САНКТ-ПЕТЕРБУРСКОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ПРОФЕССИОНАЛЬНОЕ ОБРАЗОВАТЕЛЬНОЕ УТВЕРЖДЕНИЕ <<КОЛЛЕДЖА МЕТРОПОЛИТЕНА И ЖЕЛЕЗНОДОРОЖНОГО ТРАНСПОРТА>>

**Санкт-Петербург**

**2022 год**

**Выполнил обучающийся гр. № 302**

**Мелехов М.В.**

**ОТЧЕТ ПО КУРСОВОЙ РАБОТЕ**

Тема: Разработка приложения для банкомата в среде Microsoft Visual Studio и MS SQL Server .

Оглавление

[Введение. 3](#_Toc103631683)

[**1. Разработка эскизного и технического проекта программы.** 4](#_Toc103631684)

[***1.1. Назначение и область применения.*** 4](#_Toc103631685)

[***1.2. Технические характеристики. Данный пункт должен содержать следующие подпункты:*** 4](#_Toc103631686)

[*1.2.1. Постановка задачи.* 4](#_Toc103631687)

[*1.2.2. Создание UML диаграммы* 7](#_Toc103631688)

[Модели 7](#_Toc103631689)

[*1.2.3. Организация входных и выходных данных.* 9](#_Toc103631690)

[*1.2.4. Выбор состава технических и программных средств.* 10](#_Toc103631691)

[**2. Разработка рабочего проекта.** 12](#_Toc103631692)

[***2.1. Разработка программы.*** 12](#_Toc103631693)

[***2.2. Спецификация программы.*** 20](#_Toc103631694)

[***2.3. Текст программы.*** 23](#_Toc103631695)

[***2.4. Описание программы.*** 87](#_Toc103631696)

[***2.5. Тестирование программы.*** 88](#_Toc103631697)

[**Заключение.** 90](#_Toc103631698)

[**Список использованной литературы.** 90](#_Toc103631699)

[**Приложения.** 91](#_Toc103631700)

# Введение.

На сегодняшний день трудно представить мир без технических устройств, которые уже давно вошли в нашу жизнь. В крупных городах, где проживают люди и производятся миллионы финансовых операций, особенно сейчас занимают банки. В банках оформляются кредиты, оплачиваются услуги, хранятся и переводятся денежные средства. Еще совсем недавно люди вынуждены были ходить на почту или в центры сотовой связи, чтобы заплатить деньги за коммунальные услуги или за использование сети. Все это отнимало много времени как у пользователей, так и у работников этих учреждений. Поэтому, для более надежного хранения денежных средств и для более быстрого доступа к ним придумали банковскую карту и банкомат.  
Почти во всех фирмах, сотрудники получают зарплату, приходящую на счет в банке, к которому привязана карта. Для предприятия более выгоден именно такой расчет, а не наличный, потому что наличные средства нуждаются в дополнительном обслуживании.

Целью курсовой работы является моделирование работы банкомата. Эта разработка будет полезна для обучения владельцев банковских карт.

Основными задачами курсовой работы является: разработка программного продукта и рассмотрение предметной области. Моделирование системы банкомата включает в себя:

* описание с помощью UML диаграмм.
* построение базы данных.
* Создание приложения.

## **1. Разработка эскизного и технического проекта программы.**

### ***1.1. Назначение и область применения.***

Данная программа является деловой программой. Областью применения данной программы является организация досуга человека. Так как, поставлена задача, разработать приложение под банкомат.

Сам банкомат – это автомат для выдачи наличных денег по кредитным пластиковым карточкам. В его состав входят следующие устройства: дисплей, панель управления с кнопками, приемник кредитных карт, хранилище денег и лоток для их выдачи, хранилище конфискованных кредитных карт, принтер для печати. Банкомат подключен к линии связи для обмена данных с банковской информационной системой, содержащей сведения о счетах клиентов.

## ***1.2. Технические характеристики. Данный пункт должен содержать следующие подпункты:***

### *1.2.1. Постановка задачи.*

Тип данных в данном случаи числовой (int), так как мы работаем с сомой суммой на карте.

Приложение «Модель работы банкомата» должно реализовывать следующие функции (цели с описанием):

1.Проверка PIN-кода карты.

После помещения пластиковой карты в банкомат (ввода номера карты в текстовое поле) банкомат проверяет, существует ли такой номер карты на сервере. Если введенный номер действителен, пользователю предлагается ввести PIN-код для продолжения работы с картой. После ввода четырехзначного кода банкомат проверяет на соответствие данного PIN-кода указанному номеру карты. Если соответствие установлено, пользователю будет предложено выбрать дальнейшую операцию по карте.

2. Предоставление сведений об остатке на счете.

Когда пользователь выбирает операцию «Запросить баланс», банкомат связывается с сервером и получает значение остатка на счете по соответствующему номеру карты. Затем это значение отображается на дисплее банкомата.

3. Выдача наличных.

Когда пользователь выбирает операцию «Получить наличные», банкомат предлагает выбрать требуемую сумму для выдачи: 100, 200, 500, 1000, 5000, 10000, 20000, 50000, но не выше этой суммы или свою сумму. После выбора суммы банкомат сначала проверяет остаток на счете, затем, если остаток больше требуемой суммы, проверяет возможность выдачи требуемой суммы купюрами, находящимися в хранилище купюр банкомата. Если такая возможность существует, пользователю выдаются наличные, при этом изменяется значение остатка по счету и количество купюр в хранилище.

4. Выполнение денежного перевода.

Когда пользователь выбирает операцию «Денежный перевод», на дисплей банкомата выводится форма ввода номера карты, на которую осуществляется перевод, и суммы перевода. При выполнении перевода происходит списание средств с текущего счета и зачисление их на указанный счет.

5. Оплата услуг сотовой связи.

Когда пользователь выбирает операцию «Оплата сотовой связи», на дисплей банкомата выводится форма ввода номера телефона, на который осуществляется пополнение, и суммы перевода. При выполнении перевода происходит списание средств с текущего счета.

6. Оплата коммунальных платежей.

Когда пользователь выбирает операцию «Коммунальные платежи» или «Оплата электроэнергии», на экран выводится форма с элементами выбора поставщика услуг и ввода персонального счета клиента и суммы оплаты. При выполнении платежа происходит списание средств с текущего счета.

7. Печать квитанции.

После выдачи наличных пользователю предлагается распечатать квитанцию об операции. Квитанция содержит номер счета, выданную сумму и остаток по счету.

8. Просмотр истории операций по карте.

Когда пользователь выбирает операцию «История операций», на экран выводится форма, содержащая элементы выбора периода, за который предоставить отчет, и таблицу, которая заполняется информацией об операциях по карте за указанный период.

9.Выход

После того как пользователь выполнил все свои действия, он нажимает кнопку “Выйти” и приложение переходит в начальное состояние.

Условие к «Модель работы банкомата» должно реализовывать следующие функции (условия с решением):

1. Проверка PIN-кода карты.

Если PIN-код был введен неправильно 3 раза, то кредитная пластиковая карточка блокируется и её можно разблокировать только в банке.

(то есть программа банкомата переходит в начальное состояние)

1. Если нет наличных на карте.

Если на кредитной пластиковой карточке нет средств при выполнение определённых услуг (выдача наличных, выполнение денежного перевода, оплата услуг сотовой связи, оплата коммунальных платежей), то пишется “Не достаточно денег на счёте”.

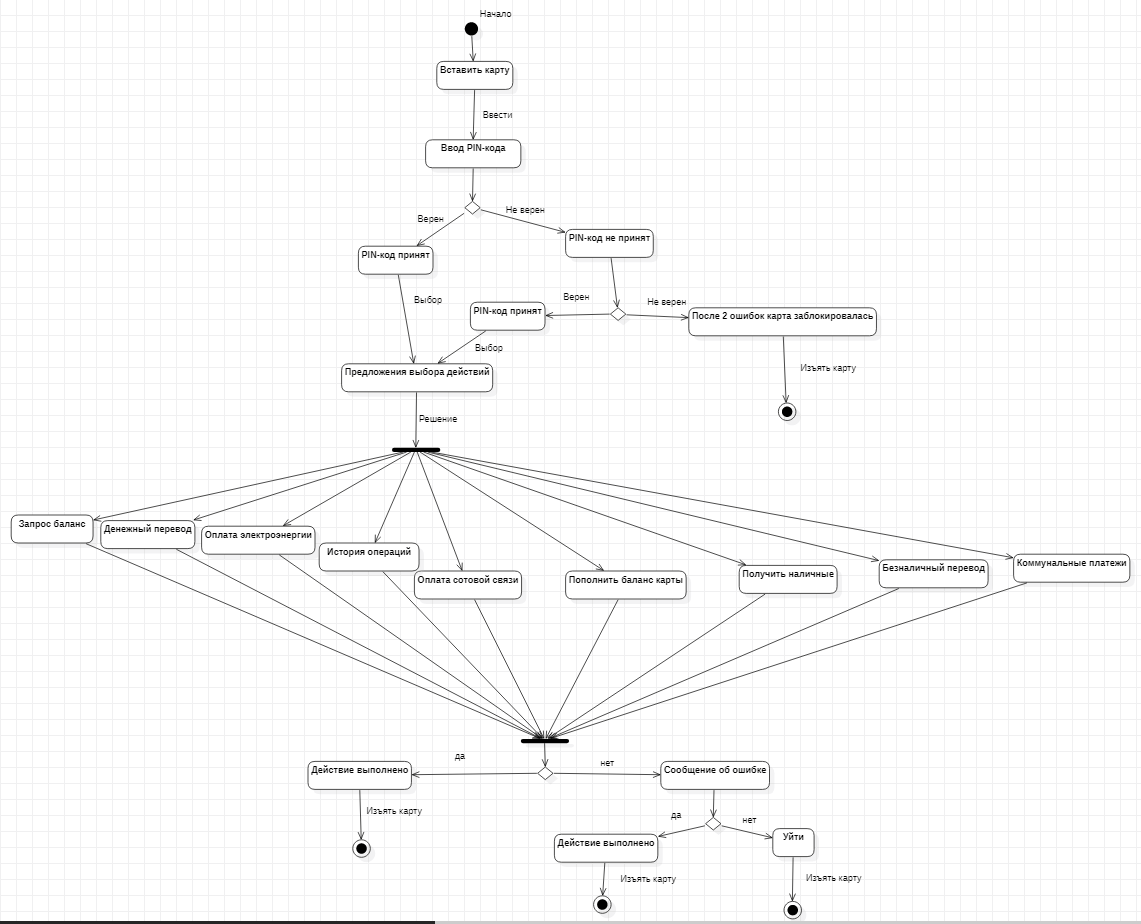
1. Нет денег в банкомате.

Если нет денег в банкомате, то выводится сообщение “Невозможно выдать требуемую сумму”.

### *1.2.2. Создание UML диаграммы*

# Модели

Диаграмма деятельности (активности) (Рисунок 1):



Пользователь вводит номер своей карты, по которой он хочет совершить какие-либо операции. Приложение осуществляет запрос к базе данных на получение подробной информации по введенному номеру карты. Если такой не найден, пользователь оповещается об этом сообщением и если пользователь ввёл 3 раза неправильно номер карты или PIN-код, то карта блокируется. Когда система находит требуемый номер счета, и пользователю предлагается ввести четырехзначный PIN-код.

