Содержание:

Введение

Банкомат – это устройство, автоматизирующее операции по выдаче и переводу денег, хранящихся в банке, лицу, которому они принадлежат. Идентификация каждого клиента происходит с помощью имеющейся у него кредитной карты банка и соответствующего карте секретного pin-кода. Поэтому человек может вне банка снимать деньги или оплачивать определенные услуги. Так как банк заинтересован в том, чтобы человек хранил в нем свои деньги, то он должен обеспечивать удобство работы своим клиентам. При этом, если банкомат будет не надежным, то страдать будет как банк, так и его клиенты, а это совершенно неприемлемая ситуация.

Целью данной работы является разработка приложения, имитирующего модель работы банкомата. Для выполнения поставленной цели необходимо решить следующие задачи:

- сформулировать требования к системе и описать их в виде алгоритмов;

- спроектировать базу данных приложения;

- спроектировать структуру приложения, разработать диаграмму классов;

- разработать пользовательский интерфейс;

- разработать программный код;

- отладить приложение.

1. Формирование требований

1.1. Задание

Спроектировать и реализовать приложение «Модель работы банкомата». Приложение должно обеспечивать ввод номера карты, ввод PIN-кода, запрос

баланса, выдачу наличных, осуществление безналичных платежей, печать квитанций.

1.2. Назначение и область применения

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.

Обслуживание клиента начинается с момента помещения пластиковой карточки в банкомат. После распознавания типа пластиковой карточки, банкомат выдает на дисплей приглашение ввести персональный код. Персональный код представляет собой четырехзначное число. Затем банкомат проверяет правильность введенного кода, сверяя с кодом, хранящимся на карте. Если код указан неверно, пользователю предоставляются еще две попытки для ввода правильного кода. В случае повторных неудач карта перемещается в хранилище карт, и сеанс обслуживания заканчивается. После ввода правильного кода банкомат предлагает пользователю выбрать операцию. Клиент может либо снять наличные со счета, либо узнать остаток на его счету, либо осуществить безналичный платеж со своего счёта.

При снятии наличных со счета банкомат предлагает указать сумму. После выбора клиентом суммы банкомат запрашивает, нужно ли печатать справку по операции. Затем банкомат посылает запрос на снятие выбранной суммы центральному компьютеру банка. В случае получения разрешения на операцию, банкомат проверяет, имеется ли требуемая сумма в его хранилище денег, и может ли он выдать запрошенную сумму банкнотами, имеющимися в наличии (например, если в банкомате нет банкнот достоинством менее 1000 рублей, он не может выдать никакую сумму не кратную 1000). После удаления карточки из приемника, банкомат выдает указанную сумму в лоток выдачи. Банкомат печатает справку по произведенной операции, если она была затребована клиентом.

Если клиент хочет узнать остаток на счету, то банкомат посылает запрос центральному компьютеру банка и выводит сумму на дисплей. По требованию клиента печатается и выдается соответствующая справка.

1.3. Состав требований

Приложение «Модель работы банкомата» должно реализовывать следующие функции:

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. 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.

1.4. Описание алгоритма

Для более наглядного отображения требований к информационной системе для каждой реализуемой функции составлен и описан подробный алгоритм.

1. Проверка PIN-кода карты.

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. 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 данныхPIN-кода карты по её номеру, затем выполняется сравнение полученногоPIN-кода и введенного пользователем. Если данные коды не совпадают, пользователь оповещается сообщением о неверно введенномPIN-коде. При совпадении кодов пользователю предлагается выбрать дальнейшую операцию по карте.

На рисунке 1 изображено графическое представление данного алгоритма.

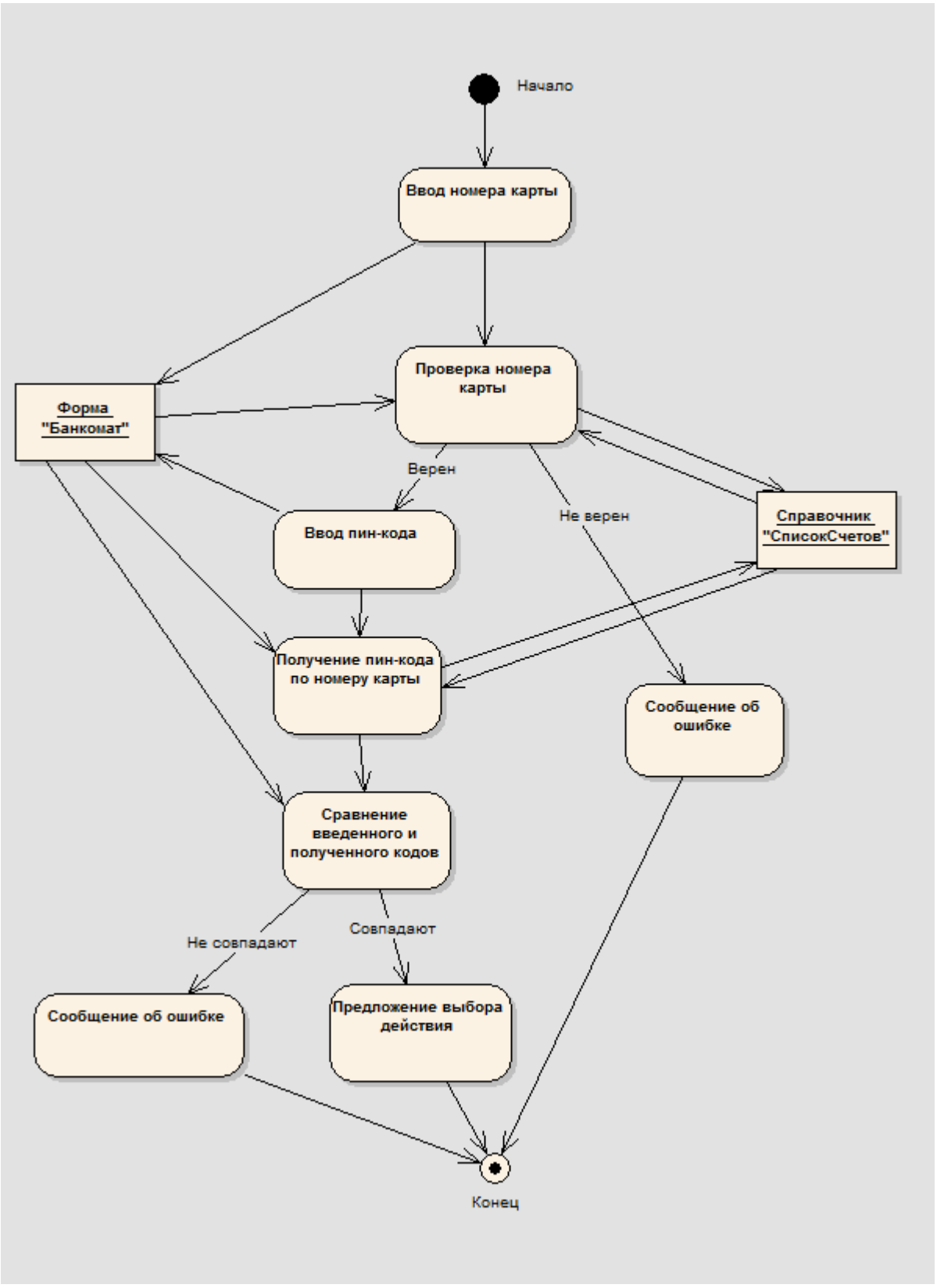

Рисунок 1 – Алгоритм функции «Проверка PIN-кода карты»

2. Предоставление сведений об остатке на счете

После выбора пользователем действия «Запросить баланс» выполняется поиск данных по указанному номеру счета. Когда соответствующая запись найдена, из нее получается значение остатка по счету. Затем полученное значение остатка отображается на дисплее банкомата.

На рисунке 2 изображено графическое представление данного алгоритма.

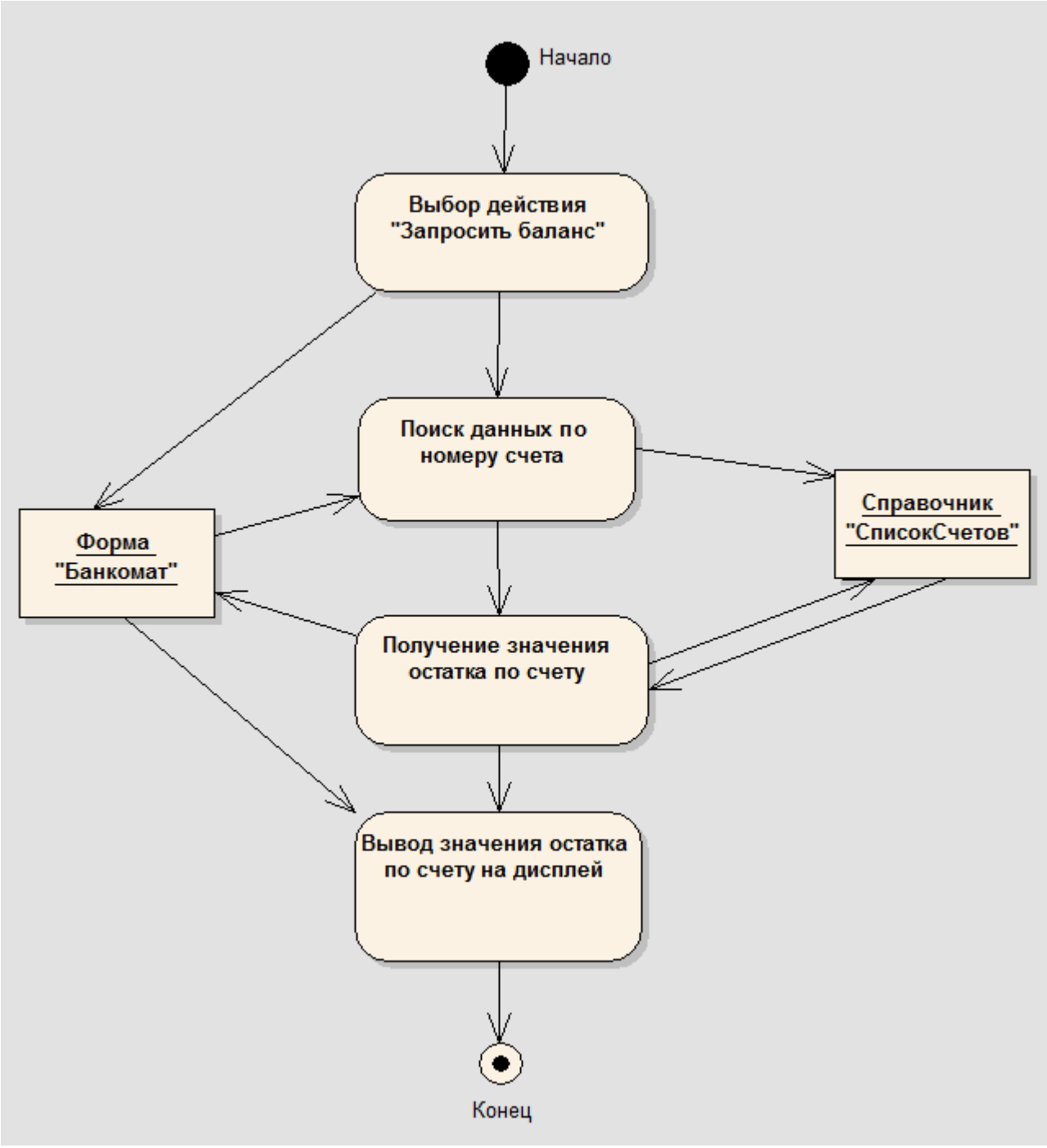

Рисунок 2 - Алгоритм функции «Предоставление сведений об остатке на счете»

3. Выдача наличных.

Когда пользователь выбирает операцию «Получить наличные», банкомат предлагает выбрать требуемую сумму для выдачи: 100, 200, 500, 1000, 5000, 10000, 20000, 50000. После выбора суммы выполняется поиск данных по указанному номеру счета. Когда соответствующая запись найдена, из нее получается значение остатка по счету. Затем выполняется проверка, является ли остаток по счету достаточным для выдачи требуемой суммы. Если остаток меньше требуемой суммы, пользователь оповещается об этом сообщением о недостаточном количестве средств на счете.

