  
 [Account\_id] [int] NOT NULL,  
 [IncomeExpenese] [money] NOT NULL,  
 [Date] [datetime] NOT NULL,  
 CONSTRAINT [PK\_OperationsHistory] PRIMARY KEY CLUSTERED   
(  
 [id] ASC  
)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]  
) ON [PRIMARY]

CREATE TABLE [dbo].[Client](  
 [id] [int] IDENTITY(1,1) NOT NULL,  
 [User\_id] [int] NOT NULL,  
 [FullName] [nchar](50) COLLATE Cyrillic\_General\_CI\_AS NOT NULL,  
 [Born] [date] NOT NULL,  
 [Adress] [nvarchar](20) COLLATE Cyrillic\_General\_CI\_AS NOT NULL,  
 [Country] [nvarchar](20) COLLATE Cyrillic\_General\_CI\_AS NOT NULL,  
 [City] [nvarchar](20) COLLATE Cyrillic\_General\_CI\_AS NOT NULL,  
 [Passport] [nchar](10) COLLATE Cyrillic\_General\_CI\_AS NOT NULL,  
 [INN] [nchar](10) COLLATE Cyrillic\_General\_CI\_AS NOT NULL,  
 [SNILLS] [nchar](11) COLLATE Cyrillic\_General\_CI\_AS NOT NULL,  
 [Phone] [nchar](11) COLLATE Cyrillic\_General\_CI\_AS NOT NULL,  
 [joinDate] [datetime] NOT NULL,  
 CONSTRAINT [PK\_Client] PRIMARY KEY CLUSTERED   
(  
 [id] ASC  
)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]  
) ON [PRIMARY]

CREATE TABLE [dbo].[BankAccount](  
 [id] [int] IDENTITY(1,1) NOT NULL,  
 [CurrentBalance] [money] NOT NULL,  
 [Client\_id] [int] NOT NULL,  
 CONSTRAINT [PK\_BankAccount] PRIMARY KEY CLUSTERED   
(  
 [id] ASC  
)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]  
) ON [PRIMARY]

CREATE TABLE [dbo].[Employee](  
 [id] [int] IDENTITY(1,1) NOT NULL,  
 [User\_id] [int] NOT NULL,  
 [FullName] [nchar](50) COLLATE Cyrillic\_General\_CI\_AS NOT NULL,  
 [Born] [date] NOT NULL,  
 [Adress] [nvarchar](20) COLLATE Cyrillic\_General\_CI\_AS NOT NULL,  
 [Country] [nvarchar](20) COLLATE Cyrillic\_General\_CI\_AS NOT NULL,  
 [City] [nvarchar](20) COLLATE Cyrillic\_General\_CI\_AS NOT NULL,  
 [Passport] [nchar](10) COLLATE Cyrillic\_General\_CI\_AS NOT NULL,  
 [INN] [nchar](10) COLLATE Cyrillic\_General\_CI\_AS NOT NULL,  
 [SNILLS] [nchar](11) COLLATE Cyrillic\_General\_CI\_AS NOT NULL,  
 [Phone] [nchar](11) COLLATE Cyrillic\_General\_CI\_AS NOT NULL,  
 [Mage] [money] NOT NULL,  
 CONSTRAINT [PK\_Employee] PRIMARY KEY CLUSTERED   
(  
 [id] ASC  
)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]  
) ON [PRIMARY]

CREATE TABLE [dbo].[Users](  
 [id] [int] IDENTITY(1,1) NOT NULL,  
 [username] [nchar](20) COLLATE Cyrillic\_General\_CI\_AS NOT NULL,  
 [password] [nchar](64) COLLATE Cyrillic\_General\_CI\_AS NOT NULL,  
 [Role] [nvarchar](20) COLLATE Cyrillic\_General\_CI\_AS NOT NULL,  
 CONSTRAINT [PK\_Users] PRIMARY KEY CLUSTERED   
(  
 [id] ASC  
)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]  
) ON [PRIMARY]

# Запросы

## Представления

1. Создать представление с Id клиентов, у которых есть незакрытые и подтвержденные кредиты.

SELECT c.FullName, c.Passport

FROM dbo.Client AS c INNER JOIN

dbo.BankAccount AS b ON b.Client\_id = c.id INNER JOIN

dbo.Credit AS cr ON cr.BankAccount\_id = b.id

WHERE (cr.Date\_Close < CURRENT\_TIMESTAMP) AND (cr.Approved = 1)

1. Создать представление, которое выводит операции по счету и информацию о кредитах.

SELECT 'Кредит' AS Name, cr.Main\_debt AS Sum, cr.Date\_Begin AS Date, cr.Approved, cl.Passport

FROM Client cl JOIN

BankAccount ba ON ba.Client\_id = cl.id JOIN

Credit cr ON cr.BankAccount\_id = ba.id AND cr.Approved = 1)

UNION ALL

(SELECT 'Операция', oh.IncomeExpenese, oh.Date, ba.CurrentBalance, cl.Passport

FROM Client cl JOIN

BankAccount ba ON ba.Client\_id = cl.id JOIN

OperationsHistory oh ON oh.Account\_id = ba.id))

1. Создать представление с количеством клиентов по городам в отсортированном виде в алфавитном порядке.

SELECT 'Кредит' AS Name, cr.Main\_debt AS Sum, cr.Date\_Begin AS Date, cr.Approved, cl.Passport

FROM Client cl JOIN

BankAccount ba ON ba.Client\_id = cl.id JOIN

Credit cr ON cr.BankAccount\_id = ba.id AND cr.Approved = 1)

UNION ALL

(SELECT 'Операция', oh.IncomeExpenese, oh.Date, ba.CurrentBalance, cl.Passport

FROM Client cl JOIN

BankAccount ba ON ba.Client\_id = cl.id JOIN

OperationsHistory oh ON oh.Account\_id = ba.id))

## Запросы на добавление

1. Добавить клиента и его банковский аккаунт

begin transaction;

insert into Users values (

'1', '1'

);

insert into Client values (

(select top 1 id from Users order by id desc)

, 'name'

, 'born'

, 'adress'

, 'country'

, 'city'

, 'passport'

, 'inn'

, 'snils'

, 'phone'

, 'joindate'

);

insert into BankAccount values (

98765

, (select top 1 id from Client order by id desc)