После ввода PIN-кода банкомат проверяет его правильность. Для этого производится получение из базы данных PIN-кода карты по её номеру, затем выполняется сравнение полученного PIN-кода и введенного пользователем. Если данные коды не совпадают, пользователь оповещается сообщением о неверно введенном PIN-коде. При совпадении кодов пользователю предлагается выбрать дальнейшую операцию по карте. Такие как запрос баланса, денежный перевод, оплата электроэнергии, история операций, оплата сотовой связи, пополнить баланс карты, получить наличные, безналичный перевод, коммунальные платежи. Дальше происходит одно из 2-ух действий либо действие выполнилось, либо в операции произошла какая-то ошибка и вывелось сообщение о ней и после этого совершаем другое действие если нужно или же мы вытаскиваем (изъятие) карты и завершаем процесс.

Для оформления данного подпункта используется ГОСТ 19.701-90 ЕСПД. Схемы алгоритмов, программ, данных и систем. Условные обозначения и правила выполнения.

### *1.2.3. Организация входных и выходных данных.*

Входными данными для приложения являются:

* Введенный PIN-код для кредитной пластиковой карточке.
* Введённая сумма для снятия, оплаты или перевода денег.

Выходными данными для приложения являются:

* Сообщения от банкомата
* Выдача суммы денег
* Выдача чека за выполнения услуги.

### *1.2.4. Выбор состава технических и программных средств.*

*Исходя из целей курсовой работы, требуется* создать программу, работающую под ОС Банкомата. В связи с тем, что разработанный алгоритм подразумевает использование графических возможностей, было принято решение использовать интегрированную среду разработки программ Visual Studio. Данная среда разработки позволяет достаточно быстро создавать приложения для банкомата.

*Достоинства:*

* Возможность получения наличных денег.
* Совершения доступных операций в любое время суток без посещения отделения и выстраивания длинных очередей.
* Банкомат обеспечивает конфиденциальность транзакций или каких-либо других услуг, потому что в салоне может находиться только один клиент, пока он не завершит транзакции или выполнение услуг.

*Недостатки:*

* Не всегда удобное для клиента расположение банкомата.
* возможные потери и кражи карточек;
* могут быть сложности в случае, если в данное время банкомат не работает по разным причинам.

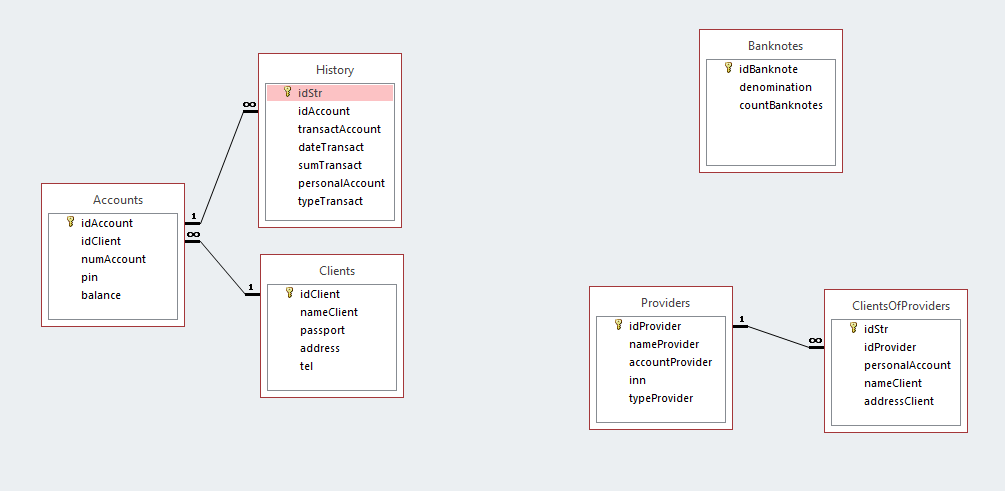
Технические средства, достаточные для оптимальной работы программы:

* Видеокамера на внешней стороне устройства — предназначена для фиксации лица человека, совершающего операцию.
* Считывающее устройство — кардридер.
* Компьютер и система безопасности.
* Принтер — отвечает за печать чеков по каждой операции.
* Кассета (кассеты) с купюрами из банка — из них банкомат осуществляет выдачу наличных денежных средств.
* Кассета (кассеты) в которые помещаются наличные, внесенные в банкомат.
* Механизм для забора денег из кассеты и их последующей выдачи.
* Клавиатуру и сенсорный экран для ввода PIN-кода.
* Модем – для отправки запроса и принятия его из банка.

## **2. Разработка рабочего проекта.**

### ***2.1. Разработка программы.***

Как уже отмечалось, для разработки приложения банкомата используется среда визуального программирования Visual Studio. В данной среде был создан проект с названием прием заказа. Данный проект состоит из Базы данных Access



1. Таблица «Clients» - содержит информации о клиентах банка.

Поля таблицы:

- idClient- код клиента, первичный ключ, счетчик, служит для автоматической индексации записей о клиентах;

- nameClient – ФИО клиента, текстовый;

- passport – серия и номер паспорта клиента, текстовый;

- address – адрес регистрации клиента, текстовый;

- tel – номер телефона клиента, текстовый.

2. Таблица «Accounts» - содержит информацию о счетах клиентов банка.

Поля таблицы:

- idAccount – код счета, первичный ключ, счетчик, служит для автоматической индексации записей о счетах клиентов;

- idClient – код клиента, внешний ключ на таблицу «Clients», числовой, служит для определения принадлежности счета конкретному клиенту;

- numAccount – номер карты клиента, текстовый;

- pin – PIN-код карты, текстовый;

- balance – значение остатка средств на счете, числовой.

3. Таблица «Providers» - содержит информацию о поставщиках услуг.

Поля таблицы:

- idProvider – код поставщика, первичный ключ, счетчик, служит для автоматической индексации записей о поставщиках;

- nameProvider – наименование поставщика, текстовый;

- accountProvider – номер банковского счета поставщика, текстовый;

- inn – ИНН поставщика, текстовый;

- typeProvider – категория оказываемых поставщиком услуг, текстовый.

4. Таблица «ClientsOfProviders» - содержит информацию о клиентах поставщиков услуг.

Поля таблицы:

- idStr – код записи, первичный ключ, счетчик, служит для автоматической индексации записей о клиентах поставщиков;

- idProvider – код поставщика, внешний ключ на таблицу «Providers», числовой, служит для определения принадлежности клиента конкретному поставщику;

- personalAccount – номер лицевого счета клиента у данного поставщика, текстовый;

- nameClient – ФИО клиента, текстовый;

- addressClient – адрес клиента, текстовый.

5. Таблица «History» - содержит информацию о севершенных операциях по картам клиентов.

Поля таблицы:

- idStr- код записи, первичный ключ, счетчик, служит для автоматической индексации записей о совершенных операциях;

- idAccount – код счета, внешний ключ на таблицу «Accounts», числовой, служит для определения принадлежности совершенной операции конкретному номеру счета;

- transactAccount – номер счета, на который или с которого были переведены деньги при безналичном переводе, текстовый;

- dateTransact – дата и время совершения операции, дата/время;

- sumTransact – сумма, на которую была совершена операция (при списании средств – отрицательная, при зачислении – положительная), числовой;

- personalAccount – номер лицевого счета при оплате коммунальных услуг, текстовый;

- typeTransact – тип совершенной операции (снятие наличных, денежный перевод и т.д.), числовой.

6. Таблица «Banknotes» - содержит информацию о количестве купюр, имеющихся в наличии в банкомате.

Поля таблицы:

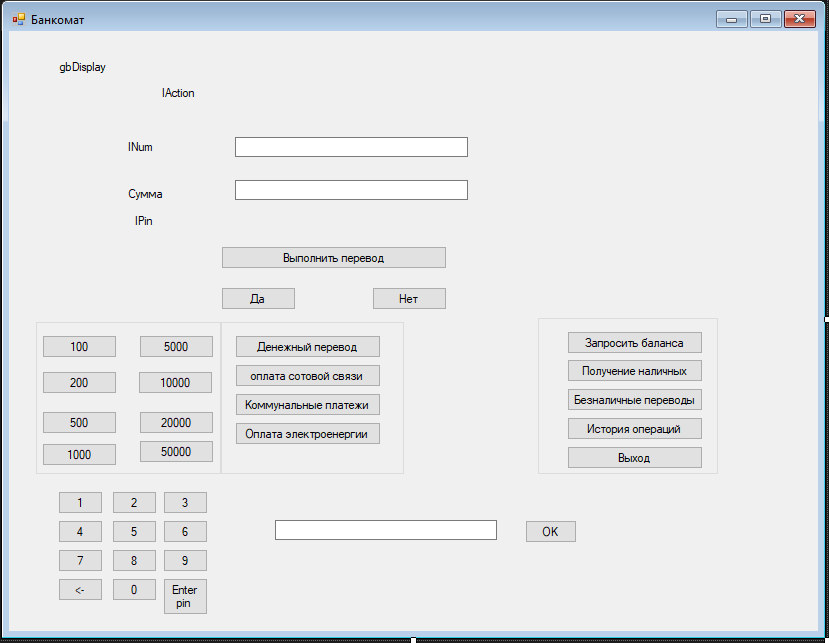
- idBanknote – код банкноты, первичный ключ, счетчик, служит для автоматической индексации записей о купюрах;

- denomination – номинал купюры, текстовый;

- countBanknotes – количество имеющихся купюр данного номинала, числовой.

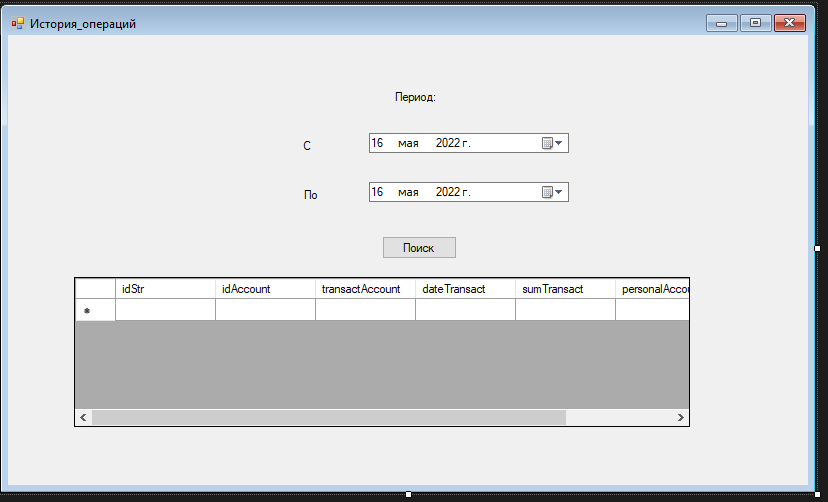
и форм:

Форма номер 1 главная (Банкомат)



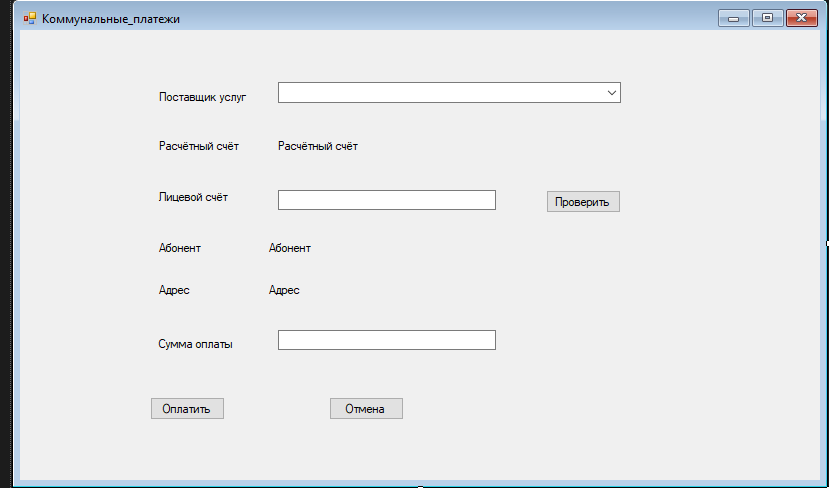
На данной форме располагаются 33 кнопки: поля для ввода данных, Ок (для авторизации пользователя), цифровой блок (от 0 до 9, clear, enter pin), запросить баланс, получение наличных, безналичный перевод, история операций, выход, выполнить перевод, да, нет, денежный перевод, оплата сотовой связи, коммунальные платежи, оплата электроэнергии и кнопки для получения наличных (100,200,500,1000,5000,10000,20000,50000)

Форма номер 2 (История операций)



На данной форме располагаются: кнопка Поиск, dataGridView(где происхлдит поиск) и 2 dateTimePicker

Форма номер 3 (Коммунальные платежи)



На данной форме располагаются: поля для ввода данных, Label для описания и 3 кнопки проверить, оплатить, отменить.