Если значение остатка больше значения запрашиваемой суммы, банкомат выполняет проверку наличия купюр для выдачи указанной сумы в хранилище купюр. Если данную сумму находящимися в хранилище купюрами выдать невозможно, пользователь оповещается об этом сообщением. Если такая возможность существует, пользователю выдаются наличные. При этом происходит списание выданной суммы со счета, т.е. изменяется значение остатка по счету. Также выполняется изменение количества купюр соответствующего номинала в хранилище купюр. Производится запись о выполнении операции в историю операций. На дисплей выводится сообщение, какими купюрами и в каком количестве выдана указанная сумма. После выдачи наличных пользователю предлагается распечатать квитанцию.

На рисунке 3 изображено графическое представление данного алгоритма.

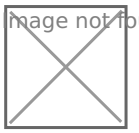

Рисунок 3 – Алгоритм функции «Выдача наличных»

4. Выполнение денежного перевода.

Когда пользователь выбирает операцию «Денежный перевод», на дисплее банкомата отображается форма ввода номера карты, на которую осуществляется перевод, и суммы перевода. Затем банкомат отправляет запрос на сервер для проверки указанного номера счета. Если в базе такой счет не найден, пользователь оповещается сообщением об ошибке. Если указанный номер счета найден в списке счетов, выполняется проверка баланса текущего счета, достаточно ли средств для выполнения перевода. Если сумма на остатке по счету меньше указанной суммы,

пользователь оповещается сообщением об ошибке. Если сумма на остатке достаточна для выполнения операции, производится списание указанной суммы с текущего счета и зачисление этой суммы на указанный счет.Производится запись о выполнении операции в историю операций.

На рисунке 4 изображено графическое представление данного алгоритма.

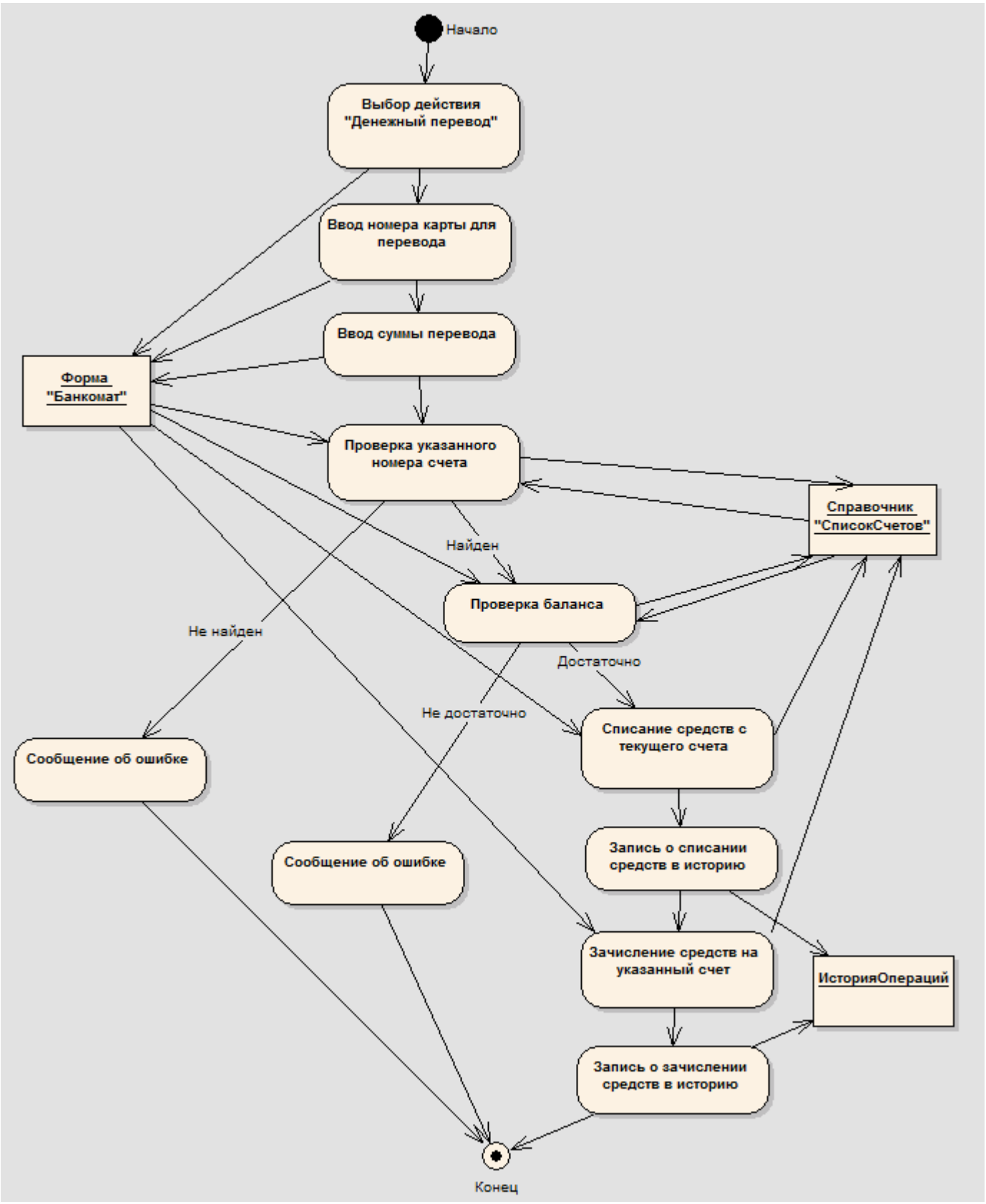

Рисунок 4 – Алгоритм функции «Выполнение денежного перевода»

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. Далее банкомат отправляет запрос на сервер для проверки баланса текущего счета. Если сумма на остатке по счету меньше указанной суммы, пользователь оповещается сообщением об ошибке. Если сумма на остатке достаточна для выполнения операции, производится списание указанной суммы с текущего счета.

На рисунке 5 изображено графическое представление данного алгоритма.

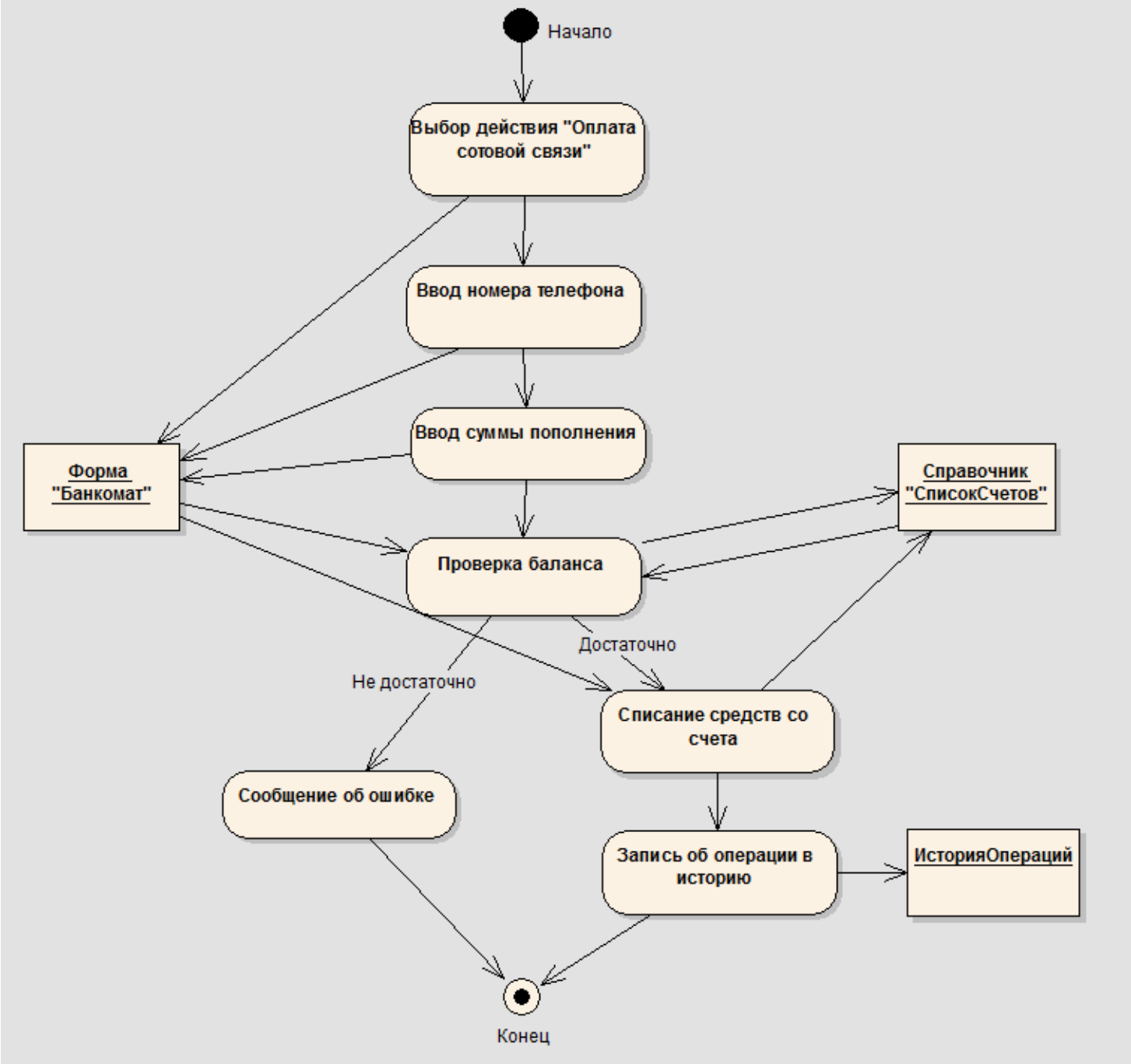

Рисунок 5 – Алгоритм функции «Оплата услуг сотовой связи»

6. Оплата коммунальных платежей.

Когда пользователь выбирает операцию «Коммунальные платежи» или «Оплата электроэнергии», на экран выводится форма с элементами ввода требуемых данных. Сначала необходимо выбрать поставщика услуг из списка, который формируется в зависимости от типа платежа. После выбора поставщика происходит обращение к базе и получение номера банковского счета поставщика, который отображается на форме. Далее пользователь вводит номер своего персонального счета у данного поставщика. Есть возможность проверить правильность введенного персонального счета нажатием на кнопку «Проверить», банкомат обратится к серверу и получит персональные данные клиента по указанному персональному счету, которые будут отображены на экране. Если такой номер не будет найден, пользователь будет повещен сообщением об ошибке. Если проверка прошла успешно, пользователь вводит сумму оплаты и нажимает кнопку «Оплатить». Выполняется проверка баланса, если сумма достаточна для платежа, производится списание средств со счета, и добавляется запись об операции в историю.

На рисунке 6 изображено графическое представление данного алгоритма.

Рисунок 6 – Алгоритм функции «Оплата коммунальных платежей»

7. Печать квитанции.

Если после выдачи наличных пользователь выбирает «Печатать квитанцию», происходит поиск данных по счету и получение значения остатка после выдачи наличных. Затем выдается квитанция, содержащая номер счета, выданную сумму и значение остатка по счету после выдачи наличных.

На рисунке 7 изображено графическое представление данного алгоритма.