);

1. Добавить банковский аккаунт для клиента

insert into BankAccount values (

98765

, (select top 1 id from Client order by id desc)

)

1. Добавить запрос на кредит

insert into Credit values (

(select id from BankAccount where Client\_id =

(select id from Client where Passport = ''))

, 'Date\_Begin'

, 'Date\_Close'

, 'sumCredit'

, 'percSum'

, percents

, 0

, (select id from Employee where Passport = '')

)

1. Добавить сотрудника банка

begin transaction;

insert into Users values (

'1', '1'

);

insert into Client values (

(select top 1 id from Users order by id desc)

, 'name'

, 'born'

, 'adress'

, 'country'

, 'city'

, 'passport'

, 'inn'

, 'snils'

, 'phone'

, 'zp'

);

1. Добавить новую операцию в историю операций.

insert into OperationsHistory values (

acc\_id

, '6546'

, GETDATE()

)

1. Добавить новый аккаунт для клиента или сотрудника банка.

insert into Users values (

'1', '1'

)

## Запросы на вывод

1. Вывести полные имена и заработную плату сотрудников банка. Отсортировать по зарплате.

select FullName, Mage

from Employee

order by Mage asc

1. Вывести все неодобренные кредиты и дату открытия. Отсортировать по возрастанию даты открытия.

select Main\_debt, Date\_Begin

from Credit

where Approved = 2

1. Вывести все одобренные кредиты и дату открытия, отсортировать по дате открытия.

select Main\_debt, Date\_Begin

from Credit

where Approved = 1

1. Вывести полные имена и дату присоединения всех клиентов, отсортировать по дате присоединения.

select FullName, joinDate

from Client

order by joinDate asc

## Запросы на удаление

1. Удалить кредит клиента со статусом 0.

delete from Credit

where BankAccount\_id = (select id from BankAccount where Client\_id =

(select id from Client where Passport = ''))

and Approved = 0

1. Удалить аккаунт клиента

delete from Users

where username = (select username from Users u join Client c on c.User\_id = u.id where c.Passport = '')

1. Удалить аккаунт сотрудника

delete from Users

where username = (select username from Users u join Employee c on c.User\_id = u.id where c.Passport = '')

## Запросы на изменение

1. Изменить паспортные данные клиента.

update Client set Passport = ''

where Passport = ''

1. Изменить адрес клиента.

update Client set Adress = ''

where Passport = ''

1. Изменить номер телефона клиента.

update Client set Phone = ''

where Passport = ''

1. Изменить паспортные данные работника.

update Employee set Passport = ''

where Passport = ''

1. Изменить адрес работника.

update Employee set Adress = ''

where Passport = ''

1. Изменить номер телефона работника.

update Employee set Phone = ''

where Passport = ''

1. Изменить заработную плату работника.

update Employee set Mage = ''

where Passport = ''

1. Изменить статус подтверждения кредита.

update Credit set Approved = 1

where BankAccount\_id = (select id from BankAccount where Client\_id =

(select id from Client where Passport = ''))

## Сложные запросы

1. Вывести полные имена клиентов и полную информацию о их незакрытых кредитах. Отсортировать по дате открытия.

select FullName

, cr.Date\_Begin

, cr.Date\_Close

, cr.Main\_debt as sum\_credit

, cr.Interest\_rate as percentage

from Client cl

join BankAccount ba

on cl.id = ba.Client\_id

join Credit cr

on cr.BankAccount\_id = ba.id

where cr.Date\_Close < GETDATE()

order by cr.Date\_Begin asc

1. Вывести информацию о прибыли банка за все время по месяцам. Отсортировать по дате в порядке убывания.

select sum(interest\_payable) as debit

, MONTH(Date\_Begin) as month

from Credit

group by MONTH(Date\_Begin)

order by MONTH(Date\_Begin) desc

1. Для подтвержденных кредитов вывести паспорт клиента, дату открытия и информацию о сотруднике, который подтвердил кредит (полное имя, паспорт , телефон). Отсортировать по дате открытия кредита.

select cl.Passport

, cr.Date\_Begin

, e.FullName

, e.Passport

, e.Phone

from Credit cr

join BankAccount ba

on ba.id = cr.BankAccount\_id

join Client cl

on cl.id = ba.Client\_id

join Employee e

on e.id = cr.Employee\_id

where cr.Approved = 1

order by cr.Date\_Begin asc

1. Вывести паспорта клиентов и средние по месяцам доходы, и расходы клиентов

select c.Passport

, avg(oh.IncomeExpenese)

, MONTH(oh.Date)

from Client c

join BankAccount ba

on c.id = ba.Client\_id

join OperationsHistory oh

on oh.Account\_id = ba.id

group by MONTH(oh.Date), c.Passport

1. Вывести все кредиты, подтвержденные конкретным работником.

select Main\_debt, Date\_Begin

from Credit c

where Approved = 1

and c.Employee\_id = 1

1. Вывести общую сумму подтвержденных кредитов по городам, а также их количество. Отсортировать по общей сумме кредитов.

select sum(cr.Main\_debt) as sum

, cl.City

, count(cr.Main\_debt) as count

from Credit cr

join BankAccount ba

on ba.id = cr.BankAccount\_id

join Client cl

on cl.id = ba.Client\_id

where cr.Approved = 1

group by cl.City

order by sum(cr.Main\_debt) desc

1. Вывести доход банка с каждого клиента

select sum(cr.interest\_payable)

, cl.Passport

from Credit cr

join BankAccount ba

on ba.id = cr.BankAccount\_id

join Client cl

on cl.id = ba.Client\_id

group by cl.Passport

1. Для конкретного клиента вывести текущий баланс и сумму всех кредитов.