### ***2.2. Спецификация программы.***

Для разработки моей банкомат системы я выбрал программы Microsoft Visual Studio и Access . Программа Microsoft Visual Studio имеет следующие спецификации:

Файл БД программы банкомат имеет название bank и расположен в каталоге программы.



|  |  |
| --- | --- |
| Год выпуска: | 2017 |
| ОС: | Windows |
| Категория: | Веб-разработка и Программирование |
| Требования: | Процессор с частотой 1,6 ГГц или более мощный. ОЗУ объемом 1 ГБ (1,5 ГБ для работы на виртуальной машине) 10 ГБ доступного пространства на жестком диске (30 Гб включая все дополнения) Жесткий диск с частотой вращения 5 400 об./мин Видеокарта с поддержкой DirectX 9 и разрешения дисплея 1024x768 или выше Windows 10/8.1/8/7 SP1/Server 2012 R2/Server 2012/Server 2008 R2 SP1 32-bit|64-bit |
| Язык интерфейса: | Русский |
| Лекарство: | не требуется |

**Microsoft Visual Studio**- линейка продуктов компании Microsoft, включающих интегрированную среду разработки программного обеспечения и ряд других инструментальных средств. Данные продукты позволяют разрабатывать как консольные приложения, так и приложения с графическим интерфейсом, в том числе с поддержкой технологии Windows Forms, а также веб-сайты, веб-приложения, веб-службы как в родном, так и в управляемом кодах для всех платформ, поддерживаемых Windows, Windows Mobile, Xbox, и Silverlight, а так же Android и iOS (Xamarin).

Программа программы Microsoft SQL studio имеет следующие спецификации:

| **Компонент** | **Требование** |
| --- | --- |
| Жесткий диск | ДляSQL Server требуется как минимум 6 ГБ свободного места на диске.  Требования к месту на диске определяются набором устанавливаемых компонентов SQL Server . Дополнительные сведения см. в разделе [Требования к месту на диске](https://docs.microsoft.com/ru-ru/sql/sql-server/install/hardware-and-software-requirements-for-installing-sql-server?view=sql-server-ver15#HardDiskSpace) далее в этой статье. Сведения о поддерживаемых типах хранилищ для файлов данных см. в разделе [Storage Types for Data Files](https://docs.microsoft.com/ru-ru/sql/sql-server/install/hardware-and-software-requirements-for-installing-sql-server?view=sql-server-ver15" \l "StorageTypes). |
| Монитор | ДляSQL Server требуется монитор Super VGA с разрешением 800x600 пикселей или более высоким. |
| Интернет | Для поддержки функциональных средств Интернета требуется доступ к Интернету (могут применяться дополнительные тарифы). |
| Память \* | **Минимальные:**  Экспресс-выпуски: 512 МБ  Все другие выпуски: 1 ГБ  **Рекомендуется:**  Экспресс-выпуски: 1 ГБ  Все другие выпуски: Для обеспечения оптимальной производительности требуется не менее 4 ГБ с последующим увеличением по мере роста размера базы данных. |
| Быстродействие процессора | **Минимум**: процессор x64 с тактовой частотой 1,4 ГГц  **Рекомендуется:** 2,0 ГГц и выше |
| Тип процессора | Процессор x64: AMD Opteron, AMD Athlon 64, Intel Xeon с поддержкой Intel EM64T, Intel Pentium IV с поддержкой EM64T. |

### ***2.3. Текст программы.***

Приводим полный листинг программы вместе с комментариями.

Листинг кода Форма номер 1 главная (Банкомат)

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Windows.Forms;

using System.IO;

using System.Data.OleDb;

namespace Дастало

{

public partial class Банкомат : Form

{

ClassBankServer server; //экземпляр класса для связи с сервером

public Банкомат()

{

InitializeComponent();

}

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

{

lAction.Text = "Вставьте карту";

server = new ClassBankServer();

tbCard.PasswordChar = '\*'; //скрыть вводимые символы номера карты

}

string numAccount = ""; //номер счета

bool access; //переменная, определяющая, дан ли доступ данной карте

private void btEnterCard\_Click(object sender, EventArgs e) //ввод карты

{

numAccount = tbCard.Text;

access = server.CheckCard(numAccount); //связь с сервром: проверка карты

if (access == true) //если карта есть в базе

{ //ввести пин-код

lAction.Text = "Введите PIN-код";

lPin.Text = "PIN: ";

gbDisplay.Visible = true;

lPin.Visible = true;

pin = "";

string numCard = "\*\*\*\*\*\*\*\*" + numAccount.Substring(8); //номер карты, где отображены только последние 4 цифры

gbDisplay.Text = numCard; //отобразить его на форме в углу дисплея

}

else

{

lAction.Text = "Данная карта недействительна";

}

}

string pin = ""; //введенный пин-код

void DisplayPIN(string code) //метод отображения на форме вводимого пин-кода

{

string pinCode = "PIN: ";

if (code.Length <= 4) //максимальная длина кода - 4 символа

{

foreach (char ch in code)

pinCode += "\*"; //отображать на форме в виде \*

lPin.Text = pinCode;

}

}

//

private void bt1\_Click(object sender, EventArgs e) //при нажатии кнопки на цифровом блоке

{

if (tbCardTansact.Visible == true && mtbMobile.Visible == false) //если отображена форма ввода номера карты для перевода

{

tbCardTansact.Text += "1"; //добавить цифру в поле номера карты

}

else if (tbCardTansact.Visible == false && mtbMobile.Visible == false) //если панель ввода не отображена

{ //происходит ввод пин-кода

if (access == true && pin.Length < 4) pin += "1"; //если карта действительна и длина введенного пин-кода меньше 4 символов, добавить цифру к пин-коду

DisplayPIN(pin); //выполнить функцию отображения пин-кода на дисплее

}

}

int attempt = 3; //количество попыток ввода пин-кода

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

{

access = server.CheckPin(numAccount, pin); //проверка правильности пин-кода

if (access == true) //если пин верен

{ //отобразить на форме элементы управления

lAction.Text = "Выберите действие"; //для совершения операций по карте

lPin.Visible = false;

gbButtons.Visible = true;

}

else

{ //если пин введен неверно

lAction.Text = "Неверный PIN-код";

pin = "";

DisplayPIN(pin);

attempt--; //уменьшение количества доступных попыток

if (attempt > 0) MessageBox.Show("Осталось попыток: " + attempt.ToString());

else

{ //если пин введен неверно 3 раза

MessageBox.Show("PIN-код неверно введен 3 раза. Обслуживание прекращено");

btExit\_Click(sender, e); //прекратить обслуживание и отобразить первоначальную форму

}

access = true;

}

}

private void bt0\_Click(object sender, EventArgs e)

{

if (tbCardTansact.Visible == true && mtbMobile.Visible == false)

{

tbCardTansact.Text += "0";

}

else if (tbCardTansact.Visible == false && mtbMobile.Visible == false)

{

if (access == true && pin.Length < 4) pin += "0";

DisplayPIN(pin);

}

}

private void btCorrect\_Click(object sender, EventArgs e) //кнопка корректировки введенных символов

{

if (tbCardTansact.Visible == true && mtbMobile.Visible == false) //корректировка номера карты для перевода

{

tbCardTansact.Text = tbCardTansact.Text.Remove(tbCardTansact.TextLength - 1);

}

else if (tbCardTansact.Visible == false && mtbMobile.Visible == false) //корректировка пин-кода

{

if (access == true) pin = pin.Remove(pin.Length - 1);

DisplayPIN(pin);

}

}

private void bt9\_Click(object sender, EventArgs e)

{

if (tbCardTansact.Visible == true && mtbMobile.Visible == false)

{

tbCardTansact.Text += "9";

}

else if (tbCardTansact.Visible == false && mtbMobile.Visible == false)

{

if (access == true && pin.Length < 4) pin += "9";

DisplayPIN(pin);

}

}

private void bt8\_Click(object sender, EventArgs e)

{

if (tbCardTansact.Visible == true && mtbMobile.Visible == false)

{

tbCardTansact.Text += "8";

}

else if (tbCardTansact.Visible == false && mtbMobile.Visible == false)

{

if (access == true && pin.Length < 4) pin += "8";

DisplayPIN(pin);

}

}

private void bt7\_Click(object sender, EventArgs e)

{

if (tbCardTansact.Visible == true && mtbMobile.Visible == false)

{

tbCardTansact.Text += "7";

}

else if (tbCardTansact.Visible == false && mtbMobile.Visible == false)

{

if (access == true && pin.Length < 4) pin += "7";

DisplayPIN(pin);

}

}

private void bt6\_Click(object sender, EventArgs e)

{

if (tbCardTansact.Visible == true && mtbMobile.Visible == false)

{

tbCardTansact.Text += "6";

}

else if (tbCardTansact.Visible == false && mtbMobile.Visible == false)

{

if (access == true && pin.Length < 4) pin += "6";

DisplayPIN(pin);

}

}

private void bt5\_Click(object sender, EventArgs e)

{

if (tbCardTansact.Visible == true && mtbMobile.Visible == false)

{

tbCardTansact.Text += "5";

}

else if (tbCardTansact.Visible == false && mtbMobile.Visible == false)

{

if (access == true && pin.Length < 4) pin += "5";

DisplayPIN(pin);

}

}

private void bt4\_Click(object sender, EventArgs e)

{

if (tbCardTansact.Visible == true && mtbMobile.Visible == false)

{

tbCardTansact.Text += "4";

}

else if (tbCardTansact.Visible == false && mtbMobile.Visible == false)

{

if (access == true && pin.Length < 4) pin += "4";

DisplayPIN(pin);

}

}

private void bt3\_Click(object sender, EventArgs e)

{

if (tbCardTansact.Visible == true && mtbMobile.Visible == false)

{

tbCardTansact.Text += "3";

}

else if (tbCardTansact.Visible == false && mtbMobile.Visible == false)