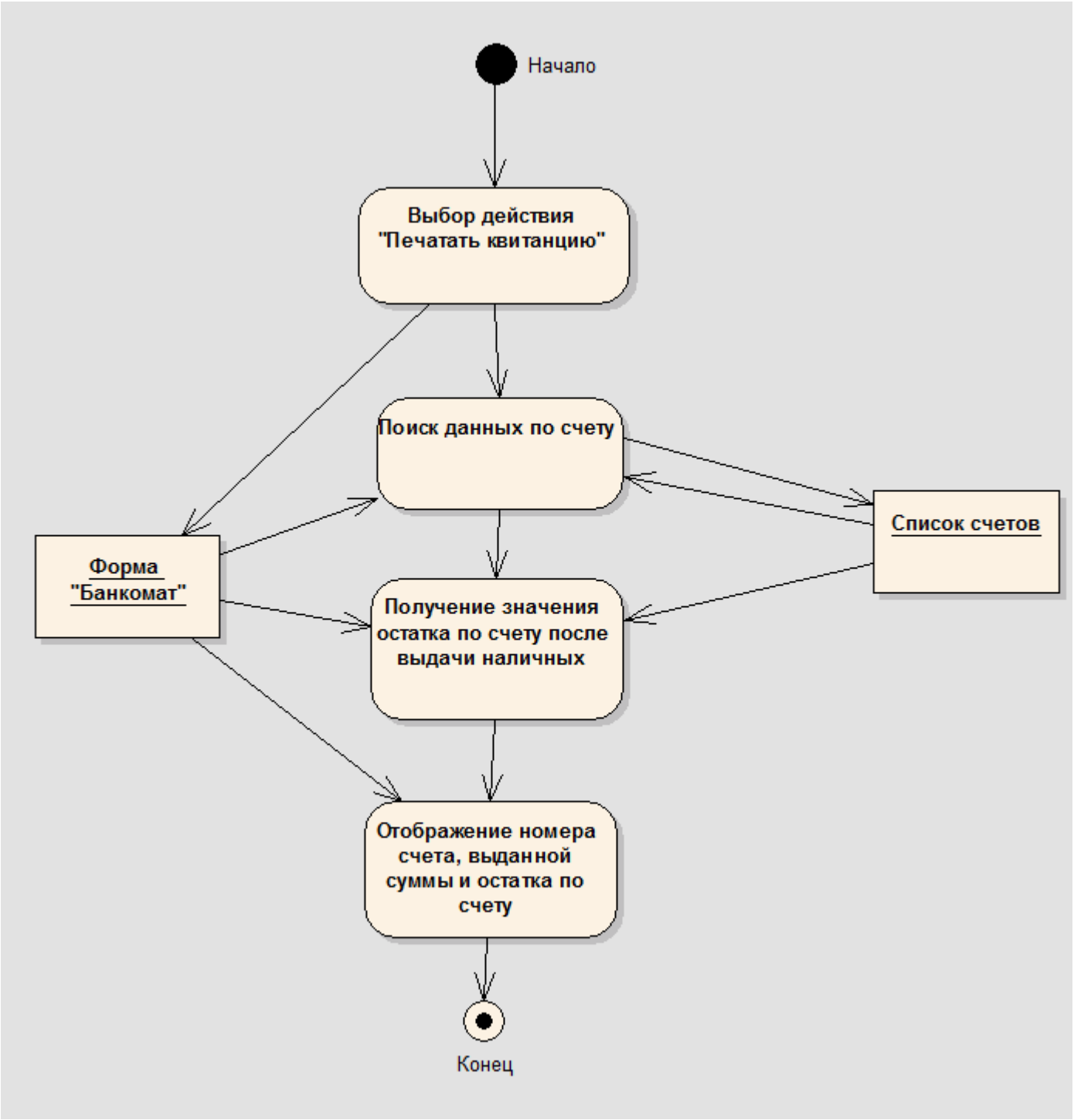

Рисунок 7 – Алгоритм функции «Печать квитанции»

8. История операций по карте.

Когда пользователь выбирает операцию «История операций», на экран выводится форма, содержащая элементы выбора даты для указания периода, за который необходимо предоставить информацию. Далее банкомат обращается к серверу и получает данные обо всех совершенных операциях по карте за указанный период.

Затем для более информативного отображения данных на форме происходит определение наименования владельца счета либо типа совершенной операции. На форме данные представлены в виде таблицы, содержащей поля «Дата», «Номер счета», «Владелец счета» и «Сумма».

На рисунке 8 изображено графическое представление данного алгоритма.

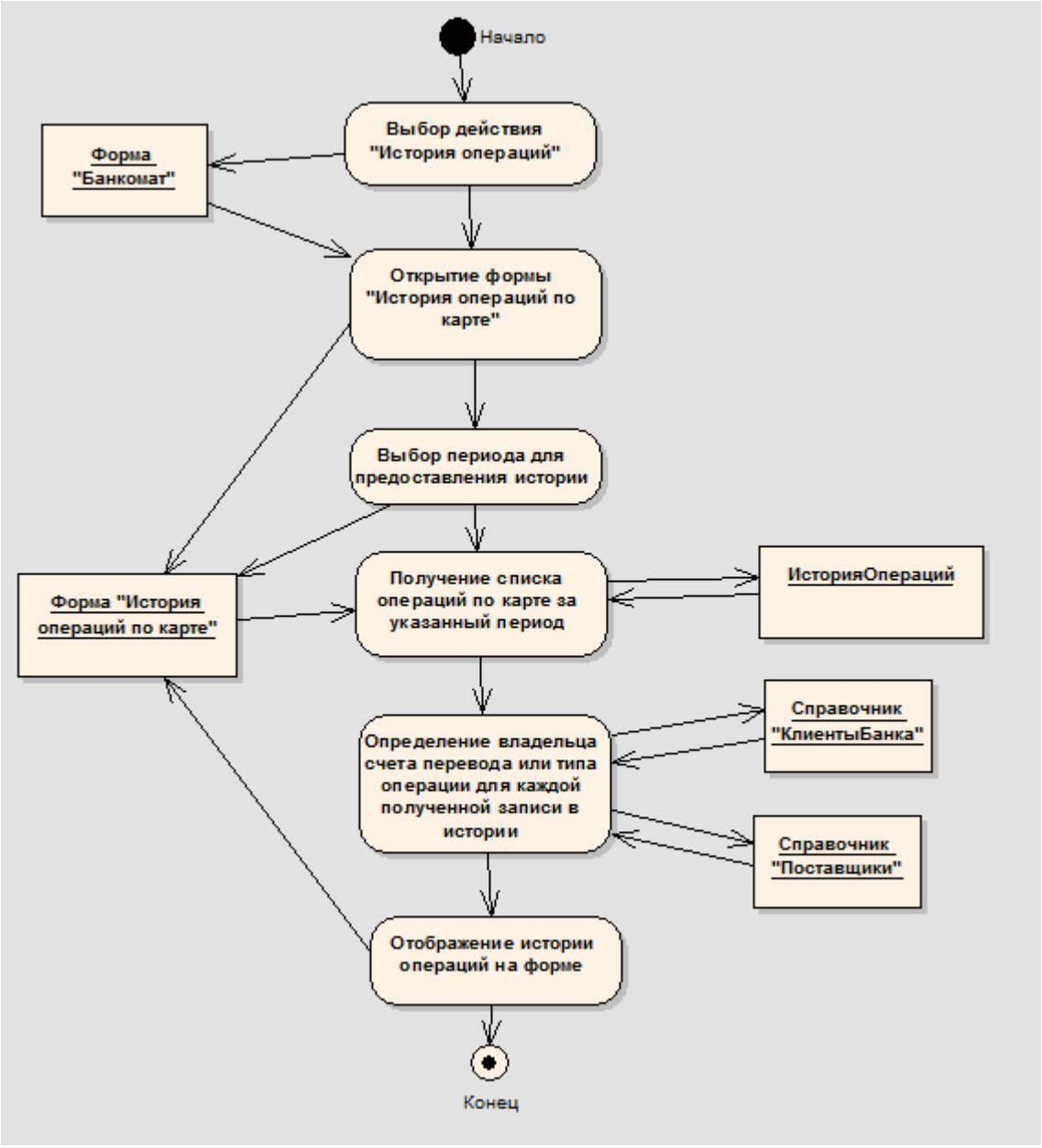

Рисунок 8 – Алгоритм функции «История операций»

1.5. Выбор состава технических и программных средств

Функционирование приложения предполагается на персональных компьютерах под управлением операционной системы Windows, версия XPSP3 или выше.

Минимальная аппаратная конфигурация системы, обеспечивающей нормальное функционирование программного обеспечения должна быть не ниже следующей:

- процессор 900 МГц и выше;
- оперативная память 256 Мбайт и выше;
- свободного места на жестком диске не менее 100 Мб.

2. Разработка рабочего проекта

2.1 Проектирование хранилища данных

В качестве СУБД для разработки системы был выбран Microsoft Access 2010.

Для реализации необходимо функционала были созданы следующие таблицы:

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.

На рисунке 9 представлена схема данных.

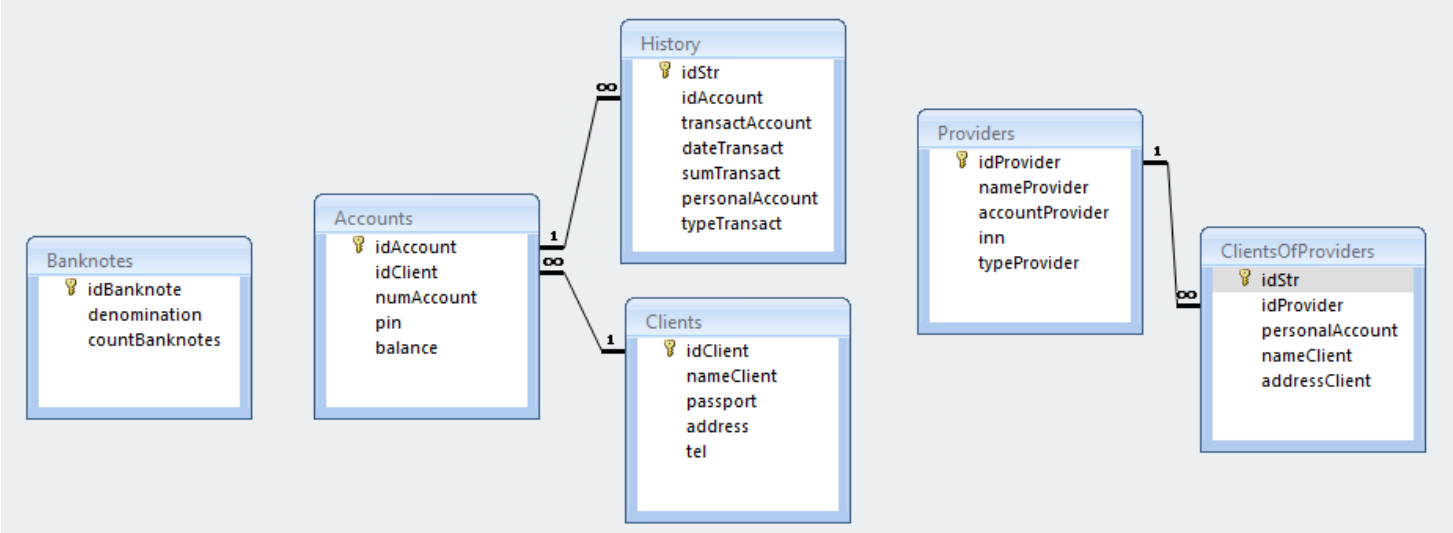

Рисунок 9 – Схема данных

2.2. Проектирование структуры приложения

Для реализации поставленных требований структура приложения «Модель работы банкомата» должна включать в себя следующие элементы.

1. Класс «Банкомат», представляющий основную форму для действий пользователя и связи с сервером банка для совершения необходимых операций. Реализует следующие функции:

- Ввод номера карты и PIN-кода;

- Предоставление пользователю выбора действий для совершения необходимых операций по карте;

- Проверка наличия купюр для выдачи – расчет необходимого количества купюр для выдачи указанной суммы и проверка хранилища купюр для определения, есть ли в наличии требуемые купюры;

- Выдача наличных – изменение значения имеющегося в наличии количества купюр соответствующего номинала с учетом выданных купюр;

- Печать квитанции – отображение на экране сообщения, содержащего номер счета, выданную сумму и остаток по счету;

2. Класс «Сервер банка», обеспечивающий подключение к базе данных банка и реализующий следующие функции:

- Проверка карты –поиск в базе данных номера карты, идентичного введенному пользователем;

- Проверка PIN-кода – получение из базы данных PIN-кода карты по номеру счета и сравнение его с введенным;

- Проверка баланса – получение из базы данных значение остатка по счету по номеру карты;

- Изменение баланса – уменьшение значения остатка по счету в базе данных;

- Зачисление средств – увеличение баланса по номеру карты при совершении денежного перевода;

- Добавление записи в историю – запись в базу данных информации о совершенной операции;

- Получение истории операций – запрос из базы данных истории операций по карте за указанный период.

3. Класс «Коммунальные платежи» - представляет форму для действий пользователя по совершении оплаты коммунальных услуг или электроэнергии. Обеспечивает связь с сервером банка и сервером поставщиков услуг и реализует следующие функции:

- Ввод всех необходимых данных для совершения платежа.

4. Класс «Сервер поставщиков услуг», обеспечивающий подключение к базе данных банка и реализующий следующие функции:

- Получение списка поставщиков услуг по категории услуг: коммунальные услуги или электроэнергия;

- Получение номера счета поставщика;

- Проверка персонального счета клиента.

5. Класс «История операций по карте» - представляет форму для выбора пользователем периода и отображения истории операций за указанный период. Обеспечивает связь с сервером банка.

6. База данных «Банк» - обеспечивает хранение информации о клиентах банка, счетах клиентов, поставщиках, клиентах поставщиков, истории операций по картам клиентов.

На рисунке 10 представлена диаграмма классов приложения «Модель работы банкомата».