(select 'sumMoney'

, sum(ba.CurrentBalance)

from BankAccount ba

full join Client cl

on cl.id = ba.Client\_id

where cl.Passport = '1256987456')

union all

(select 'sumCredit'

, sum(cr.Main\_debt)

from BankAccount ba

full join Client cl

on cl.id = ba.Client\_id

full join Credit cr

on cr.BankAccount\_id = ba.id

where cl.Passport = '1256987456')

1. Вывести имена и паспорта сотрудников и кол-во подтвержденных кредитов.

select count(cr.Approved)

, e.Passport

, e.FullName

from Employee e

join Credit cr

on cr.Employee\_id = e.id

group by e.Passport, e.FullName

1. Вывести максимальную сумму подтвержденного кредита, а также соответствующие имена клиентов, которым принадлежат данные кредиты.

select top 1 cr.Main\_debt

, cl.Passport

, cl.FullName

from Credit cr

join BankAccount ba

on ba.id = cr.BankAccount\_id

join Client cl

on cl.id = ba.Client\_id

where cr.Approved = 1

group by cr.Main\_debt, cl.Passport, cl.FullName

order by cr.Main\_debt desc

## ХП

1. Создать хранимую процедуру, которая по паспорту будет выводить баланс и кредиты

ALTER PROCEDURE [dbo].[credit\_view]

@Passport [varchar](10)

WITH EXECUTE AS CALLER

AS

((SELECT ''Кредит'' AS Name, cr.Main\_debt AS Sum, cr.Date\_Begin AS Date, cr.Approved

FROM Client cl JOIN

BankAccount ba ON ba.Client\_id = cl.id JOIN

Credit cr ON cr.BankAccount\_id = ba.id

WHERE Passport = @Passport

AND cr.Approved = 1)

UNION ALL

(SELECT ''Операция'', oh.IncomeExpenese, oh.Date, ba.CurrentBalance

FROM Client cl JOIN

BankAccount ba ON ba.Client\_id = cl.id JOIN

OperationsHistory oh ON oh.Account\_id = ba.id

WHERE Passport = @Passport))

1. Создать хранимую процедуру, которая будет выводить паспорт клиента, дату открытия и информацию о сотруднике, который подтвердил кредит

ALTER PROCEDURE [dbo].[credit\_info]

WITH EXECUTE AS CALLER

AS

select cl.Passport

, cr.Date\_Begin

, e.FullName

, e.Passport

, e.Phone

from Credit cr

join BankAccount ba

on ba.id = cr.BankAccount\_id

join Client cl

on cl.id = ba.Client\_id

join Employee e

on e.id = cr.Employee\_id

where cr.Approved = 1

order by cr.Date\_Begin asc

1. Создать хранимую процедуру, которая будет выводить общий доход банка

ALTER PROCEDURE [dbo].[debit\_proc]

WITH EXECUTE AS CALLER

AS

select sum(cr.interest\_payable)

from Credit cr

## Триггеры

1. Создать триггер, который не будет добавлять клиентов, которым нет 18

create trigger [dbo].[Date\_not\_18]

on Client

after insert, update

as

begin

set nocount on;

declare @date date

set @date = (select Born from inserted)

if @date >= DATEADD(YEAR, 18, GETDATE())

begin

Rollback Tran

raiserror('Вам должно быть больше 18 лет', 16, 10)

end

end

1. Триггер, который не дает изменить зп на отрицательную

create trigger [dbo].[Sum\_credit\_Not\_Null]

on Employee

after insert, update

as

begin

set nocount on;

declare @price int

set @price = (select Mage from inserted)

declare @id int

set @id = (select id from inserted)

if @price <= 0

begin

Rollback Tran

raiserror('Зарплата не может быть меньше 0', 16, 10)

end

else

begin

update Employee

set Mage = @price

where @price > 0 and Employee.id = @id

end

end

1. Триггер, который не позволяет удалить непогашенный кредит

create trigger [dbo].[Data\_Check]

on [dbo].Credit

after delete

as

begin

set nocount on;

declare @date datetime

set @date = (select Date\_Close from deleted)

declare @app int

set @app = (select Approved from deleted)

if @date >= GETDATE() and @app != 0 begin

Rollback Tran

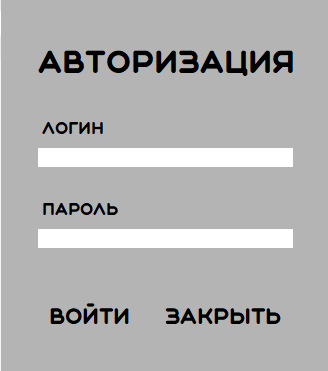
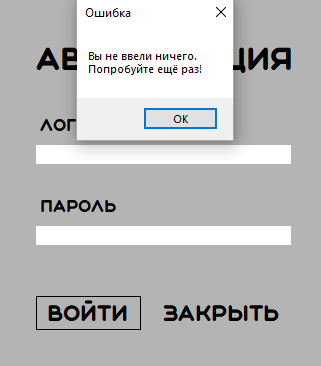
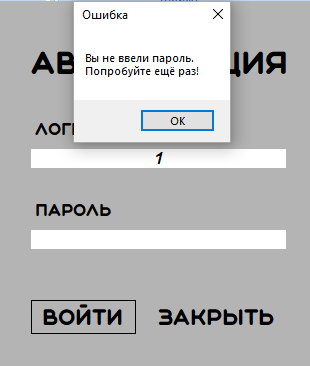
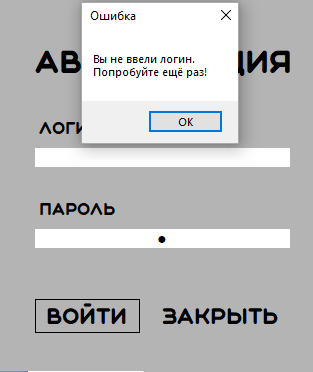
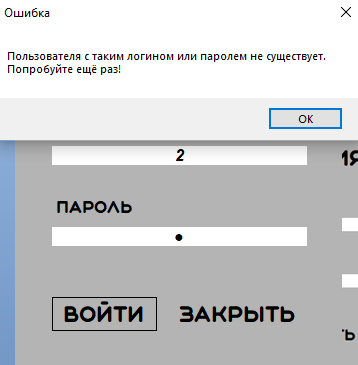
raiserror('Нельзя удалить непогашенный кредит', 16, 10)