{

if (access == true && pin.Length < 4) pin += "3";

DisplayPIN(pin);

}

}

private void bt2\_Click(object sender, EventArgs e)

{

if (tbCardTansact.Visible == true && mtbMobile.Visible == false)

{

tbCardTansact.Text += "2";

}

else if (tbCardTansact.Visible == false && mtbMobile.Visible == false)

{

if (access == true && pin.Length < 4) pin += "2";

DisplayPIN(pin);

}

}

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

{

lAction.Text = "";

lPin.Visible = true;

lPin.Text = "Остаток по счету: \n" + server.GetBalance(numAccount).ToString() + " рублей"; //связь с сервером: проверка баланса

btYes.Visible = false;

btNo.Visible = false;

gbSum.Visible = false;

pTransact.Visible = false;

pTransact.Visible = false;

}

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

{

lPin.Visible = false; //отобразить на форме элементы

lAction.Text = "Выберите сумму"; //выбора суммы для получения

gbSum.Visible = true;

pTransact.Visible = false;

pTransact.Visible = false;

}

DateTime dateTransact;

double givenSum = 0; //выданная сумма

void GetCash(int sum) //выдача наличных

{

access = false;

givenSum = 0;

dateTransact = DateTime.Now;

lAction.Text = "Получить " + sum.ToString() + " рублей";

if (sum <= server.GetBalance(numAccount)) access = true; //связь с сервером: проверка наличия требуемой суммы на счете

if (access == true) //если такая сумма имеется

{

lPin.Visible = true; //отображение на форме

lPin.Text = "Распечатать квитанцию?"; //предложения распечать чек

btYes.Visible = true;

btNo.Visible = true;

string given = "Выдано: ";

int count5tr = sum / 5000; //количесто выдаваемых купюр по 5000

int count = GetCountBanknotes("5000");

if (count >= count5tr)

{

givenSum += count5tr \* 5000;

sum -= count5tr \* 5000; //вычислить остаток, который выдавать другими купюрами

given += ("5000 - " + count5tr.ToString() + ", ");

}

else if (count < count5tr)

{

givenSum += count \* 5000;

sum -= count \* 5000; //вычислить остаток, который выдавать другими купюрами

given += ("5000 - " + count.ToString() + ", ");

count5tr = count;

}

int count1tr = sum / 1000;

count = GetCountBanknotes("1000");

if (count >= count1tr)

{

givenSum += count1tr \* 1000;

sum -= count1tr \* 1000; //вычислить остаток, который выдавать другими купюрами

given += ("1000 - " + count1tr.ToString() + ", ");

}

else if (count < count1tr)

{

givenSum += count \* 1000;

sum -= count \* 1000; //вычислить остаток, который выдавать другими купюрами

given += ("1000 - " + count.ToString() + ", ");

count1tr = count;

}

int count5sr = sum / 500;

count = GetCountBanknotes("500");

if (count >= count5sr)

{

givenSum += count5sr \* 500;

sum -= count5sr \* 500; //вычислить остаток, который выдавать другими купюрами

given += ("500 - " + count5sr.ToString() + ", ");

}

else if (count < count5sr)

{

givenSum += count \* 500;

sum -= count5sr \* 500; //вычислить остаток, который выдавать другими купюрами

given += ("500 - " + count.ToString() + ", ");

count5sr = count;

}

int count1sr = sum / 100;

count = GetCountBanknotes("100");

if (count >= count1sr)

{

givenSum += count1sr \* 100;

sum -= count1sr \* 100; //вычислить остаток, который выдавать другими купюрами

given += ("100 - " + count1sr.ToString() + ", ");

}

else if (count < count1sr)

{

givenSum += count \* 100;

sum -= count1sr \* 100; //вычислить остаток

given += ("100 - " + count.ToString() + ", ");

count1sr = count;

}

if (sum != 0) //в итоге остаток должен стать равен 0

{ //если этого не случилось, значит в хранилище нет каких-либо купюр

MessageBox.Show("Невозможно выдать требумую сумму"); //в достаточном количестве

lPin.Visible = false;

btYes.Visible = false;

btNo.Visible = false;

}

else

{

UpdateBanknotes("5000", count5tr);

UpdateBanknotes("1000", count1tr);

UpdateBanknotes("500", count5sr);

UpdateBanknotes("100", count1sr);

server.GetCash(numAccount, givenSum);

server.InsertToHistory("", dateTransact, -givenSum, "", "getCash");

MessageBox.Show(given); //отобразить сообщение, какими купюрами в каком количестве

} //выдана данная сумма

}

else // если на балансе нет трубуемой суммы

{

lPin.Visible = true;

lPin.Text = "Недостаточно денег";

}

}

int GetCountBanknotes(string denomination)

{

int count = 0;

OleDbConnection conn = new OleDbConnection("Provider=Microsoft.ACE.OleDb.12.0;Data Source=" + Application.StartupPath.ToString() + "\\bank.accdb;Persist Security Info=False;");

conn.Open();

string sqlFindCard = "SELECT Banknotes.countBanknotes FROM Banknotes WHERE denomination=@denomination";

OleDbCommand cmd = new OleDbCommand(sqlFindCard, conn);

OleDbParameter[] prms = new OleDbParameter[1];

prms[0] = new OleDbParameter("@denomination", OleDbType.VarChar);

prms[0].Value = denomination;

cmd.Parameters.AddRange(prms);

OleDbDataReader dr = cmd.ExecuteReader();

while (dr.Read()) count = Convert.ToInt32(dr.GetValue(0));

return count;

}

void UpdateBanknotes(string denomination, int count)

{

OleDbConnection conn = new OleDbConnection("Provider=Microsoft.ACE.OleDb.12.0;Data Source=" + Application.StartupPath.ToString() + "\\bank.accdb;Persist Security Info=False;");

conn.Open();

string sqlUpdateBalance = "UPDATE Banknotes SET countBanknotes=(countBanknotes-@count) WHERE denomination=@denomination";

OleDbCommand cmd = new OleDbCommand(sqlUpdateBalance, conn);

OleDbParameter[] prms = new OleDbParameter[2];

prms[0] = new OleDbParameter("@count", OleDbType.Integer);

prms[0].Value = count;

prms[1] = new OleDbParameter("@denomination", OleDbType.VarChar);

prms[1].Value = denomination;

cmd.Parameters.AddRange(prms);

cmd.ExecuteNonQuery();

conn.Close();

conn.Dispose();

}

private void bt100r\_Click(object sender, EventArgs e)

{

GetCash(100);

}

private void bt200r\_Click(object sender, EventArgs e)

{

GetCash(200);

}

private void bt500r\_Click(object sender, EventArgs e)

{

GetCash(500);

}

private void bt1tr\_Click(object sender, EventArgs e)

{

GetCash(1000);

}

private void bt5tr\_Click(object sender, EventArgs e)

{

GetCash(5000);

}

private void bt10tr\_Click(object sender, EventArgs e)

{

GetCash(10000);

}

private void bt20tr\_Click(object sender, EventArgs e)

{

GetCash(20000);

}

private void bt50tr\_Click(object sender, EventArgs e)

{

GetCash(50000);

}

private void btYes\_Click(object sender, EventArgs e) //печать чека

{

string date = DateTime.Now.ToString();

string numCard = "\*\*\*\*\*\*\*\*" + numAccount.Substring(8); //номер карты, где отображены только последние 4 цифры

string print = "Время оперции: " + date +

"\n\nНомер счета: " + numCard + //распечать номер счета

"\nВыдано: " + givenSum + //выданную сумму

"\nОстаток: " + server.GetBalance(numAccount).ToString(); //и остаток по карте

MessageBox.Show(print);

btNo\_Click(sender, e); //отобразить элементы для выполнения новой операции

}

private void btNo\_Click(object sender, EventArgs e) //не печатать чек

{

lAction.Text = "Выберите действие";

lPin.Visible = false;

gbSum.Visible = false;

btNo.Visible = false;

btYes.Visible = false;

pTransact.Visible = false;

gbSum.Visible = false;

}

private void btExit\_Click(object sender, EventArgs e) //выход, прекращение обслуживания

{ //отображение первоначальной формы

lAction.Text = "Вставьте карту";

lPin.Visible = false;

attempt = 3;

gbButtons.Visible = false;

pTransact.Visible = false;

pTransact.Visible = false;

tbCard.Clear();

tbCardTansact.Clear();

tbSum.Clear();

mtbMobile.Clear();

pin = "";

gbDisplay.Text = "";

}

string transact = ""; //переменная для определения типа перевода

private void btTransact\_Click(object sender, EventArgs e) //денежный перевод

{

lAction.Text = "Денежный перевод";

lPin.Visible = false;

gbSum.Visible = false;

lNum.Visible = true;

//отобразить панель для ввода номера карты и суммы для перевода

pTransact.Visible = true;

mtbMobile.Visible = false;

tbCardTansact.Visible = true;

tbSum.Visible = true;

Summ.Visible = true;

btRunTransact.Visible = true;

lNum.Text = "Номер карты";

transact = "card";

}

//выполнить перевод

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

{

//денежный перевод на карту

if (transact == "card")

{

string numCard = tbCardTansact.Text; //номер карты

double sum = Convert.ToDouble(tbSum.Text); //сумма

dateTransact = DateTime.Now;

access = server.CheckCard(numCard); //проверка указанной карты

if (access == true) //если карта найдена

{

if (sum <= server.GetBalance(numAccount))

{

server.GetCash(numAccount, sum);

server.Transact(numCard, sum); //зачисление средств на указанный счет

server.InsertToHistory(numAccount, dateTransact, -sum, "", "transact");

double s = server.GetBalance(numCard);

server.InsertToHistory(numCard, dateTransact, sum, "", "transact");

MessageBox.Show("Перевод успешно выполнен");

pTransact.Visible = false;

lAction.Text = "Выберите действие";

}

else

{

MessageBox.Show("Недостаточно денег на счете");

}

}

else MessageBox.Show("Номер карты не найден");

}

//оплата сотовой связи

if (transact == "tel")

{

string tel = mtbMobile.Text; //номер телефона

tbSum.Visible = true;

double sum = Convert.ToDouble(tbSum.Text); //сумма пополнения

if (sum <= server.GetBalance(numAccount))

{

server.GetCash(numAccount, sum);

server.InsertToHistory(tel, dateTransact, -sum, "", "tel");

MessageBox.Show("Пополнение счета " + tel + " успешно выполнено");

pTransact.Visible = false;

lAction.Text = "Выберите действие";

}

else

{

MessageBox.Show("Недостаточно денег на счете");

}

}

}

//оплата сотовой связи

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

{

lAction.Text = "Оплата сотовой связи";

lPin.Visible = false;

gbSum.Visible = false;

tbSum.Visible = true;

lNum.Visible = true;

Summ.Visible = true;

btRunTransact.Visible = true;

//отобразить панель для ввода номера телефона и суммы для пополнения

pTransact.Visible = true;

mtbMobile.Visible = true;

tbCardTansact.Visible = false;

lNum.Text = "Номер телефона";

transact = "tel";

}

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

{

lAction.Text = "Выберите вид перевода";

lPin.Visible = false;

gbSum.Visible = false;

pTransact.Visible = true;

}

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

{

FormCommunalPaid fcp = new FormCommunalPaid();

fcp.typeTransact = "communal";

fcp.numAccount = numAccount;

fcp.Show();