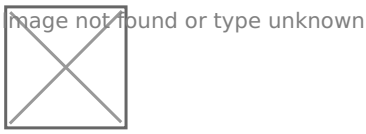

Рисунок 10 – Диаграмма классов

2.3. Разработка пользовательского интерфейса.

Графический интерфейс приложения должен предоставлять пользователю доступ ко всем вышеперечисленным функциям и реализовывать классы «Банкомат», «Коммунальные платежи» и «История операций по карте».

Графический интерфейс формы «Банкомат» содержит следующие элементы:

- приемник карт, выполненный в виде текстового поля и кнопки «OK» для ввода номера карты;

- панель с кнопками для ввода PIN-кода и номера карты для перевода средств, содержащая цифровой блок, кнопку корректировки и кнопку ввода;

- дисплей, на котором отображается информация о совершаемой операции, кнопки выбора действия («Запросить баланс», «Получить наличные», «Безналичные переводы», «История операций» «Выход»), кнопки выбора суммы для выдачи, кнопки для выбора вида безналичного перевода, кнопки для подтверждения печати чека («Да», «Нет»). Общий вид графического интерфейса представлен на рисунке 11.

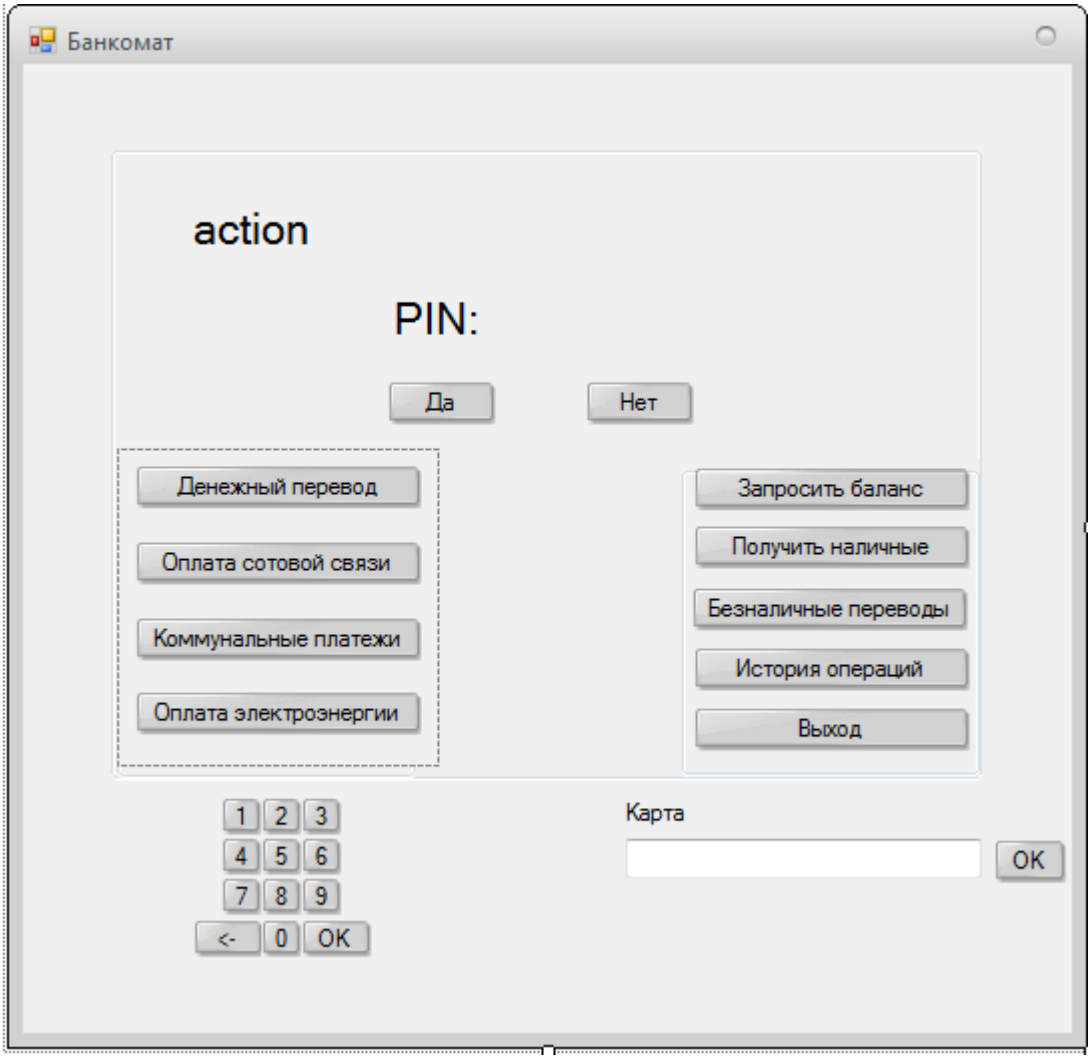

Рисунок 11 - Графический интерфейс формы «Банкомат»

- панель для ввода номера карты и суммы для выполнения денежного перевода (рисунок 12);

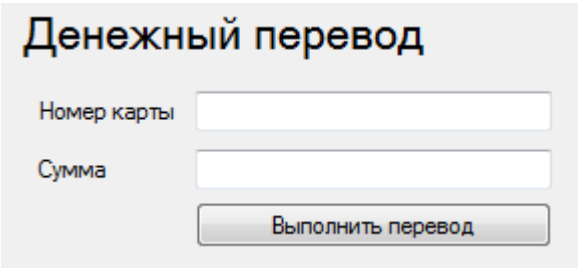

Рисунок 12 – Панель ввода данных для денежного перевода

- панель для ввода номера телефона и суммы для оплаты сотовой связи (рисунок 13).

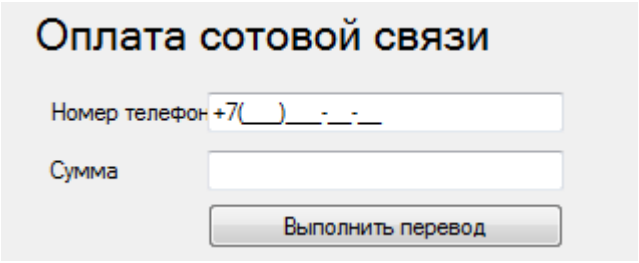

Рисунок 13 – Панель ввода данных для оплаты сотовой связи

Графический интерфейс формы «Коммунальные платежи» содержит следующие элементы:

- выпадающий список для выбора поставщика;

- элемент отображения номера счета выбранного поставщика;

- текстовое поле ввода номера лицевого счета клиента и кнопка «Проверить»;

- элементы отображения ФИО и адреса клиента;

- текстовое поле ввода суммы оплаты;

- кнопки «Оплатить» и «Отмена».

Общий вид графического интерфейса формы «Коммунальные платежи» представлен на рисунке 14.

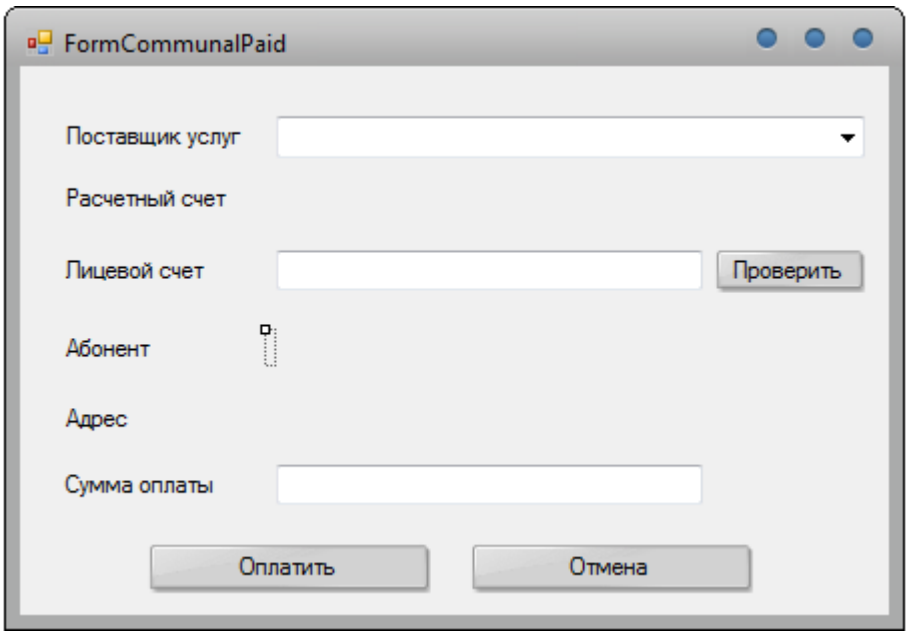

Рисунок 14 - Графический интерфейс формы «Коммунальные платежи»

Графический интерфейс формы «История операций по карте» содержит следующие элементы:

- элементы выбора даты для указания периода;

- кнопка «Поиск»;

- таблицы для отображения истории операций.

Общий вид графического интерфейса формы «Коммунальные платежи» представлен на рисунке 15.

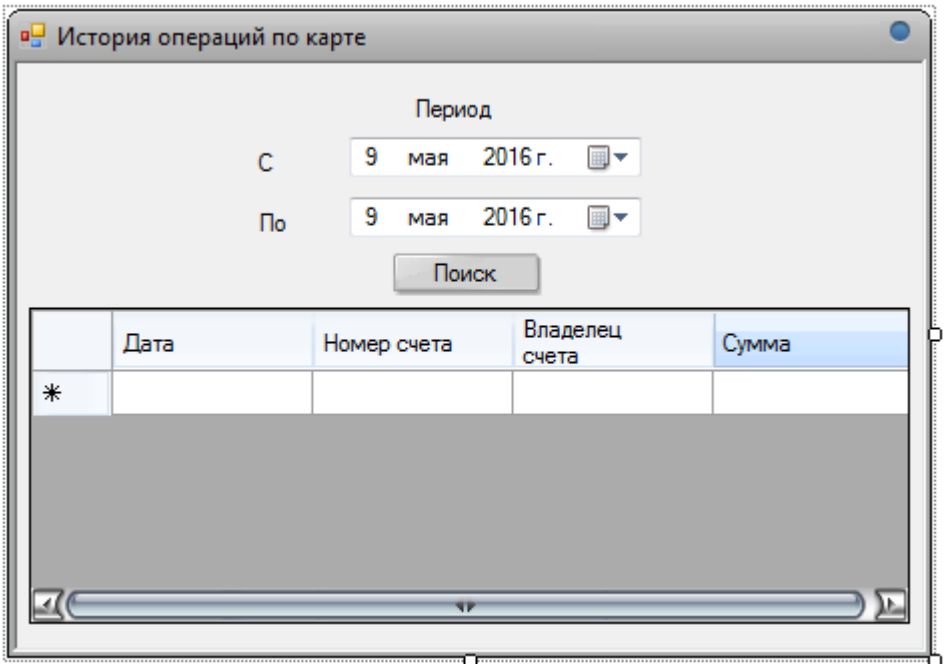

Рисунок 15 - Графический интерфейс формы «История операций по карте»

2.4. Разработка программного кода

Исходный код разрабатываемого проекта представлен в приложении 1.

2.5. Тестирование приложения.