end

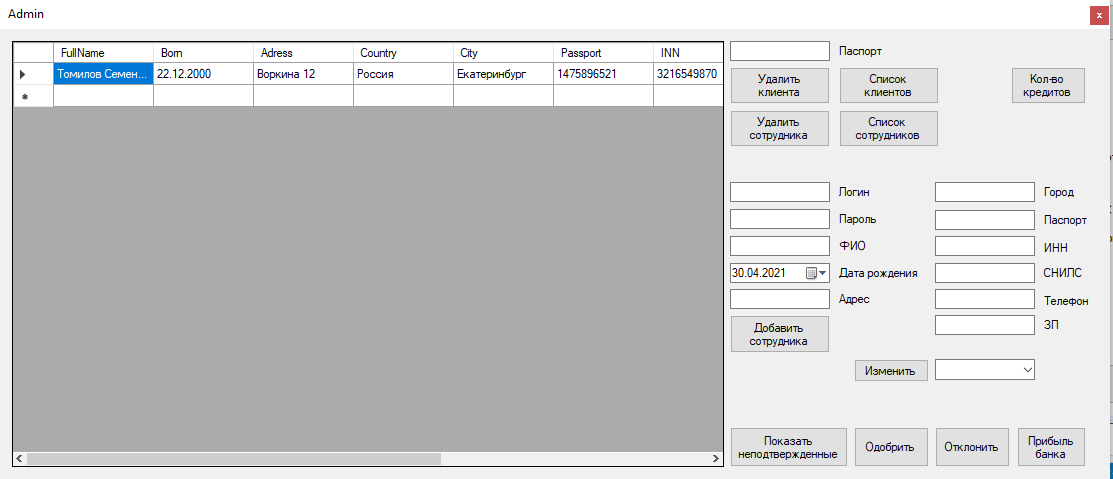
end

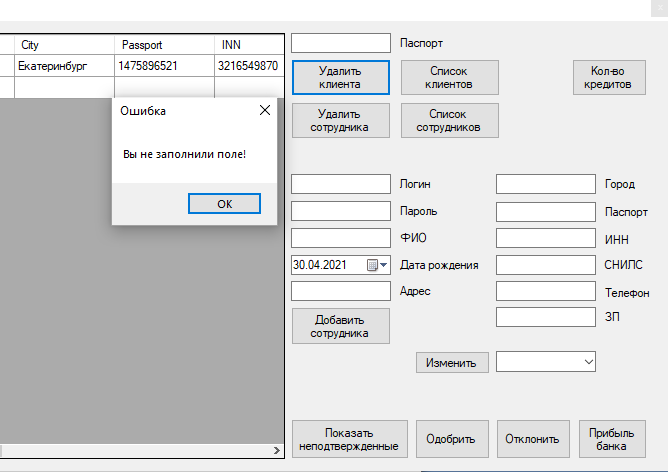
# Приложение:

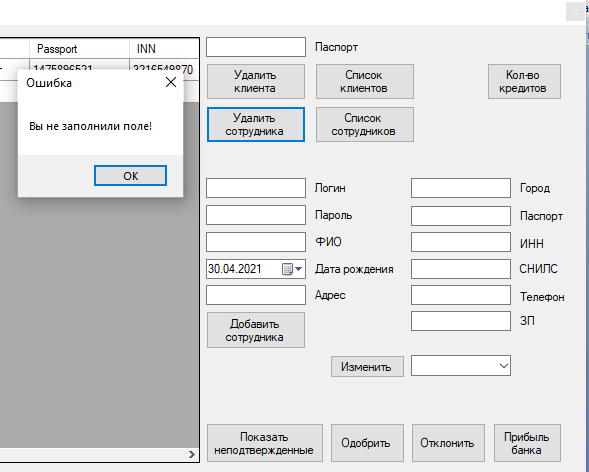
**Авторизация:**

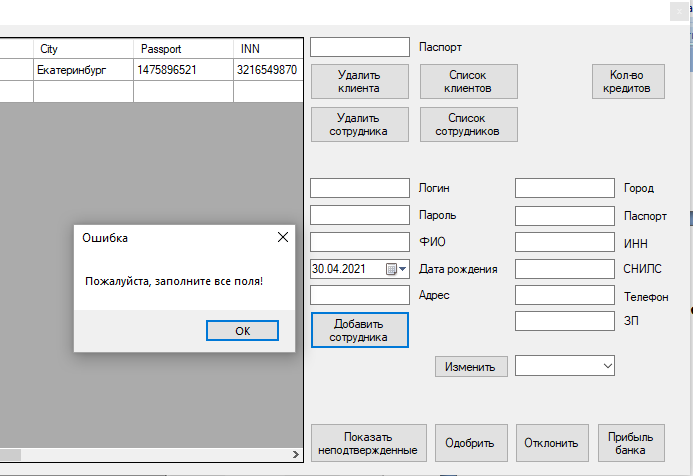
**    **

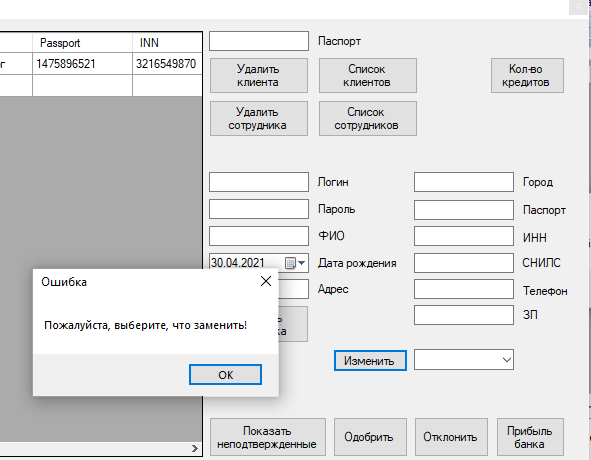
**Окно администратора**

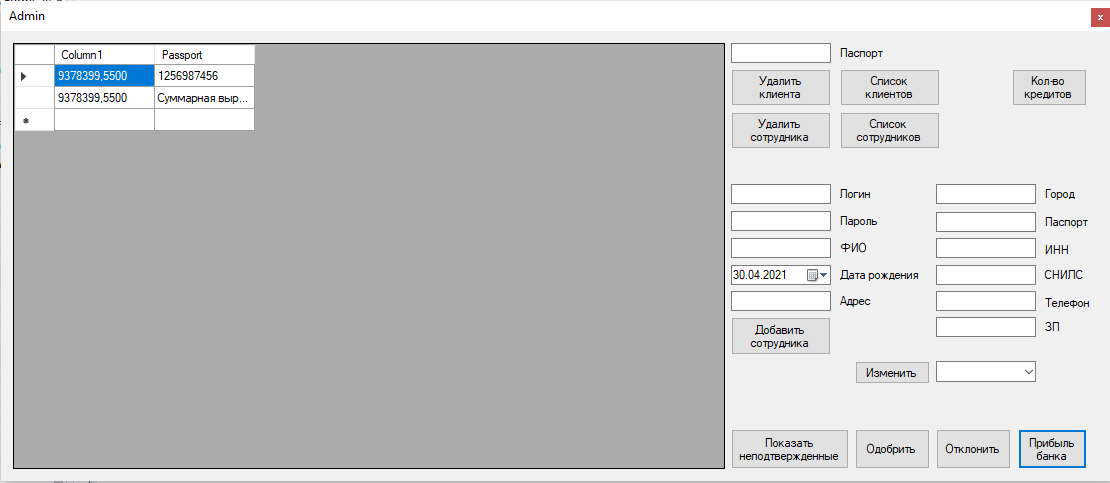
****

****

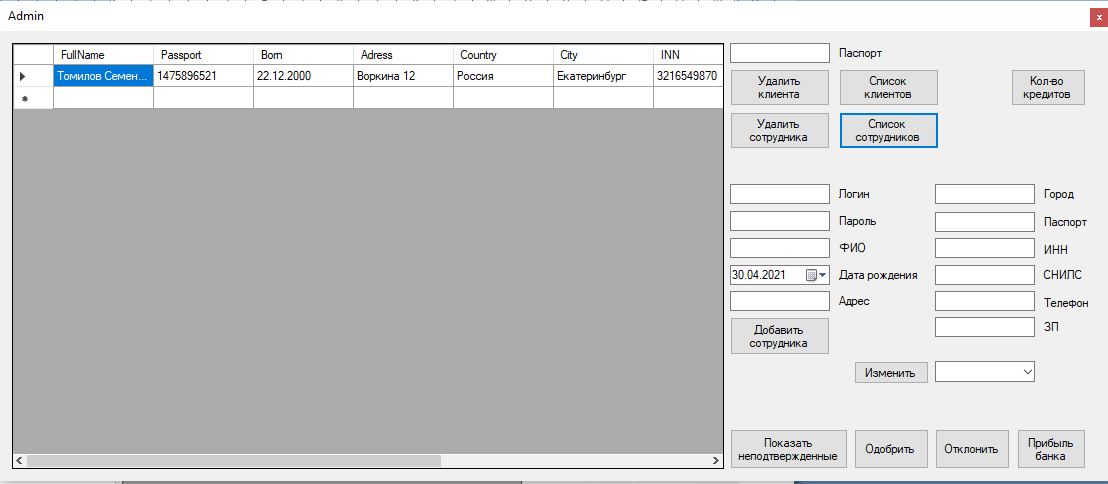
****

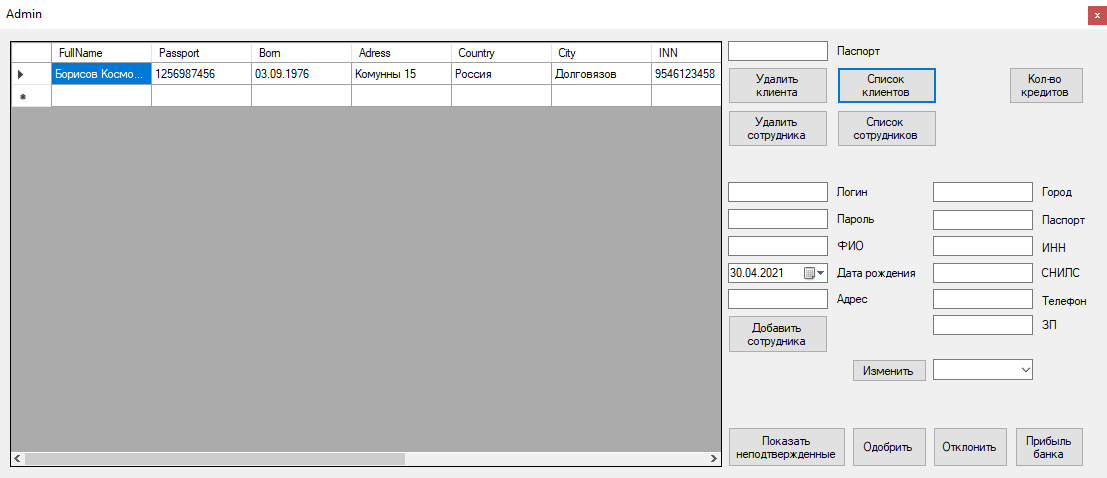