}

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

{

FormCommunalPaid fcp = new FormCommunalPaid();

fcp.typeTransact = "energy";

fcp.numAccount = numAccount;

fcp.Show();

}

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

{

lPin.Visible = false;

gbSum.Visible = false;

pTransact.Visible = false;

FormHistory fh = new FormHistory();

fh.numAccount = numAccount;

fh.Show();

}

}

}

Листинг кода Форма номер 2 (История операций)

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Windows.Forms;

using System.Data.OleDb;

namespace Дастало

{

public partial class FormHistory : Form

{

public FormHistory()

{

InitializeComponent();

}

public string numAccount; //номер счета

private void btOK\_Click\_1(object sender, EventArgs e)

{

dgvHistory.Rows.Clear(); //очистить таблицу

DateTime dateStart = dtDateEnd.Value.Date; //дата начала периода

DateTime dateEnd = dtDateEnd.Value.Date; //дата окончания периода для выборки

ClassBankServer server = new ClassBankServer();

server.GetBalance(numAccount); //определение id счета

OleDbDataReader dr = server.History(dateStart, dateEnd); //вызов метода истории операций по карте

string name; //наименование поставщика

while (dr.Read())

{

if (dr.GetValue(1).ToString() == "") //если номер счета для перевода отсутствует (были сняты наличные)

{

dgvHistory.Rows.Add(dr.GetValue(0).ToString(), "", "Выдача наличных", dr.GetValue(2).ToString()); //заполнение строки таблицы

}

else

{

name = server.GetName(dr.GetValue(3).ToString(), dr.GetValue(1).ToString()); //получение наименования поставщика

dgvHistory.Rows.Add(dr.GetValue(0).ToString(), dr.GetValue(1).ToString(), name, dr.GetValue(2).ToString()); //заполнение строк таблицы

}

}

}

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

{

// TODO: данная строка кода позволяет загрузить данные в таблицу "bankDataSet.History". При необходимости она может быть перемещена или удалена.

this.historyTableAdapter.Fill(this.bankDataSet.History);

}

}

}

Форма номер 3 (Коммунальные платежи)

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Windows.Forms;

using System.Data.OleDb;

namespace Дастало

{

public partial class FormCommunalPaid : Form

{

public FormCommunalPaid()

{

InitializeComponent();

}

public string typeTransact, numAccount; //номера текущего счета и счета для перевода

ClassProviders provider;

ClassBankServer server;

string accountProvider, personalAccount; //номер счета поставщика и лицевой счет

int idProvider;

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

{

if (typeTransact == "communal") //при оплате коммунальныъ платежей

{

this.Text = "Оплата коммунальных услуг";

}

if (typeTransact == "energy") //при оплате электроэнергии

{

this.Text = "Оплата электроэнергии";

}

provider = new ClassProviders();

OleDbDataReader dr = provider.FillProviders(typeTransact); //получение списка поставщиков по типу предоставляемых услуг

DataTable prov = new DataTable(); //создание таблицы для связывания с ней элементов выпадащего списка

prov.Columns.Add("id"); //колонка код поставщика

prov.Columns.Add("name"); //колонка наименование

while (dr.Read())

{

prov.Rows.Add(Convert.ToInt32(dr.GetValue(0)), dr.GetValue(1).ToString()); //заполнение таблицы

}

cbProvider.DisplayMember = "name"; //отображаемые члены - наименование

cbProvider.ValueMember = "id"; //члены значений - код

cbProvider.DataSource = prov; //связывание таблицы с элементами списка

}

private void cbProvider\_SelectedIndexChanged(object sender, EventArgs e) //при смене поставщика

{

idProvider = Convert.ToInt32(cbProvider.SelectedValue); //получаем код поставщика

accountProvider = provider.FindAccountProvider(idProvider); //получаем номер банковского счета этого поставщика

lAccount.Text = accountProvider; //выводим номер на форму

}

private void btCheck\_Click(object sender, EventArgs e) //проверить лицевой счет клиента

{

lClient.Text = ""; //очитска полей ФИО и адреса клиента

lAddress.Text = "";

personalAccount = tbPersonalAccount.Text;

OleDbDataReader dr = provider.CheckPersonalAccount(personalAccount); //вызов метода проверки лицевого счета

while (dr.Read())

{

lClient.Text = dr.GetValue(0).ToString(); //заполнение полей ФИО и адреса клиента

lAddress.Text = dr.GetValue(1).ToString();

}

if (lClient.Text == "") MessageBox.Show("Указанный лицевой счет не найден");

}

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

{

}

private void btPaid\_Click(object sender, EventArgs e) //выполнить перевод

{

server = new ClassBankServer();

double sum = Convert.ToDouble(tbSum.Text); //сумма перевода

DateTime date = DateTime.Now; //текущая дата

if (lAddress.Text == "") btCheck\_Click(sender, e);

if (sum <= server.GetBalance(numAccount)) //проверка баланса, продолжать, если на остатке достаточно средств для операции

{

server.GetCash(numAccount, sum); //вызов метода для изменения остака на карте

server.InsertToHistory(accountProvider, date, -sum, personalAccount, typeTransact); //вызов метода для записи операции в историю

MessageBox.Show("Оплата услуг успешно выполнена");

}

else MessageBox.Show("Недостаточно средств на счете");

}

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

{

Close(); //закрыть форму

}

}

}

Листинг кода класса «Сервер банка»

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.IO;

using System.Data.OleDb;

using System.Windows.Forms;

namespace Дастало

{

class ClassBankServer

{

OleDbDataReader dr;

int idAccount;

OleDbDataReader SqlSelectCard(string numCard) //получение информации по карте

{

OleDbConnection conn = new OleDbConnection("Provider=Microsoft.ACE.OleDb.12.0;Data Source=" + Application.StartupPath.ToString() + "\\bank.accdb;Persist Security Info=False;");

conn.Open(); // отурываем БД

string sqlFindCard = "SELECT Accounts.idAccount, Clients.nameClient, Accounts.pin, Accounts.balance " +

"FROM Accounts INNER JOIN Clients ON Accounts.idClient = Clients.idClient WHERE Accounts.numAccount=@numCard"; // проводим сравнение данных из таблиц клиент и аккаунт

OleDbCommand cmd = new OleDbCommand(sqlFindCard, conn);

OleDbParameter[] prms = new OleDbParameter[1];

prms[0] = new OleDbParameter("@numCard", OleDbType.VarChar);

prms[0].Value = numCard;

cmd.Parameters.AddRange(prms);

dr = cmd.ExecuteReader();

return dr; //это завершает выполнение функции и возвращает результат работы функции в точку вызова этой функции

}

public bool CheckCard(string numCard) //проверка карты

{

bool access = false;

dr = SqlSelectCard(numCard);

while (dr.Read()) access = true;

return access; //это завершает выполнение функции и возвращает результат работы функции в точку вызова этой функции

}

public bool CheckPin(string numCard, string pin) //проверка пин-кода

{

bool access = false;

dr = SqlSelectCard(numCard);

while (dr.Read())

{

if (pin == dr.GetValue(2).ToString()) access = true;

}

return access; //это завершает выполнение функции и возвращает результат работы функции в точку вызова этой функции

}

public double GetBalance(string numCard) //проверка баланса

{

double balance = 0;

dr = SqlSelectCard(numCard);

while (dr.Read())

{

balance = Convert.ToDouble(dr.GetValue(3));

idAccount = Convert.ToInt32(dr.GetValue(0));

}

return balance; //это завершает выполнение функции и возвращает результат работы функции в точку вызова этой функции

}

public void GetCash(string numCard, double sum) //выдача денег - изменение баланса по карте

{

//перезаписать данные о счете,

OleDbConnection conn = new OleDbConnection("Provider=Microsoft.ACE.OleDb.12.0;Data Source=" + Application.StartupPath.ToString() + "\\bank.accdb;Persist Security Info=False;");

conn.Open();

string sqlUpdateBalance = "UPDATE Accounts SET balance=(balance-@sum) WHERE numAccount=@numCard";

OleDbCommand cmd = new OleDbCommand(sqlUpdateBalance, conn);

OleDbParameter[] prms = new OleDbParameter[2];

prms[0] = new OleDbParameter("@sum", OleDbType.Double);

prms[0].Value = sum;

prms[1] = new OleDbParameter("@numCard", OleDbType.VarChar);

prms[1].Value = numCard;

cmd.Parameters.AddRange(prms);

cmd.ExecuteNonQuery();

conn.Close(); // завершаем сеанс работы

conn.Dispose();

}

public void Transact(string numCard, double sum) //зачисление средств на счет при денежном переводе другому клиенту

{

OleDbConnection conn = new OleDbConnection("Provider=Microsoft.ACE.OleDb.12.0;Data Source=" + Application.StartupPath.ToString() + "\\bank.accdb;Persist Security Info=False;");

conn.Open();

string sqlUpdateBalance = "UPDATE Accounts SET balance=(balance+@sum) WHERE numAccount=@numCard";

OleDbCommand cmd = new OleDbCommand(sqlUpdateBalance, conn);

OleDbParameter[] prms = new OleDbParameter[2];

prms[0] = new OleDbParameter("@sum", OleDbType.Double);

prms[0].Value = sum;

prms[1] = new OleDbParameter("@numCard", OleDbType.VarChar);

prms[1].Value = numCard;

cmd.Parameters.AddRange(prms);

cmd.ExecuteNonQuery();

conn.Close(); // завершаем сеанс работы

conn.Dispose();

}

public void InsertToHistory(string transactAccount, DateTime dateTransact, double sumTransact, string personalAccount, string typeTransact) //добавить запись об операции в историю

{

OleDbConnection conn = new OleDbConnection("Provider=Microsoft.ACE.OleDb.12.0;Data Source=" + Application.StartupPath.ToString() + "\\bank.accdb;Persist Security Info=False;");

conn.Open();

string sqlInsertHistory = "INSERT INTO History(idAccount, transactAccount, dateTransact, sumTransact, personalAccount, typeTransact) " +

"VALUES (@idAccount, @transactAccount, @dateTransact, @sumTransact, @personalAccount, @typeTransact) ";

OleDbCommand cmd = new OleDbCommand(sqlInsertHistory, conn);

OleDbParameter[] prms = new OleDbParameter[6];

prms[0] = new OleDbParameter("@idAccount", OleDbType.Integer);

prms[0].Value = idAccount;

prms[1] = new OleDbParameter("@transactAccount", OleDbType.VarChar);

prms[1].Value = transactAccount;

prms[2] = new OleDbParameter("@dateTransact", OleDbType.Date);

prms[2].Value = dateTransact;

prms[3] = new OleDbParameter("@sumTransact", OleDbType.Double);

prms[3].Value = sumTransact;

prms[4] = new OleDbParameter("@personalAccount", OleDbType.VarChar);

prms[4].Value = personalAccount;

prms[5] = new OleDbParameter("@typeTransact", OleDbType.VarChar);

prms[5].Value = typeTransact;

cmd.Parameters.AddRange(prms);

cmd.ExecuteNonQuery();

conn.Close();

conn.Dispose();

}

public OleDbDataReader History(DateTime dateStart, DateTime dateEnd) //получение истории операций по карте

{

OleDbConnection conn = new OleDbConnection("Provider=Microsoft.ACE.OleDb.12.0;Data Source=" + Application.StartupPath.ToString() + "\\bank.accdb;Persist Security Info=False;");

conn.Open();

string sqlSelectHistory = "SELECT History.dateTransact, History.transactAccount, History.sumTransact, History.typeTransact " +