При запуске приложения открывается первоначальная форма. На дисплее отображается надпись «Вставьте карту» (рисунок 16)

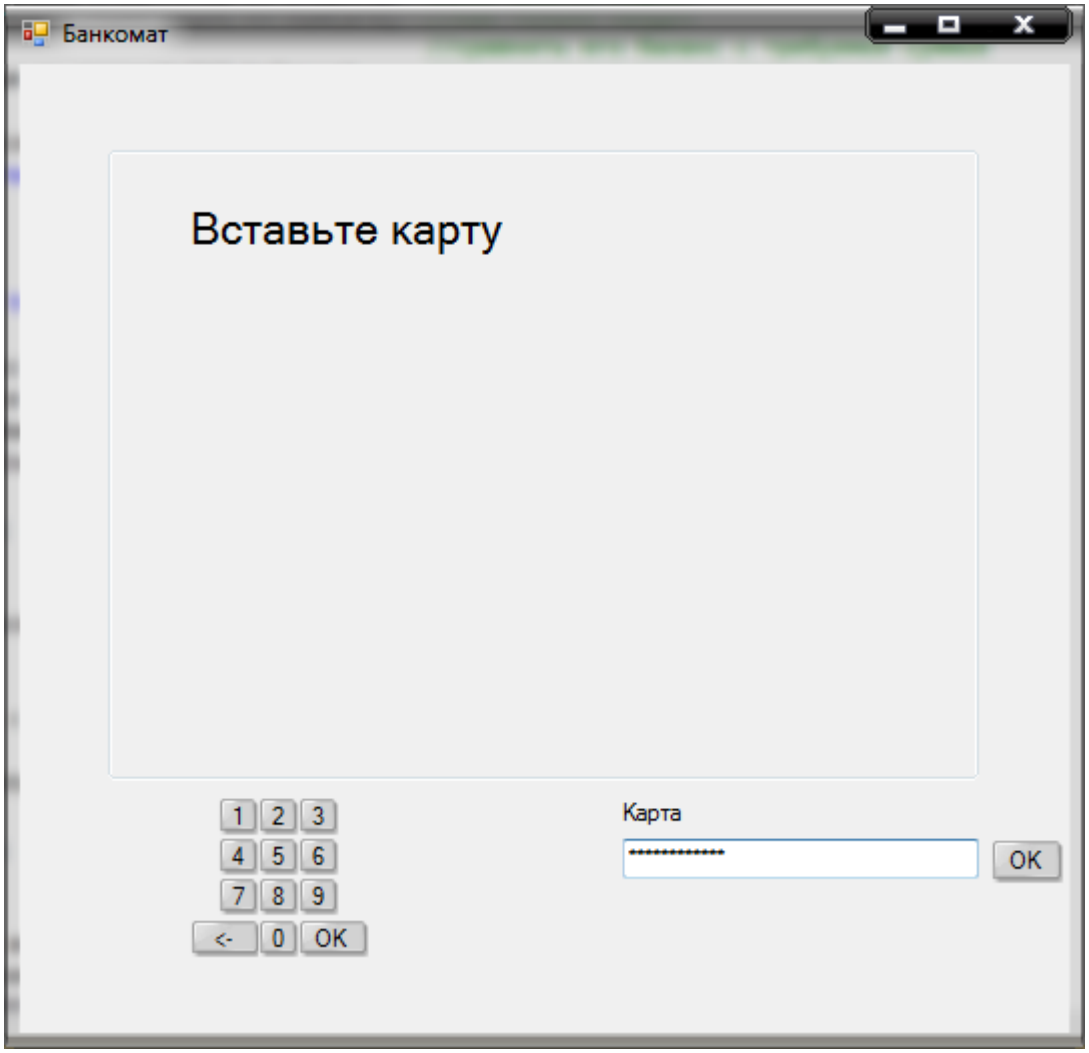

Рисунок 16 - «Вставьте карту» для начала работы

После ввода номера карты происходит проверка на сервере, является ли данная карта доступной. Если данный номер карты не найден, на дисплее отображается сообщение «Данная карта недействительна» (рисунок 17).

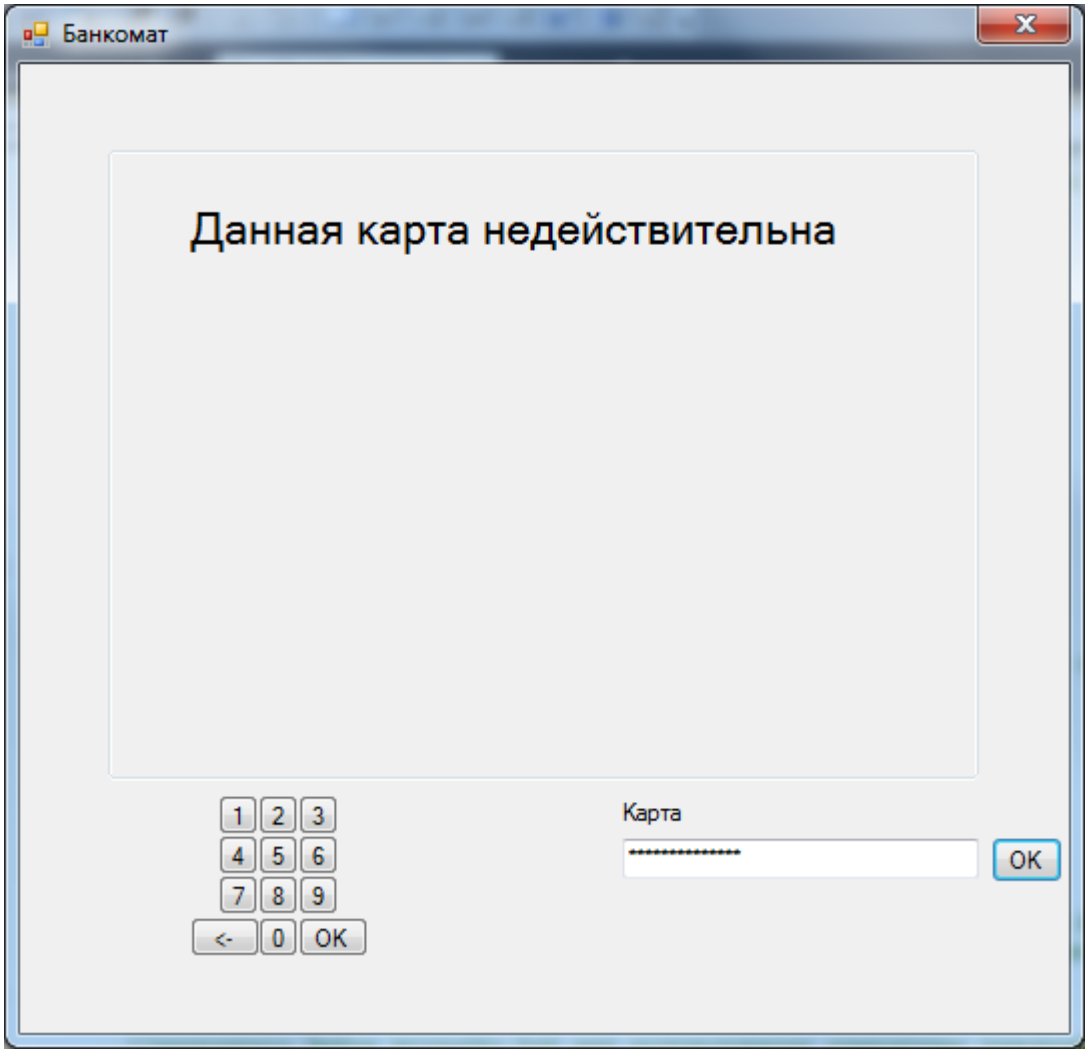

Рисунок 17 – «Данная карта недействительна»

Если карта в порядке, на дисплее отображается номер карты в виде скрытых символов и открытых 4 последних цифр и предложение ввести PIN-код (рисунок 18).

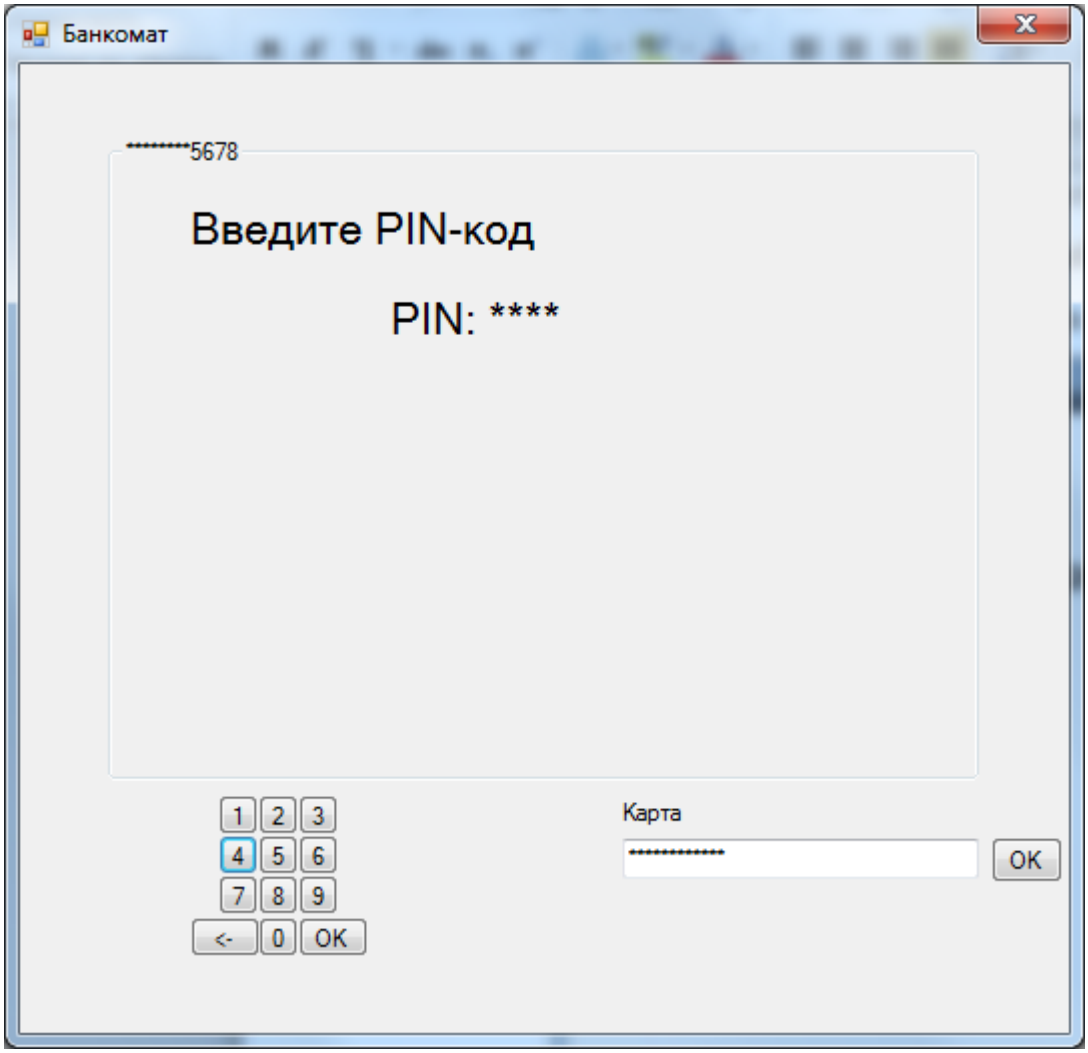

Рисунок 18 – «Введите PIN-код»

После ввода PIN-кода происходит проверка на сервере, верно ли введенPIN. При неверном вводе PIN-кода отображается сообщение об ошибке и уменьшается количество возможных попыток ввода верного PIN-кода (рисунок 19).

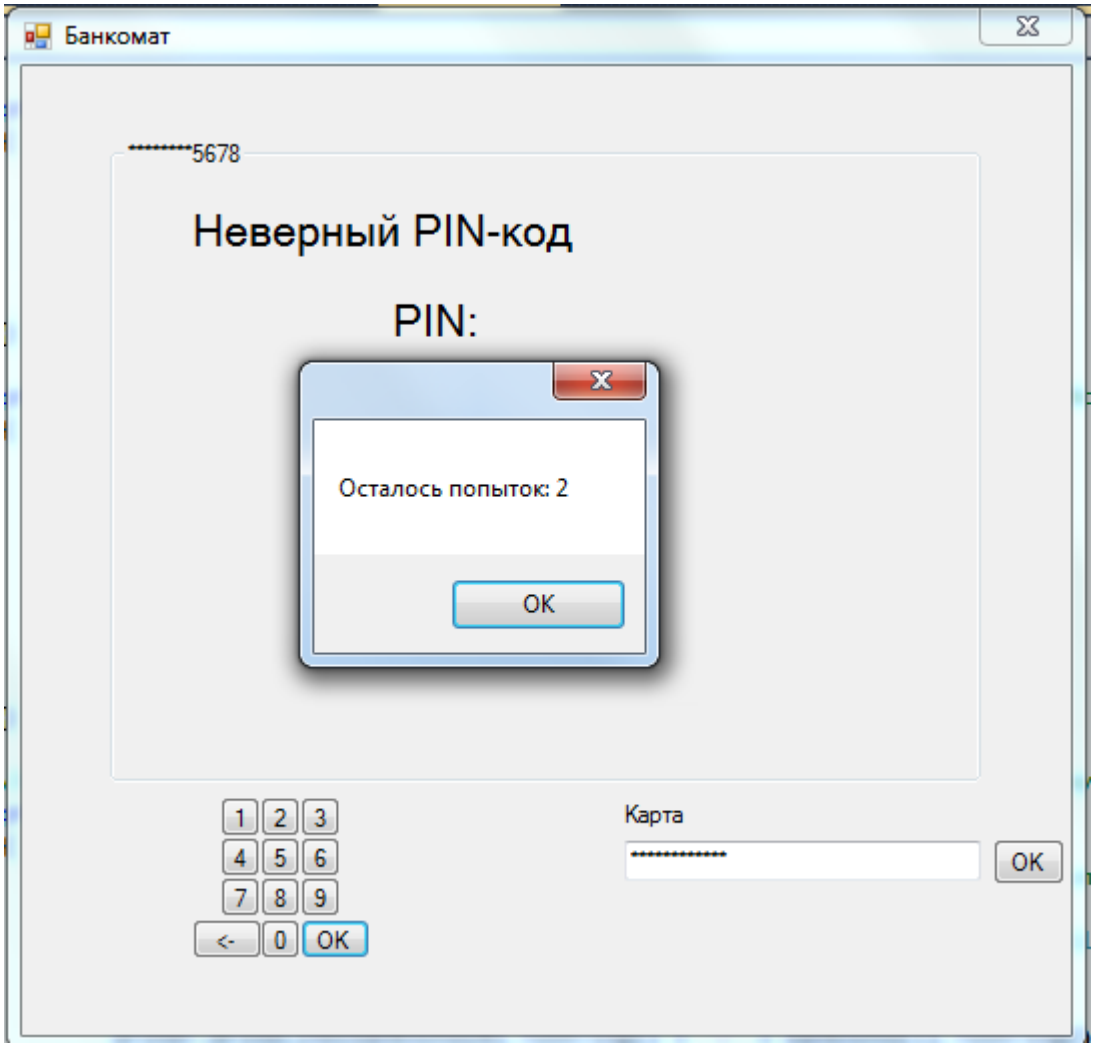

Рисунок 19 – «Неверный PIN-код»

Если PIN-код введен верно, на дисплее отображаются кнопки выбора дальнейшего действия.