****

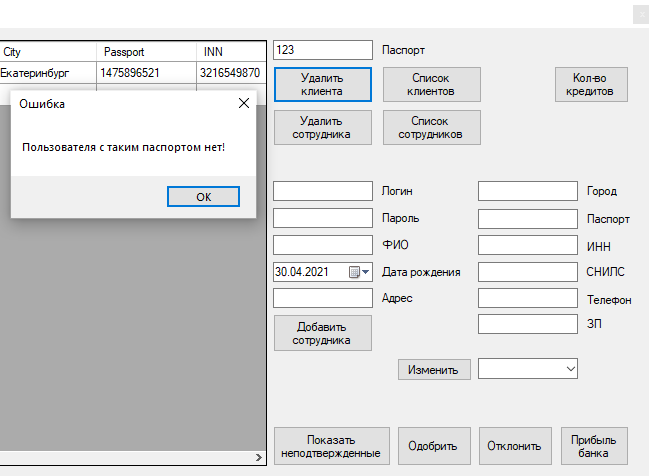
****

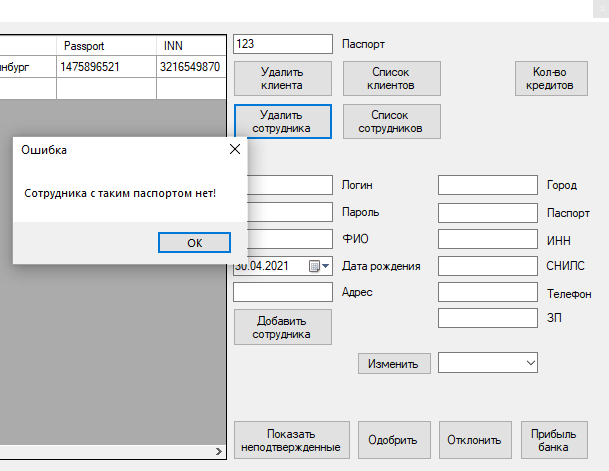
****

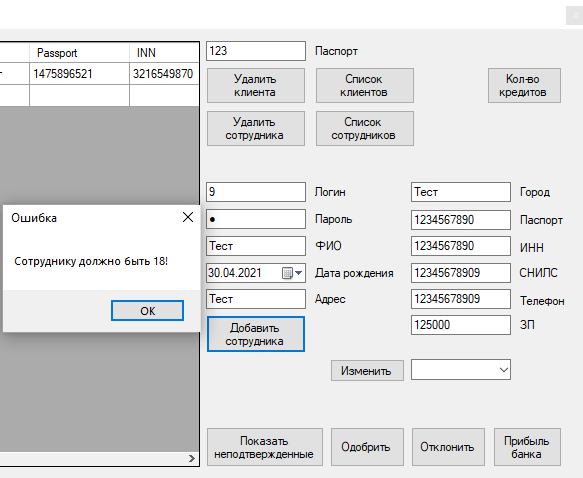
****

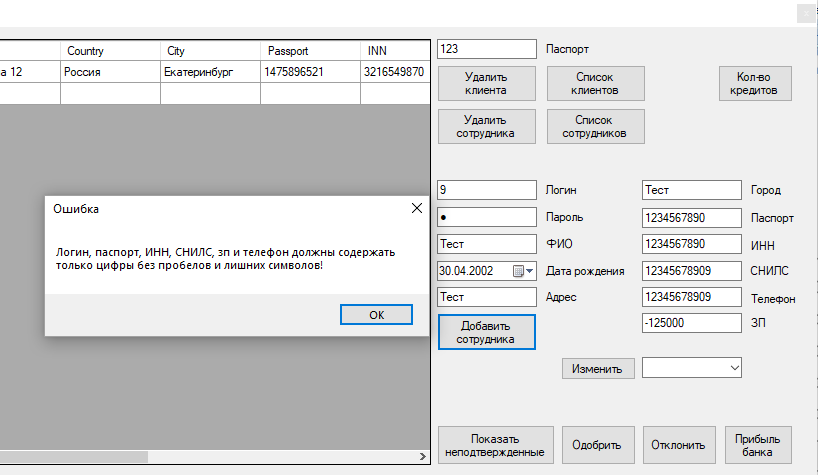
****

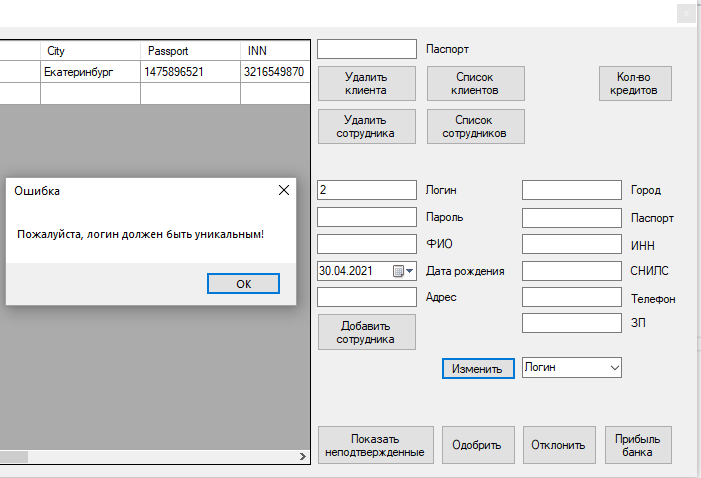
****

****

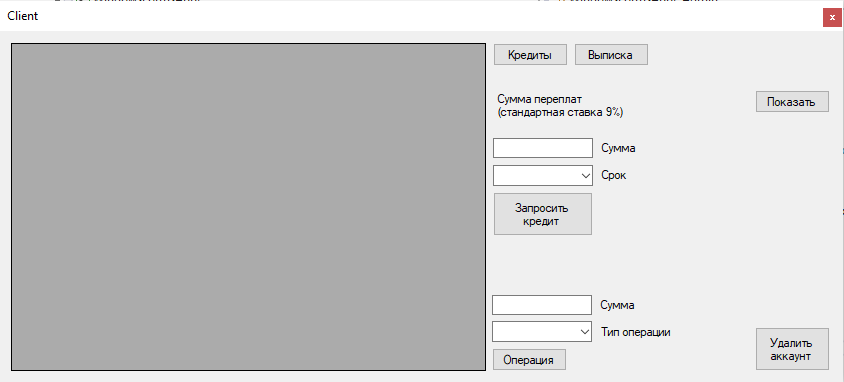