"FROM History WHERE History.idAccount=@idAccount AND History.dateTransact>=@dateStart AND History.dateTransact<=@dateEnd";

OleDbCommand cmd = new OleDbCommand(sqlSelectHistory, conn);

OleDbParameter[] prms = new OleDbParameter[3];

prms[0] = new OleDbParameter("@idAccount", OleDbType.VarChar);

prms[0].Value = idAccount;

prms[1] = new OleDbParameter("@dateStart", OleDbType.Date);

prms[1].Value = dateStart;

prms[2] = new OleDbParameter("@dateEnd", OleDbType.Date);

prms[2].Value = dateEnd;

cmd.Parameters.AddRange(prms);

dr = cmd.ExecuteReader();

return dr; //это завершает выполнение функции и возвращает результат работы функции в точку вызова этой функции

}

public string GetName(string typeTransact, string transactAccount) //получение наименования владельца счета для перевода средств

{

string name = "";

OleDbConnection conn = new OleDbConnection("Provider=Microsoft.ACE.OleDb.12.0;Data Source=" + Application.StartupPath.ToString() + "\\bank.accdb;Persist Security Info=False;");

conn.Open();

string sqlSelectName = "";

if (typeTransact == "transact") //денежные переводы другому клиенту банка

{

sqlSelectName = "SELECT Clients.nameClient FROM Accounts INNER JOIN Clients ON Accounts.idClient = Clients.idClient WHERE Accounts.numAccount=@transactAccount";

}

if (typeTransact == "communal" || typeTransact == "energy") //коммунальные услуги

{

sqlSelectName = "SELECT nameProvider FROM Providers WHERE accountProvider=@transactAccount";

}

OleDbCommand cmd = new OleDbCommand(sqlSelectName, conn);

OleDbParameter[] prms = new OleDbParameter[1];

prms[0] = new OleDbParameter("@transactAccount", OleDbType.VarChar);

prms[0].Value = transactAccount;

cmd.Parameters.AddRange(prms);

try

{

dr = cmd.ExecuteReader();

while (dr.Read()) name = dr.GetValue(0).ToString();

}

catch { name = "Мобильная связь"; }

return name; //это завершает выполнение функции и возвращает результат работы функции в точку вызова этой функции

}

}

}

### ***2.4. Описание программы.***

Данный подпункт оформлен согласно ГОСТ 19.402-78 ЕСПД и содержит следующие подразделы:

- Общие сведения – Программа написана на 2-х языках: в среде Microsoft Visual Studio и Access.

- Функциональное назначение – Данная программа предназначена для пользователей банкомата и помогает им легче справляться с работой.

- Используемые технические средства – Программа создавалась на устройстве с техническими характеристиками:

* Видеокамера на внешней стороне устройства — предназначена для фиксации лица человека, совершающего операцию.
* Считывающее устройство — кардридер.
* Компьютер и система безопасности.
* Принтер — отвечает за печать чеков по каждой операции.
* Кассета (кассеты) с купюрами из банка — из них банкомат осуществляет выдачу наличных денежных средств.
* Кассета (кассеты) в которые помещаются наличные, внесенные в банкомат.
* Механизм для забора денег из кассеты и их последующей выдачи.
* Клавиатуру и сенсорный экран для ввода PIN-кода.
* Модем – для отправки запроса и принятия его из банка.

- Вызов и загрузка – Программа запускается с флешки.

- Входные данные – Данные хранятся на флешке в формате Visual ST и Access данных

- Выходные данные - Данные хранятся на флешке в формате Visual ST и Access данных

### ***2.5. Тестирование программы.***

**Microsoft Visual Studio**- линейка продуктов компании Microsoft, включающих интегрированную среду разработки программного обеспечения и ряд других инструментальных средств. Данные продукты позволяют разрабатывать как консольные приложения, так и приложения с графическим интерфейсом, в том числе с поддержкой технологии Windows Forms, а также веб-сайты, веб-приложения, веб-службы как в родном, так и в управляемом кодах для всех платформ, поддерживаемых Windows, Windows Mobile, Xbox, и Silverlight, а так же Android и iOS (Xamarin).

При разработке программы использовалась среда программирования. Эта среда представляет собой набор средств быстрой разработки и отладки приложений для Windows. При разработке приложения применялся визуальный редактор форм, редактор программного кода, палитра компонентов и инспектор объектов.

Основными операциями в процессе разработки являлись: создание форм, размещение компонентов, редактирование свойств этих компонентов, формирование визуального пользовательского интерфейса, написание кода программы для обработки событий, сохранение и компиляция проекта.

Помимо указанных операций применялись дополнительные операции для отладки созданного приложения. Для этого в составе С# имеется интегральный отладчик, который позволяет обнаруживать и исправлять ошибки как на стадии написания кода, так и на стадии отладки работающей программы.

Кроме стандартных средств отладки применялась методика «силовой отладки» приложения. В код программы вставлялся дополнительный код, ставящий своей целью вывод промежуточных значений во время расчетов.

При испытании программы ставилась цель проверить работоспособность программы, корректность расчетов и отсутствие логических ошибок. Также ставилась задача исключить «человеческий фактор» возникновения исключительных ситуаций и ошибок, за счет применения методик «защиты от дурака».

Результатом тестирования и отладки стало обнаружение нескольких «узких мест» в работе программы, которые могли бы ухудшить эксплуатационные характеристики программы. Все эти недостатки были исправлены или заменены наиболее оптимальными решениями.

## **Заключение.**

Подводим итоги, я написал программу для банкомата. Данная система будет применятся Банками для удобства работы с клиентами и как каждое разработанное приложение у него могут быть изъяны, но благодаря отзывам клиентов каждый изъян можно исправить со временем и сделать приложение более идеальным.

## **Список использованной литературы.**

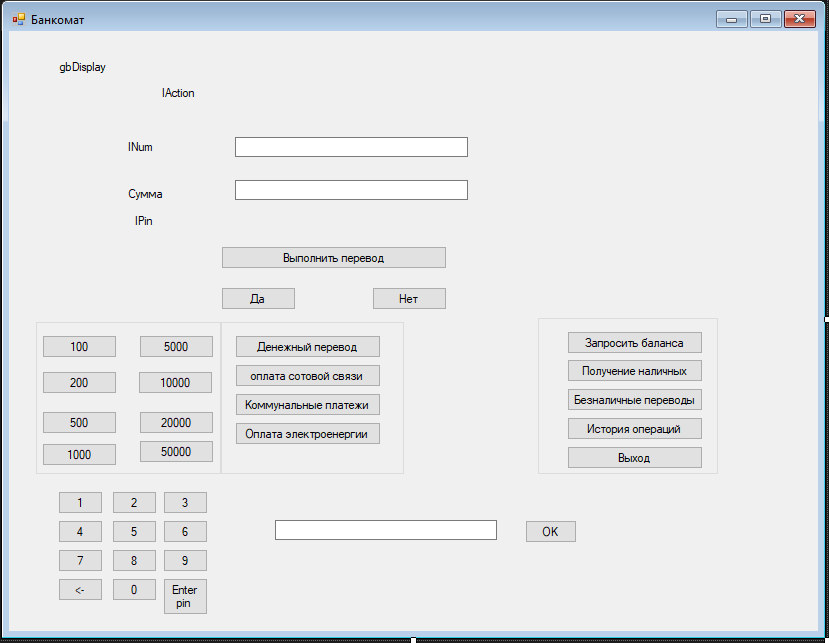
1. <https://brobank.ru/kak-ustroen-bankomat/>
2. [**https://ru.wikipedia.org/wiki/Банкомат**](https://ru.wikipedia.org/wiki/Банкомат)
3. [**https://wmcash24.com/zachem-nuzhny-bankomaty/**](https://wmcash24.com/zachem-nuzhny-bankomaty/)
4. [**https://www.papabankir.ru/tips/kak-rabotaet-bankomat/**](https://www.papabankir.ru/tips/kak-rabotaet-bankomat/)
5. [**https://habr.com/ru/company/gazprombank/blog/654235/**](https://habr.com/ru/company/gazprombank/blog/654235/)

ГОСТЫ:

1. ГОСТ 19.402-78 ЕСПД
2. ГОСТ 19.701-90 ЕСПД
3. ГОСТ 19.202-78 ЕСПД

## **Приложения.**

Форма номер 1 главная (Банкомат)



Листинг кода Форма номер 1 главная (Банкомат)

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Windows.Forms;

using System.IO;

using System.Data.OleDb;

namespace Дастало

{

public partial class Банкомат : Form

{

ClassBankServer server; //экземпляр класса для связи с сервером

public Банкомат()

{

InitializeComponent();

}

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

{

lAction.Text = "Вставьте карту";

server = new ClassBankServer();

tbCard.PasswordChar = '\*'; //скрыть вводимые символы номера карты

}

string numAccount = ""; //номер счета

bool access; //переменная, определяющая, дан ли доступ данной карте

private void btEnterCard\_Click(object sender, EventArgs e) //ввод карты

{

numAccount = tbCard.Text;

access = server.CheckCard(numAccount); //связь с сервром: проверка карты

if (access == true) //если карта есть в базе

{ //ввести пин-код

lAction.Text = "Введите PIN-код";

lPin.Text = "PIN: ";

gbDisplay.Visible = true;

lPin.Visible = true;

pin = "";

string numCard = "\*\*\*\*\*\*\*\*" + numAccount.Substring(8); //номер карты, где отображены только последние 4 цифры

gbDisplay.Text = numCard; //отобразить его на форме в углу дисплея

}

else

{

lAction.Text = "Данная карта недействительна";

}

}

string pin = ""; //введенный пин-код

void DisplayPIN(string code) //метод отображения на форме вводимого пин-кода

{

string pinCode = "PIN: ";

if (code.Length <= 4) //максимальная длина кода - 4 символа

{

foreach (char ch in code)

pinCode += "\*"; //отображать на форме в виде \*

lPin.Text = pinCode;

}

}

//

private void bt1\_Click(object sender, EventArgs e) //при нажатии кнопки на цифровом блоке

{

if (tbCardTansact.Visible == true && mtbMobile.Visible == false) //если отображена форма ввода номера карты для перевода

{

tbCardTansact.Text += "1"; //добавить цифру в поле номера карты

}

else if (tbCardTansact.Visible == false && mtbMobile.Visible == false) //если панель ввода не отображена

{ //происходит ввод пин-кода

if (access == true && pin.Length < 4) pin += "1"; //если карта действительна и длина введенного пин-кода меньше 4 символов, добавить цифру к пин-коду

DisplayPIN(pin); //выполнить функцию отображения пин-кода на дисплее

}

}

int attempt = 3; //количество попыток ввода пин-кода

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

{

access = server.CheckPin(numAccount, pin); //проверка правильности пин-кода

if (access == true) //если пин верен

{ //отобразить на форме элементы управления

lAction.Text = "Выберите действие"; //для совершения операций по карте

lPin.Visible = false;

gbButtons.Visible = true;

}

else

{ //если пин введен неверно

lAction.Text = "Неверный PIN-код";

pin = "";

DisplayPIN(pin);

attempt--; //уменьшение количества доступных попыток

if (attempt > 0) MessageBox.Show("Осталось попыток: " + attempt.ToString());

else

{ //если пин введен неверно 3 раза

MessageBox.Show("PIN-код неверно введен 3 раза. Обслуживание прекращено");

btExit\_Click(sender, e); //прекратить обслуживание и отобразить первоначальную форму