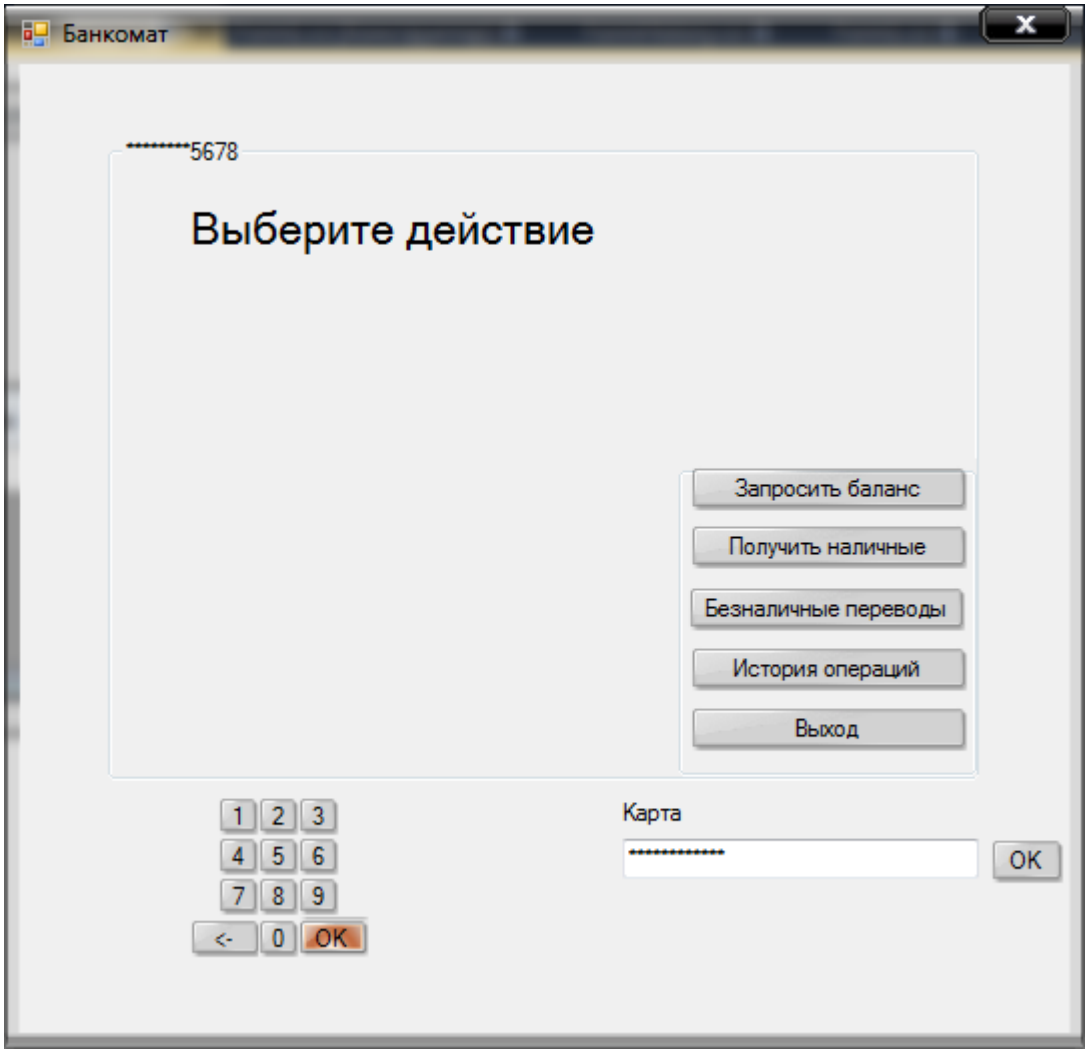

Рисунок 20 – «Выберите действие»

При нажатии кнопки «Запросить баланс» банкомат обращается к серверу, где происходит проверка баланса по номеру карты. Полученная информация отображается на дисплее (рисунок 21).

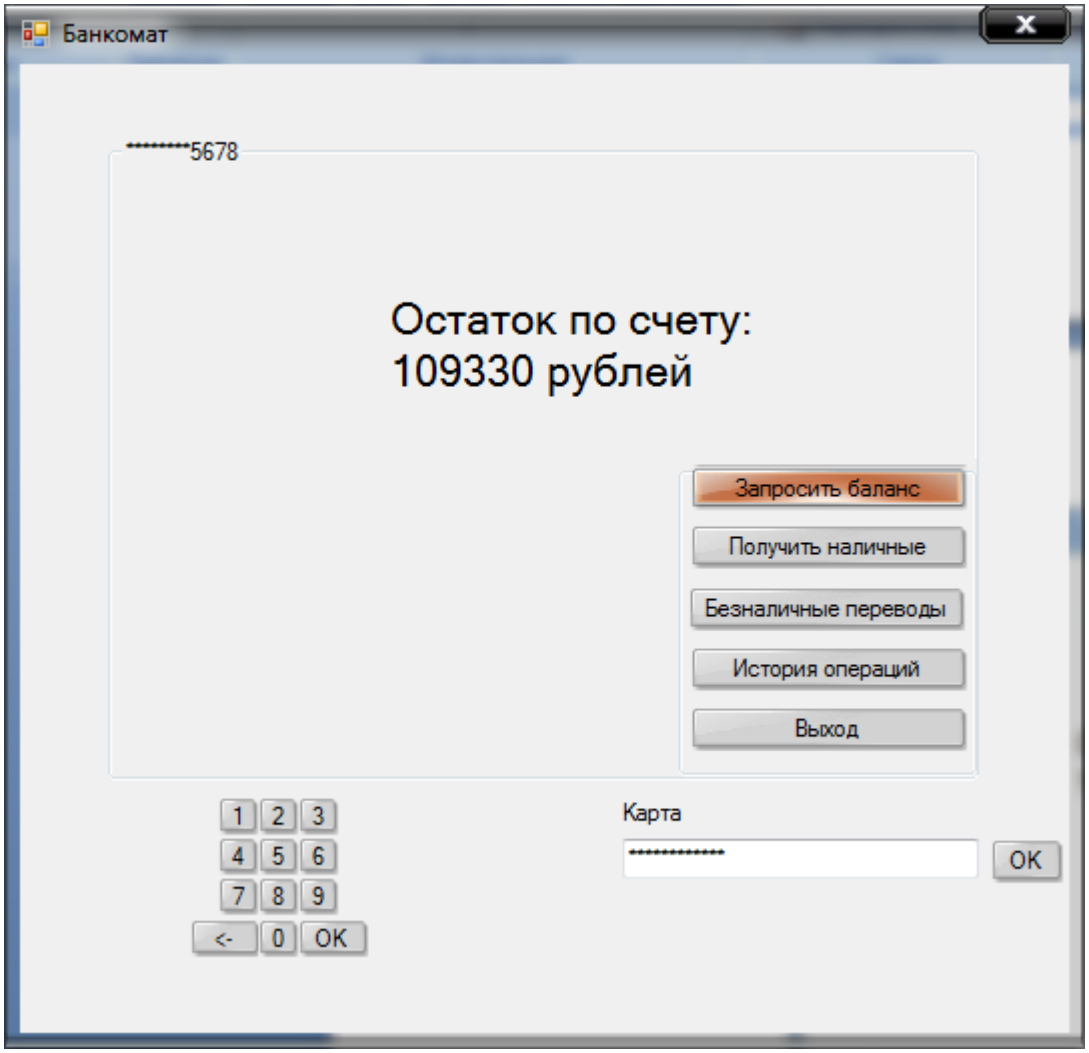

Рисунок 21 – Проверка баланса

При нажатии кнопки «Получить наличные» на дисплее отображаются кнопки выбора суммы. Затем банкомат обращается к серверу, где происходит проверка на наличие требуемой суммы на балансе данной карты и вычитание этой суммы из баланса. При выдаче наличных отображается сообщение, какими купюрами и в каком количестве была выдана данная сумма. На дисплее отображается предложение распечатать квитанцию (рисунок 22).

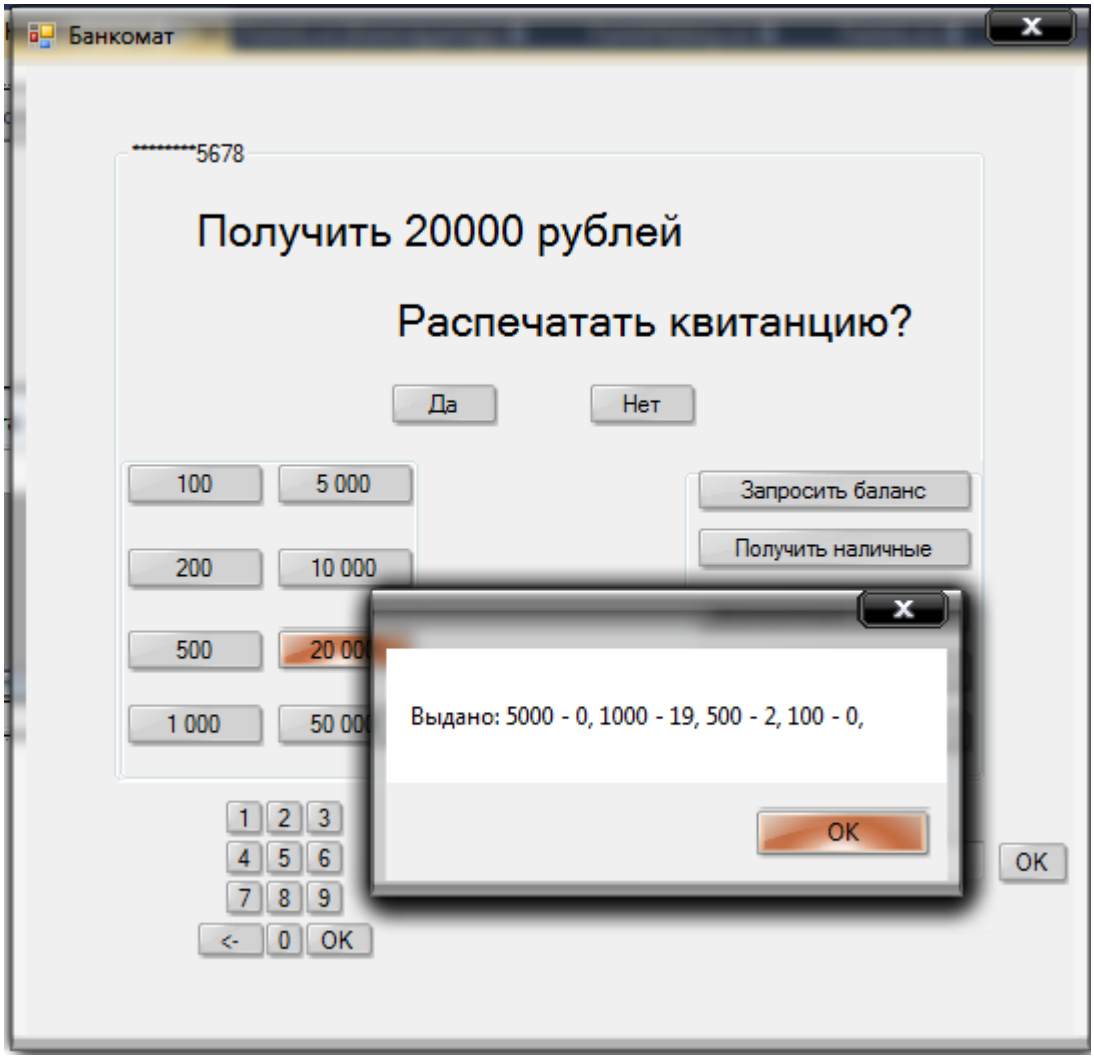

Рисунок 22 – Выдача наличных

При печати квитанции отображается сообщение, содержащее номер счета, выданную сумму и остаток на счете (рисунок 23).