****

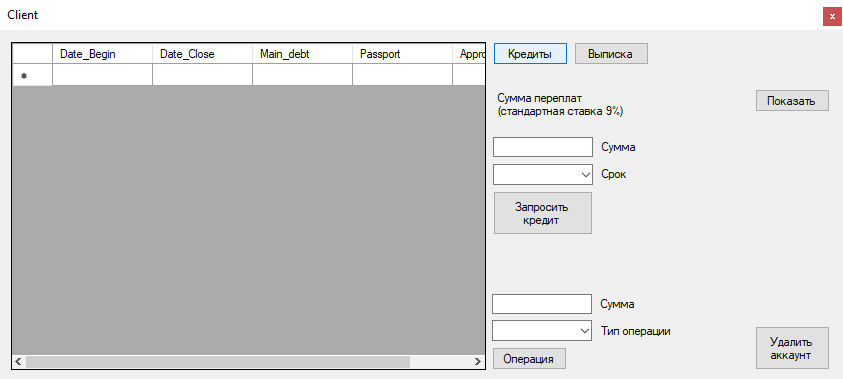
****

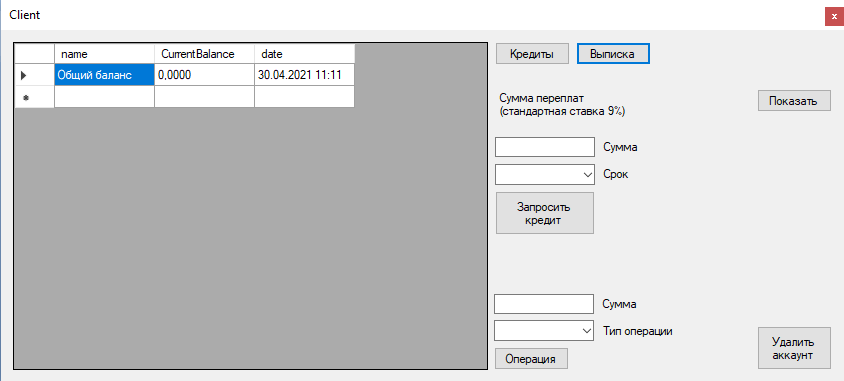
****

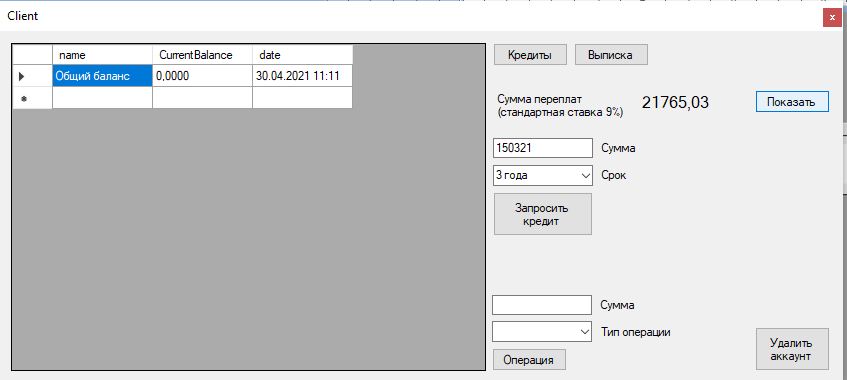
****

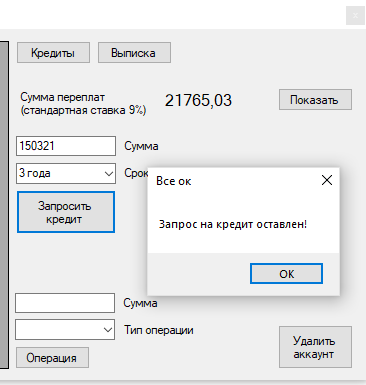
**Окно клиента**

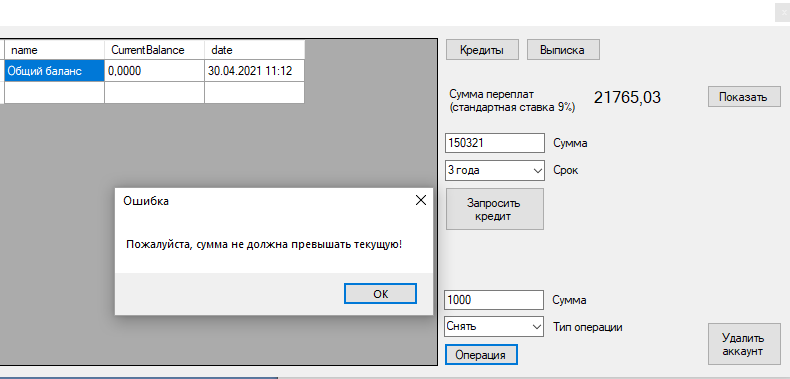
****

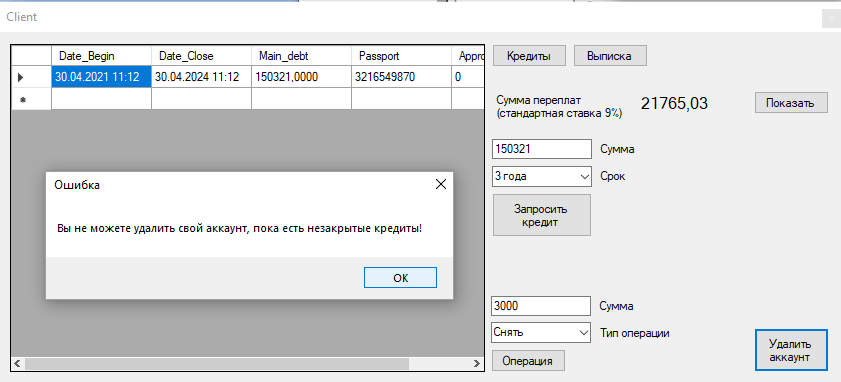
****

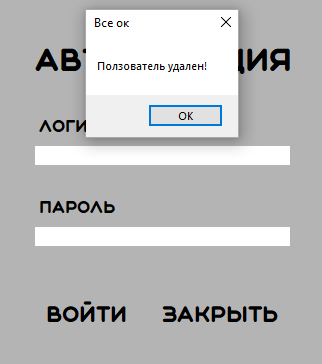