}

access = true;

}

}

private void bt0\_Click(object sender, EventArgs e)

{

if (tbCardTansact.Visible == true && mtbMobile.Visible == false)

{

tbCardTansact.Text += "0";

}

else if (tbCardTansact.Visible == false && mtbMobile.Visible == false)

{

if (access == true && pin.Length < 4) pin += "0";

DisplayPIN(pin);

}

}

private void btCorrect\_Click(object sender, EventArgs e) //кнопка корректировки введенных символов

{

if (tbCardTansact.Visible == true && mtbMobile.Visible == false) //корректировка номера карты для перевода

{

tbCardTansact.Text = tbCardTansact.Text.Remove(tbCardTansact.TextLength - 1);

}

else if (tbCardTansact.Visible == false && mtbMobile.Visible == false) //корректировка пин-кода

{

if (access == true) pin = pin.Remove(pin.Length - 1);

DisplayPIN(pin);

}

}

private void bt9\_Click(object sender, EventArgs e)

{

if (tbCardTansact.Visible == true && mtbMobile.Visible == false)

{

tbCardTansact.Text += "9";

}

else if (tbCardTansact.Visible == false && mtbMobile.Visible == false)

{

if (access == true && pin.Length < 4) pin += "9";

DisplayPIN(pin);

}

}

private void bt8\_Click(object sender, EventArgs e)

{

if (tbCardTansact.Visible == true && mtbMobile.Visible == false)

{

tbCardTansact.Text += "8";

}

else if (tbCardTansact.Visible == false && mtbMobile.Visible == false)

{

if (access == true && pin.Length < 4) pin += "8";

DisplayPIN(pin);

}

}

private void bt7\_Click(object sender, EventArgs e)

{

if (tbCardTansact.Visible == true && mtbMobile.Visible == false)

{

tbCardTansact.Text += "7";

}

else if (tbCardTansact.Visible == false && mtbMobile.Visible == false)

{

if (access == true && pin.Length < 4) pin += "7";

DisplayPIN(pin);

}

}

private void bt6\_Click(object sender, EventArgs e)

{

if (tbCardTansact.Visible == true && mtbMobile.Visible == false)

{

tbCardTansact.Text += "6";

}

else if (tbCardTansact.Visible == false && mtbMobile.Visible == false)

{

if (access == true && pin.Length < 4) pin += "6";

DisplayPIN(pin);

}

}

private void bt5\_Click(object sender, EventArgs e)

{

if (tbCardTansact.Visible == true && mtbMobile.Visible == false)

{

tbCardTansact.Text += "5";

}

else if (tbCardTansact.Visible == false && mtbMobile.Visible == false)

{

if (access == true && pin.Length < 4) pin += "5";

DisplayPIN(pin);

}

}

private void bt4\_Click(object sender, EventArgs e)

{

if (tbCardTansact.Visible == true && mtbMobile.Visible == false)

{

tbCardTansact.Text += "4";

}

else if (tbCardTansact.Visible == false && mtbMobile.Visible == false)

{

if (access == true && pin.Length < 4) pin += "4";

DisplayPIN(pin);

}

}

private void bt3\_Click(object sender, EventArgs e)

{

if (tbCardTansact.Visible == true && mtbMobile.Visible == false)

{

tbCardTansact.Text += "3";

}

else if (tbCardTansact.Visible == false && mtbMobile.Visible == false)

{

if (access == true && pin.Length < 4) pin += "3";

DisplayPIN(pin);

}

}

private void bt2\_Click(object sender, EventArgs e)

{

if (tbCardTansact.Visible == true && mtbMobile.Visible == false)

{

tbCardTansact.Text += "2";

}

else if (tbCardTansact.Visible == false && mtbMobile.Visible == false)

{

if (access == true && pin.Length < 4) pin += "2";

DisplayPIN(pin);

}

}

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

{

lAction.Text = "";

lPin.Visible = true;

lPin.Text = "Остаток по счету: \n" + server.GetBalance(numAccount).ToString() + " рублей"; //связь с сервером: проверка баланса

btYes.Visible = false;

btNo.Visible = false;

gbSum.Visible = false;

pTransact.Visible = false;

pTransact.Visible = false;

}

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

{

lPin.Visible = false; //отобразить на форме элементы

lAction.Text = "Выберите сумму"; //выбора суммы для получения

gbSum.Visible = true;

pTransact.Visible = false;

pTransact.Visible = false;

}

DateTime dateTransact;

double givenSum = 0; //выданная сумма

void GetCash(int sum) //выдача наличных

{

access = false;

givenSum = 0;

dateTransact = DateTime.Now;

lAction.Text = "Получить " + sum.ToString() + " рублей";

if (sum <= server.GetBalance(numAccount)) access = true; //связь с сервером: проверка наличия требуемой суммы на счете

if (access == true) //если такая сумма имеется

{

lPin.Visible = true; //отображение на форме

lPin.Text = "Распечатать квитанцию?"; //предложения распечать чек

btYes.Visible = true;

btNo.Visible = true;

string given = "Выдано: ";

int count5tr = sum / 5000; //количесто выдаваемых купюр по 5000

int count = GetCountBanknotes("5000");

if (count >= count5tr)

{

givenSum += count5tr \* 5000;

sum -= count5tr \* 5000; //вычислить остаток, который выдавать другими купюрами

given += ("5000 - " + count5tr.ToString() + ", ");

}

else if (count < count5tr)

{

givenSum += count \* 5000;

sum -= count \* 5000; //вычислить остаток, который выдавать другими купюрами

given += ("5000 - " + count.ToString() + ", ");

count5tr = count;

}

int count1tr = sum / 1000;

count = GetCountBanknotes("1000");

if (count >= count1tr)

{

givenSum += count1tr \* 1000;

sum -= count1tr \* 1000; //вычислить остаток, который выдавать другими купюрами

given += ("1000 - " + count1tr.ToString() + ", ");

}

else if (count < count1tr)

{

givenSum += count \* 1000;

sum -= count \* 1000; //вычислить остаток, который выдавать другими купюрами

given += ("1000 - " + count.ToString() + ", ");

count1tr = count;

}

int count5sr = sum / 500;

count = GetCountBanknotes("500");

if (count >= count5sr)

{

givenSum += count5sr \* 500;

sum -= count5sr \* 500; //вычислить остаток, который выдавать другими купюрами

given += ("500 - " + count5sr.ToString() + ", ");

}

else if (count < count5sr)

{

givenSum += count \* 500;

sum -= count5sr \* 500; //вычислить остаток, который выдавать другими купюрами

given += ("500 - " + count.ToString() + ", ");

count5sr = count;

}

int count1sr = sum / 100;

count = GetCountBanknotes("100");

if (count >= count1sr)

{

givenSum += count1sr \* 100;

sum -= count1sr \* 100; //вычислить остаток, который выдавать другими купюрами

given += ("100 - " + count1sr.ToString() + ", ");

}

else if (count < count1sr)

{

givenSum += count \* 100;

sum -= count1sr \* 100; //вычислить остаток

given += ("100 - " + count.ToString() + ", ");

count1sr = count;

}

if (sum != 0) //в итоге остаток должен стать равен 0

{ //если этого не случилось, значит в хранилище нет каких-либо купюр

MessageBox.Show("Невозможно выдать требумую сумму"); //в достаточном количестве

lPin.Visible = false;

btYes.Visible = false;

btNo.Visible = false;

}

else

{

UpdateBanknotes("5000", count5tr);

UpdateBanknotes("1000", count1tr);

UpdateBanknotes("500", count5sr);

UpdateBanknotes("100", count1sr);

server.GetCash(numAccount, givenSum);

server.InsertToHistory("", dateTransact, -givenSum, "", "getCash");

MessageBox.Show(given); //отобразить сообщение, какими купюрами в каком количестве

} //выдана данная сумма

}

else // если на балансе нет трубуемой суммы

{

lPin.Visible = true;

lPin.Text = "Недостаточно денег";

}

}

int GetCountBanknotes(string denomination)

{

int count = 0;

OleDbConnection conn = new OleDbConnection("Provider=Microsoft.ACE.OleDb.12.0;Data Source=" + Application.StartupPath.ToString() + "\\bank.accdb;Persist Security Info=False;");

conn.Open();

string sqlFindCard = "SELECT Banknotes.countBanknotes FROM Banknotes WHERE denomination=@denomination";

OleDbCommand cmd = new OleDbCommand(sqlFindCard, conn);

OleDbParameter[] prms = new OleDbParameter[1];

prms[0] = new OleDbParameter("@denomination", OleDbType.VarChar);

prms[0].Value = denomination;

cmd.Parameters.AddRange(prms);

OleDbDataReader dr = cmd.ExecuteReader();

while (dr.Read()) count = Convert.ToInt32(dr.GetValue(0));

return count;

}

void UpdateBanknotes(string denomination, int count)

{

OleDbConnection conn = new OleDbConnection("Provider=Microsoft.ACE.OleDb.12.0;Data Source=" + Application.StartupPath.ToString() + "\\bank.accdb;Persist Security Info=False;");

conn.Open();

string sqlUpdateBalance = "UPDATE Banknotes SET countBanknotes=(countBanknotes-@count) WHERE denomination=@denomination";

OleDbCommand cmd = new OleDbCommand(sqlUpdateBalance, conn);

OleDbParameter[] prms = new OleDbParameter[2];

prms[0] = new OleDbParameter("@count", OleDbType.Integer);

prms[0].Value = count;

prms[1] = new OleDbParameter("@denomination", OleDbType.VarChar);

prms[1].Value = denomination;

cmd.Parameters.AddRange(prms);

cmd.ExecuteNonQuery();

conn.Close();

conn.Dispose();

}

private void bt100r\_Click(object sender, EventArgs e)

{

GetCash(100);

}

private void bt200r\_Click(object sender, EventArgs e)

{

GetCash(200);

}

private void bt500r\_Click(object sender, EventArgs e)

{

GetCash(500);

}

private void bt1tr\_Click(object sender, EventArgs e)

{

GetCash(1000);

}

private void bt5tr\_Click(object sender, EventArgs e)

{

GetCash(5000);

}

private void bt10tr\_Click(object sender, EventArgs e)

{

GetCash(10000);

}

private void bt20tr\_Click(object sender, EventArgs e)

{

GetCash(20000);

}

private void bt50tr\_Click(object sender, EventArgs e)

{

GetCash(50000);

}

private void btYes\_Click(object sender, EventArgs e) //печать чека

{

string date = DateTime.Now.ToString();

string numCard = "\*\*\*\*\*\*\*\*" + numAccount.Substring(8); //номер карты, где отображены только последние 4 цифры

string print = "Время оперции: " + date +

"\n\nНомер счета: " + numCard + //распечать номер счета

"\nВыдано: " + givenSum + //выданную сумму

"\nОстаток: " + server.GetBalance(numAccount).ToString(); //и остаток по карте

MessageBox.Show(print);

btNo\_Click(sender, e); //отобразить элементы для выполнения новой операции

}

private void btNo\_Click(object sender, EventArgs e) //не печатать чек

{

lAction.Text = "Выберите действие";

lPin.Visible = false;

gbSum.Visible = false;

btNo.Visible = false;

btYes.Visible = false;

pTransact.Visible = false;

gbSum.Visible = false;