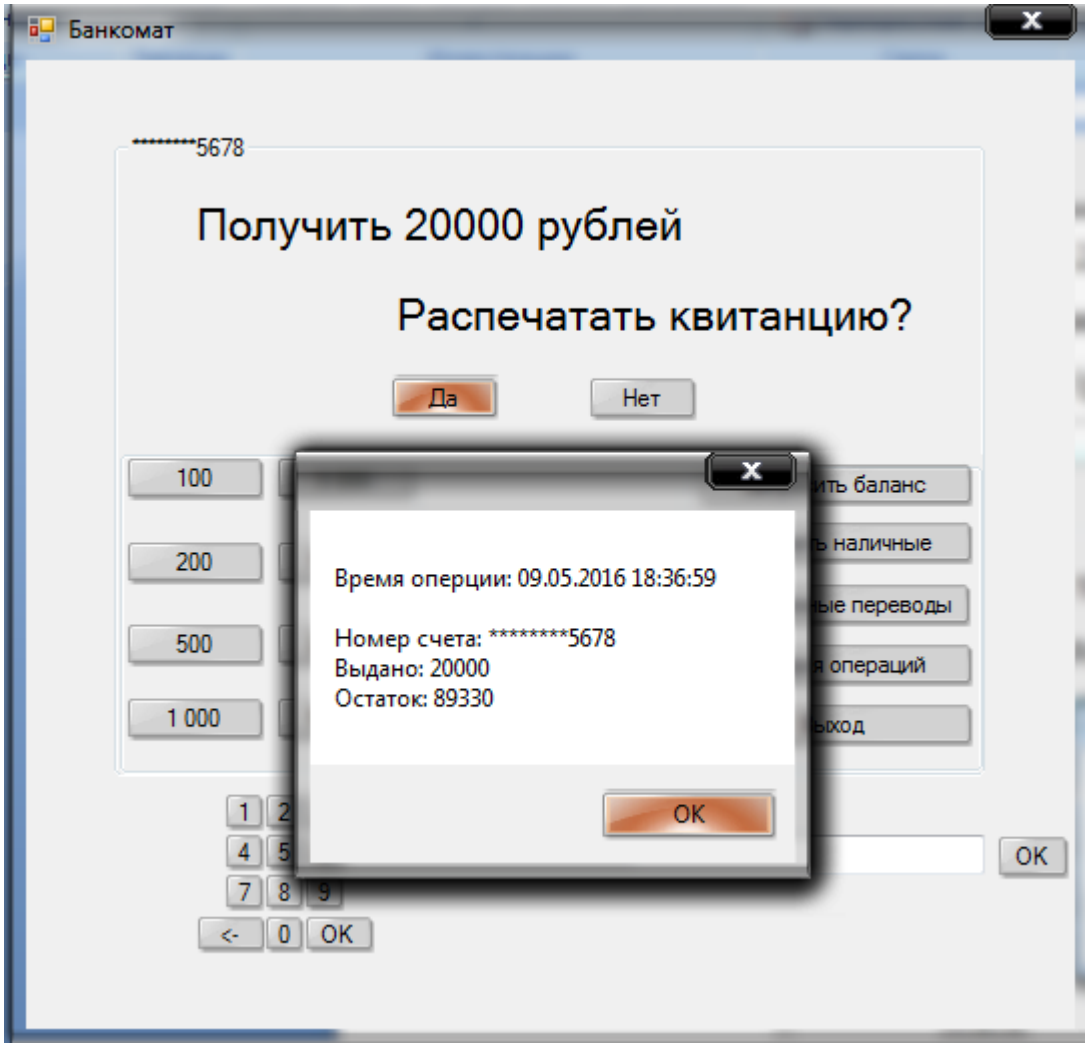

Рисунок 23 – Печать квитанции

При нажатии кнопки «Безналичные переводы» на дисплее банкомата отображается панель с кнопками выбора вида перевода (рисунок 24).

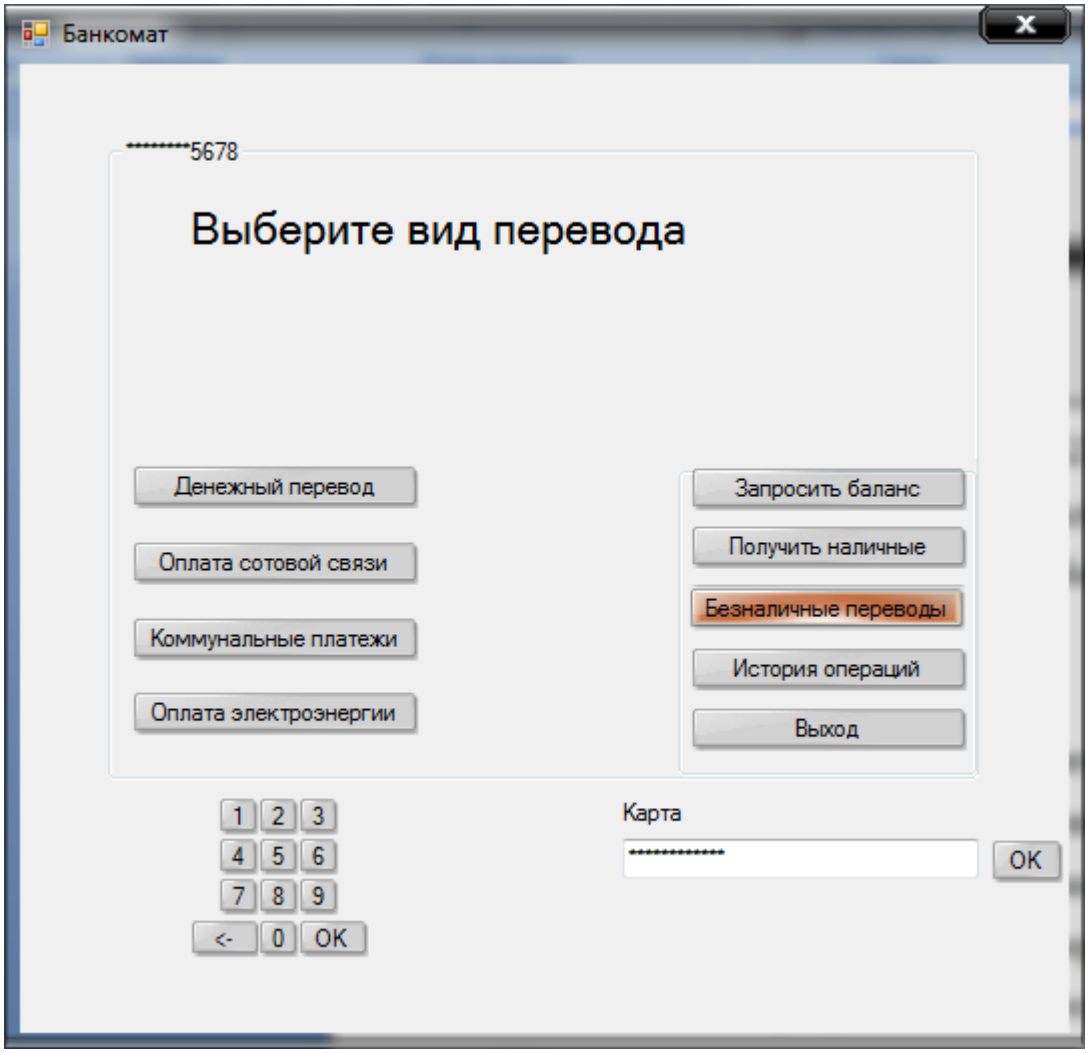

Рисунок 24 – «Выберите вид перевода»

При нажатии кнопки «Денежный перевод» на дисплее банкомата отображается панель с элементами ввода номера карты и суммы для перевода. После ввода данных и нажатия на кнопку «Выполнить перевод» банкомат отправляет запрос серверу на проверку номера карты. Если указанный номер счета в базе на сервере не найдет, отображается сообщение об ошибке (рисунок 25).

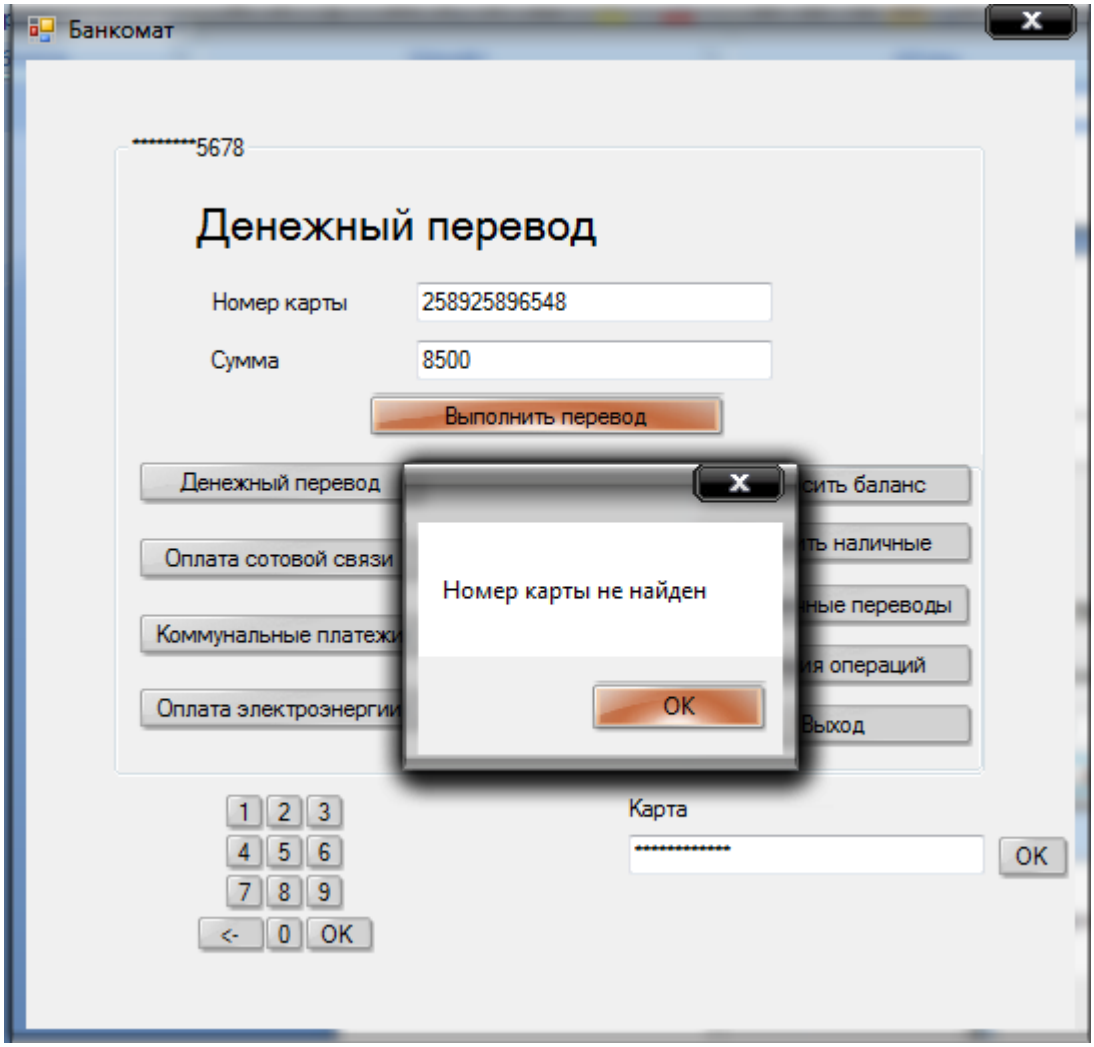

Рисунок 25 – «Номер карты не найден»

После проверки номера карты осуществляется проверка, достаточно ли средств на текущем счете для выполнения запрашиваемой операции. Если средств недостаточно, отображается сообщение об ошибке (рисунок 26).