****

****

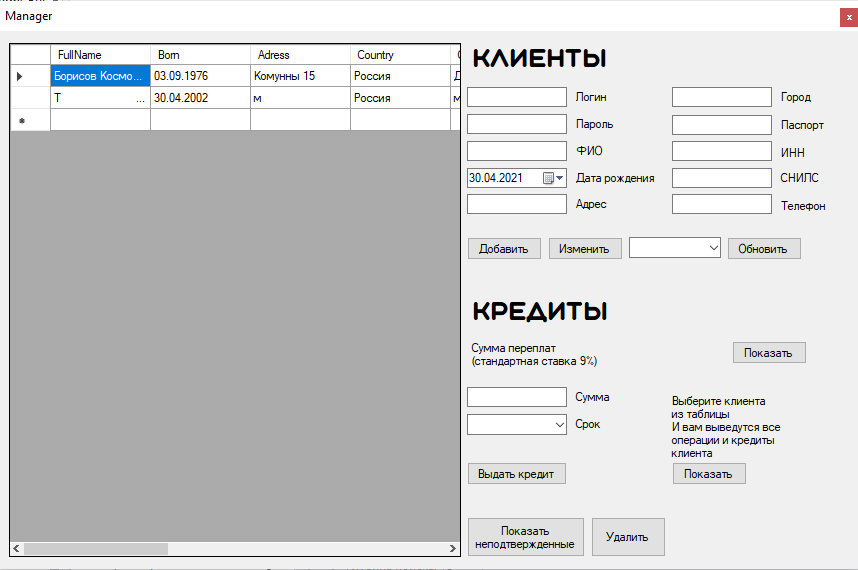
****

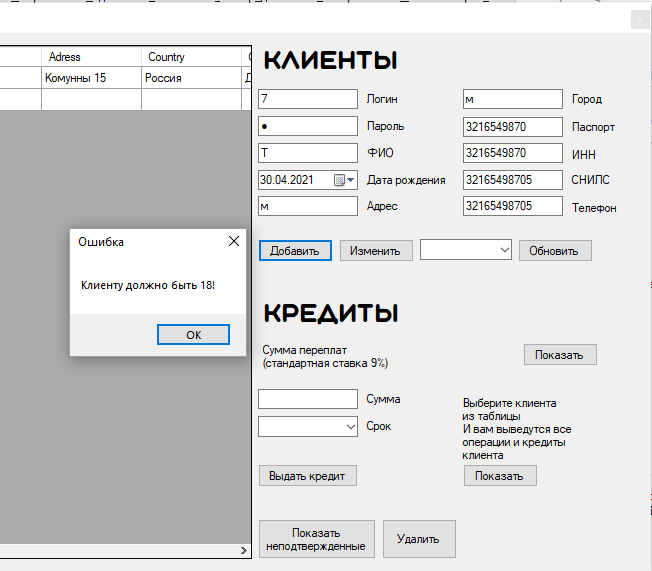
****

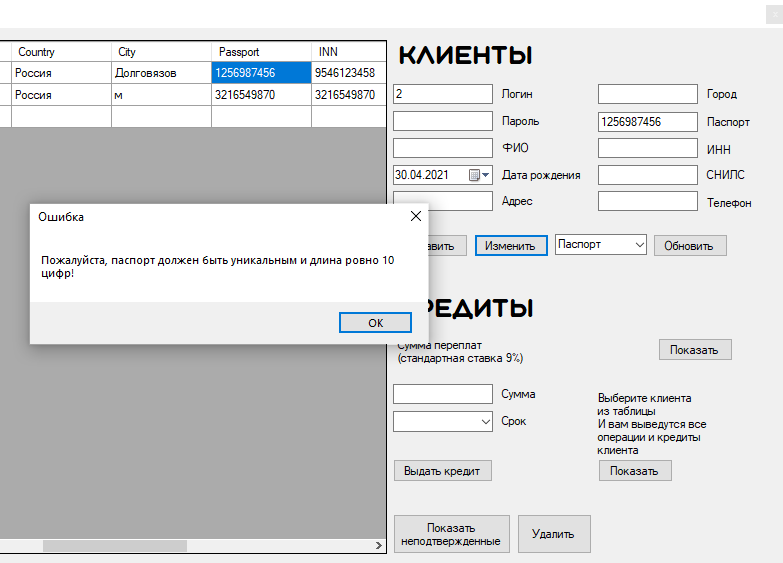
****

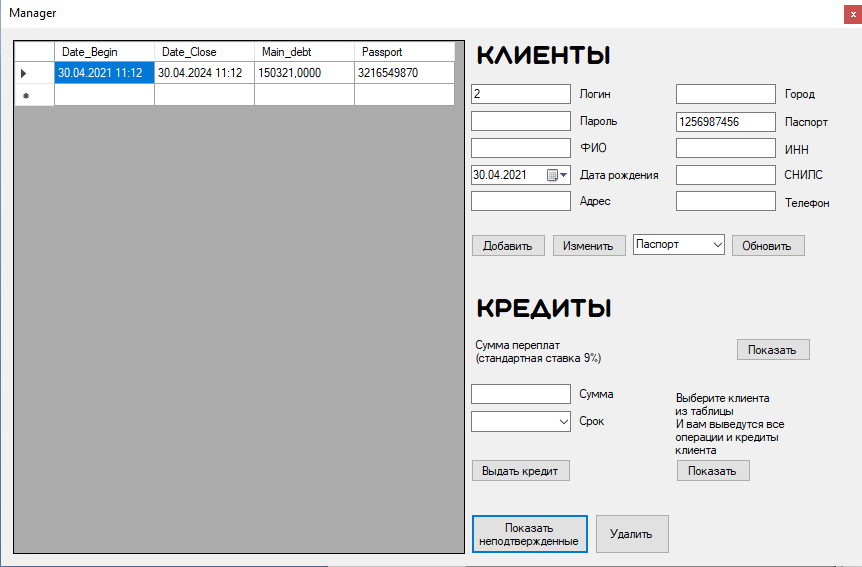
****

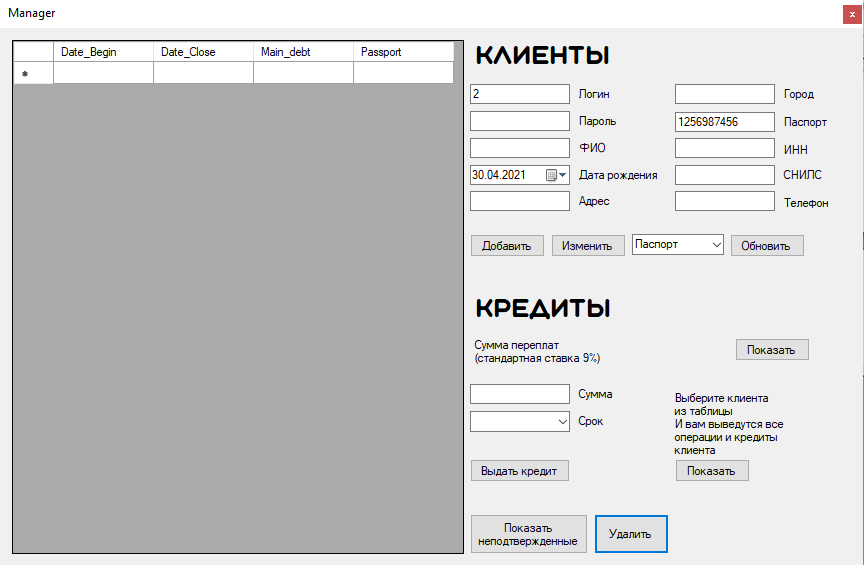
**Окно менеджера**

****

****

****

****

****