}

private void btExit\_Click(object sender, EventArgs e) //выход, прекращение обслуживания

{ //отображение первоначальной формы

lAction.Text = "Вставьте карту";

lPin.Visible = false;

attempt = 3;

gbButtons.Visible = false;

pTransact.Visible = false;

pTransact.Visible = false;

tbCard.Clear();

tbCardTansact.Clear();

tbSum.Clear();

mtbMobile.Clear();

pin = "";

gbDisplay.Text = "";

}

string transact = ""; //переменная для определения типа перевода

private void btTransact\_Click(object sender, EventArgs e) //денежный перевод

{

lAction.Text = "Денежный перевод";

lPin.Visible = false;

gbSum.Visible = false;

lNum.Visible = true;

//отобразить панель для ввода номера карты и суммы для перевода

pTransact.Visible = true;

mtbMobile.Visible = false;

tbCardTansact.Visible = true;

tbSum.Visible = true;

Summ.Visible = true;

btRunTransact.Visible = true;

lNum.Text = "Номер карты";

transact = "card";

}

//выполнить перевод

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

{

//денежный перевод на карту

if (transact == "card")

{

string numCard = tbCardTansact.Text; //номер карты

double sum = Convert.ToDouble(tbSum.Text); //сумма

dateTransact = DateTime.Now;

access = server.CheckCard(numCard); //проверка указанной карты

if (access == true) //если карта найдена

{

if (sum <= server.GetBalance(numAccount))

{

server.GetCash(numAccount, sum);

server.Transact(numCard, sum); //зачисление средств на указанный счет

server.InsertToHistory(numAccount, dateTransact, -sum, "", "transact");

double s = server.GetBalance(numCard);

server.InsertToHistory(numCard, dateTransact, sum, "", "transact");

MessageBox.Show("Перевод успешно выполнен");

pTransact.Visible = false;

lAction.Text = "Выберите действие";

}

else

{

MessageBox.Show("Недостаточно денег на счете");

}

}

else MessageBox.Show("Номер карты не найден");

}

//оплата сотовой связи

if (transact == "tel")

{

string tel = mtbMobile.Text; //номер телефона

tbSum.Visible = true;

double sum = Convert.ToDouble(tbSum.Text); //сумма пополнения

if (sum <= server.GetBalance(numAccount))

{

server.GetCash(numAccount, sum);

server.InsertToHistory(tel, dateTransact, -sum, "", "tel");

MessageBox.Show("Пополнение счета " + tel + " успешно выполнено");

pTransact.Visible = false;

lAction.Text = "Выберите действие";

}

else

{

MessageBox.Show("Недостаточно денег на счете");

}

}

}

//оплата сотовой связи

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

{

lAction.Text = "Оплата сотовой связи";

lPin.Visible = false;

gbSum.Visible = false;

tbSum.Visible = true;

lNum.Visible = true;

Summ.Visible = true;

btRunTransact.Visible = true;

//отобразить панель для ввода номера телефона и суммы для пополнения

pTransact.Visible = true;

mtbMobile.Visible = true;

tbCardTansact.Visible = false;

lNum.Text = "Номер телефона";

transact = "tel";

}

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

{

lAction.Text = "Выберите вид перевода";

lPin.Visible = false;

gbSum.Visible = false;

pTransact.Visible = true;

}

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

{

FormCommunalPaid fcp = new FormCommunalPaid();

fcp.typeTransact = "communal";

fcp.numAccount = numAccount;

fcp.Show();

}

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

{

FormCommunalPaid fcp = new FormCommunalPaid();

fcp.typeTransact = "energy";

fcp.numAccount = numAccount;

fcp.Show();

}

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

{

lPin.Visible = false;

gbSum.Visible = false;

pTransact.Visible = false;

FormHistory fh = new FormHistory();

fh.numAccount = numAccount;

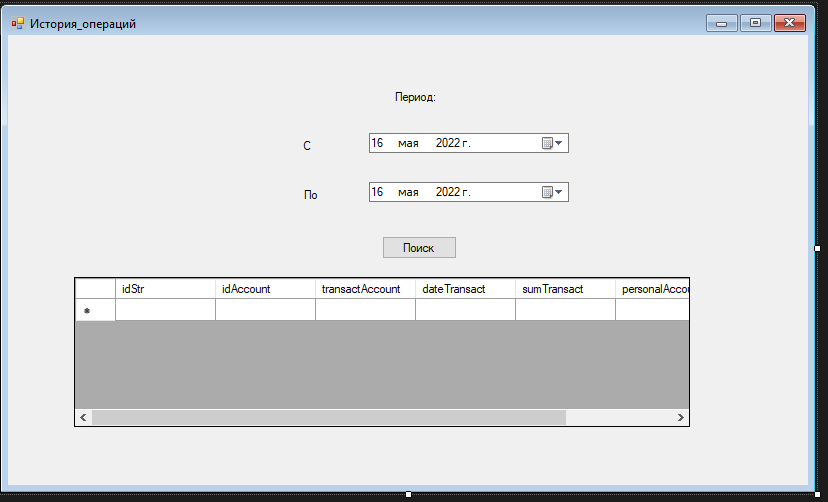
fh.Show();

}

}

}

Форма номер 2 (История операций)



using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Windows.Forms;

using System.Data.OleDb;

namespace Дастало

{

public partial class FormHistory : Form

{

public FormHistory()

{

InitializeComponent();

}

public string numAccount; //номер счета

private void btOK\_Click\_1(object sender, EventArgs e)

{

dgvHistory.Rows.Clear(); //очистить таблицу

DateTime dateStart = dtDateEnd.Value.Date; //дата начала периода

DateTime dateEnd = dtDateEnd.Value.Date; //дата окончания периода для выборки

ClassBankServer server = new ClassBankServer();

server.GetBalance(numAccount); //определение id счета

OleDbDataReader dr = server.History(dateStart, dateEnd); //вызов метода истории операций по карте

string name; //наименование поставщика

while (dr.Read())

{

if (dr.GetValue(1).ToString() == "") //если номер счета для перевода отсутствует (были сняты наличные)

{

dgvHistory.Rows.Add(dr.GetValue(0).ToString(), "", "Выдача наличных", dr.GetValue(2).ToString()); //заполнение строки таблицы

}

else

{

name = server.GetName(dr.GetValue(3).ToString(), dr.GetValue(1).ToString()); //получение наименования поставщика

dgvHistory.Rows.Add(dr.GetValue(0).ToString(), dr.GetValue(1).ToString(), name, dr.GetValue(2).ToString()); //заполнение строк таблицы

}

}

}

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

{

// TODO: данная строка кода позволяет загрузить данные в таблицу "bankDataSet.History". При необходимости она может быть перемещена или удалена.

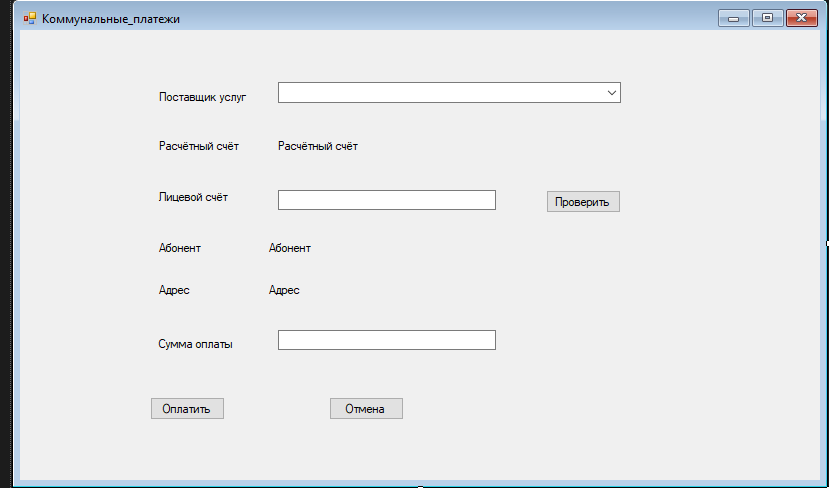
this.historyTableAdapter.Fill(this.bankDataSet.History);

}

}

}

Форма номер 3 (Коммунальные платежи)



using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Windows.Forms;

using System.Data.OleDb;

namespace Дастало

{

public partial class FormCommunalPaid : Form

{

public FormCommunalPaid()

{

InitializeComponent();

}

public string typeTransact, numAccount; //номера текущего счета и счета для перевода

ClassProviders provider;

ClassBankServer server;

string accountProvider, personalAccount; //номер счета поставщика и лицевой счет

int idProvider;

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

{

if (typeTransact == "communal") //при оплате коммунальныъ платежей

{

this.Text = "Оплата коммунальных услуг";

}

if (typeTransact == "energy") //при оплате электроэнергии

{

this.Text = "Оплата электроэнергии";

}

provider = new ClassProviders();

OleDbDataReader dr = provider.FillProviders(typeTransact); //получение списка поставщиков по типу предоставляемых услуг

DataTable prov = new DataTable(); //создание таблицы для связывания с ней элементов выпадащего списка

prov.Columns.Add("id"); //колонка код поставщика

prov.Columns.Add("name"); //колонка наименование

while (dr.Read())

{

prov.Rows.Add(Convert.ToInt32(dr.GetValue(0)), dr.GetValue(1).ToString()); //заполнение таблицы

}

cbProvider.DisplayMember = "name"; //отображаемые члены - наименование

cbProvider.ValueMember = "id"; //члены значений - код

cbProvider.DataSource = prov; //связывание таблицы с элементами списка

}

private void cbProvider\_SelectedIndexChanged(object sender, EventArgs e) //при смене поставщика

{

idProvider = Convert.ToInt32(cbProvider.SelectedValue); //получаем код поставщика

accountProvider = provider.FindAccountProvider(idProvider); //получаем номер банковского счета этого поставщика

lAccount.Text = accountProvider; //выводим номер на форму

}

private void btCheck\_Click(object sender, EventArgs e) //проверить лицевой счет клиента

{

lClient.Text = ""; //очитска полей ФИО и адреса клиента

lAddress.Text = "";

personalAccount = tbPersonalAccount.Text;

OleDbDataReader dr = provider.CheckPersonalAccount(personalAccount); //вызов метода проверки лицевого счета

while (dr.Read())

{

lClient.Text = dr.GetValue(0).ToString(); //заполнение полей ФИО и адреса клиента

lAddress.Text = dr.GetValue(1).ToString();

}

if (lClient.Text == "") MessageBox.Show("Указанный лицевой счет не найден");

}

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

{

}

private void btPaid\_Click(object sender, EventArgs e) //выполнить перевод

{

server = new ClassBankServer();

double sum = Convert.ToDouble(tbSum.Text); //сумма перевода

DateTime date = DateTime.Now; //текущая дата

if (lAddress.Text == "") btCheck\_Click(sender, e);

if (sum <= server.GetBalance(numAccount)) //проверка баланса, продолжать, если на остатке достаточно средств для операции

{

server.GetCash(numAccount, sum); //вызов метода для изменения остака на карте

server.InsertToHistory(accountProvider, date, -sum, personalAccount, typeTransact); //вызов метода для записи операции в историю

MessageBox.Show("Оплата услуг успешно выполнена");

}

else MessageBox.Show("Недостаточно средств на счете");

}

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

{

Close(); //закрыть форму

}

}

}