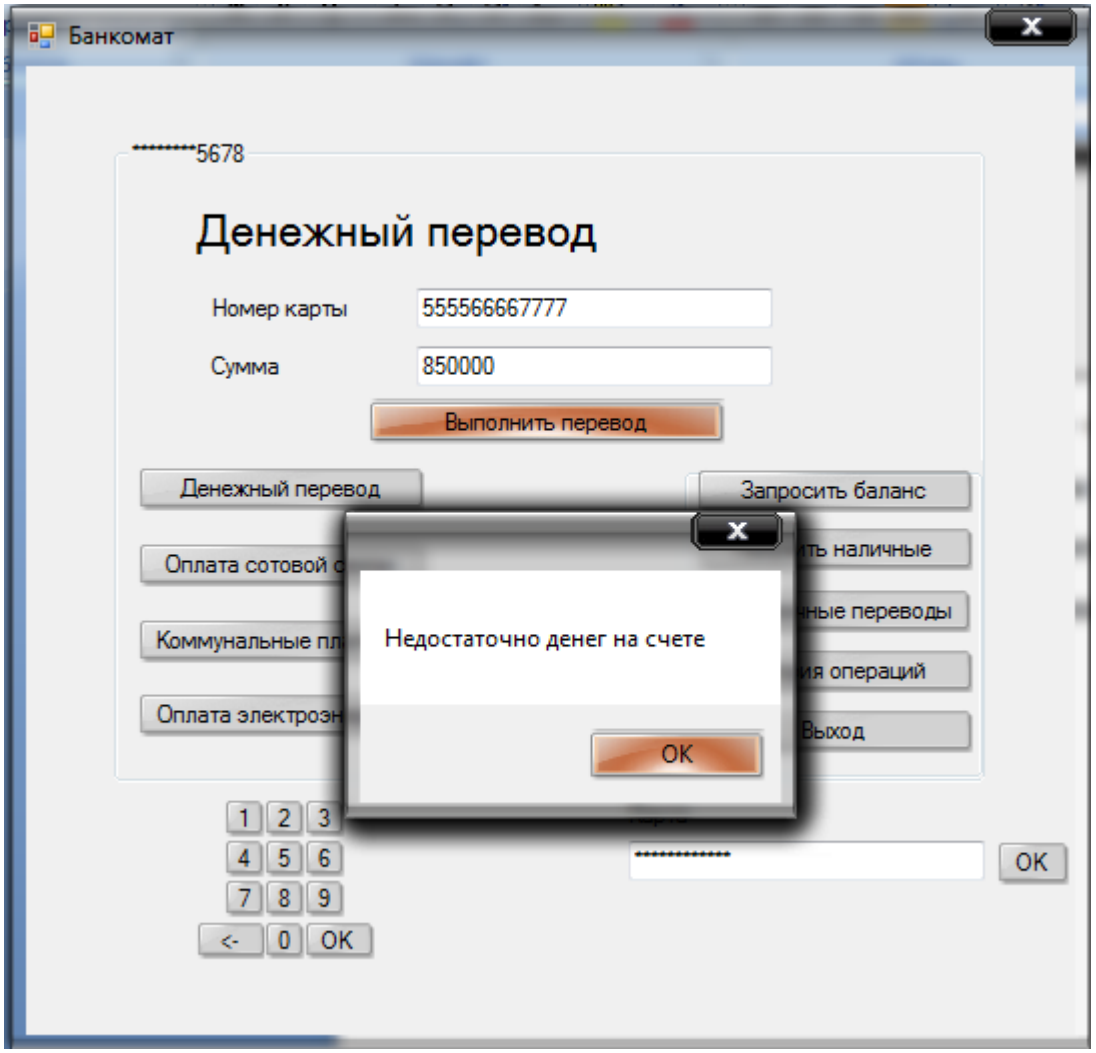

Рисунок 26 – «Недостаточно средств на счете»

Если средств достаточно, выполняется списание указанной суммы с текущего счета и зачисление на счет, на который производится перевод. Об успешном переводе средств пользователь оповещается сообщением (рисунок 27).

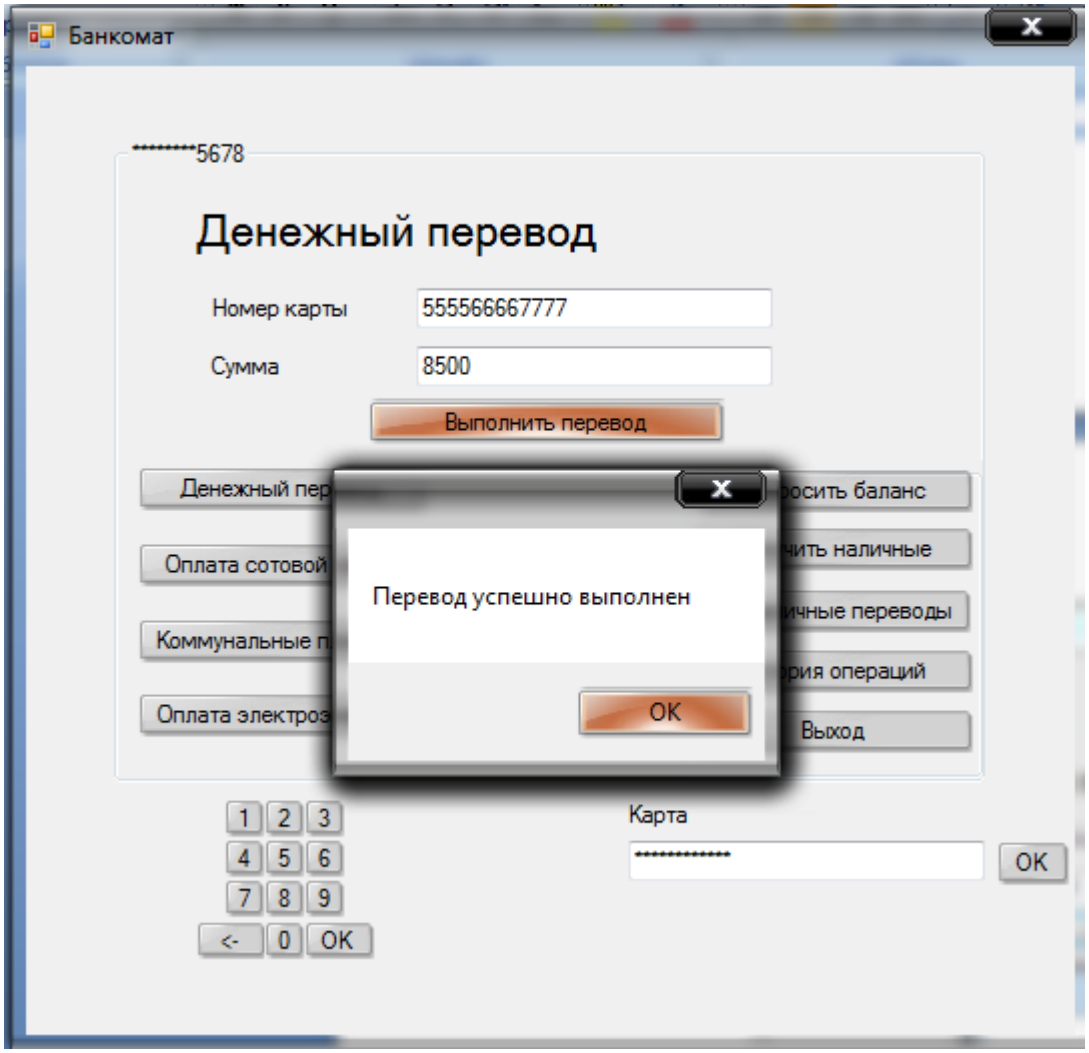

Рисунок 27 – Денежный перевод

При нажатии на кнопку «Оплата сотовой связи» на дисплее банкомата отображается панель с элементами ввода номера телефон и суммы. После ввода требуемых данных и нажатии на кнопку «Выполнить перевод» выполняется проверка, достаточно ли на остатке по счету средств для выполнения операции. Если средств достаточно, производится списание указанной суммы, пользователь оповещается сообщением об успешном выполнении операции (рисунок 28).

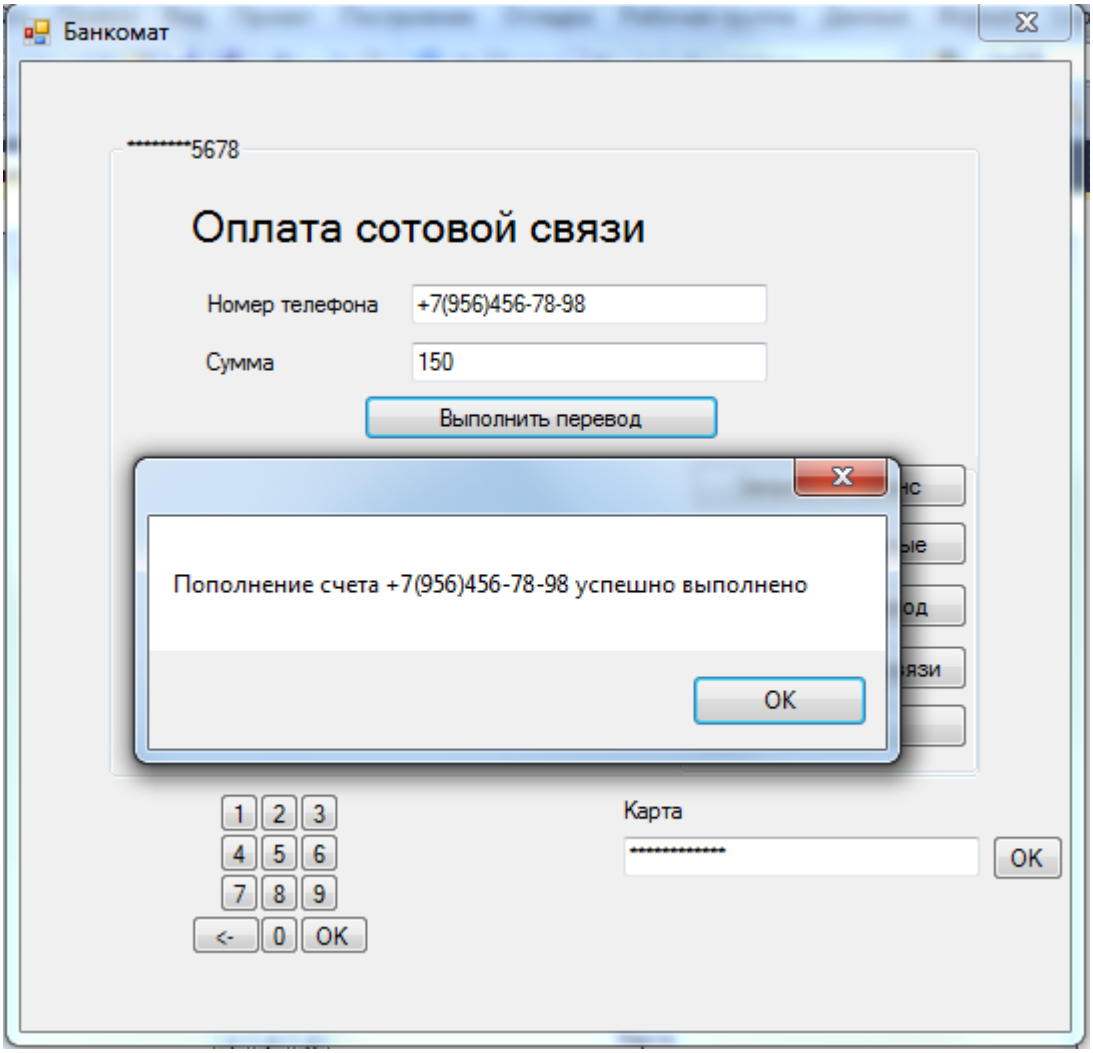

Рисунок 28 – Оплата сотовой связи

При нажатии на кнопки «Коммунальные платежи» или «Оплата электроэнергии» открывается форма «Коммунальные платежи». На ней необходимо выбрать поставщика услуг, отобразится номер счета поставщика для перевода. Далее требуется ввести номер лицевого счета клиента и указать сумму платежа. Будет выполнена проверка лицевого счета и баланса клиента. ОБ у спешном переводе средств пользователь будет оповещен сообщением (рисунок 29).

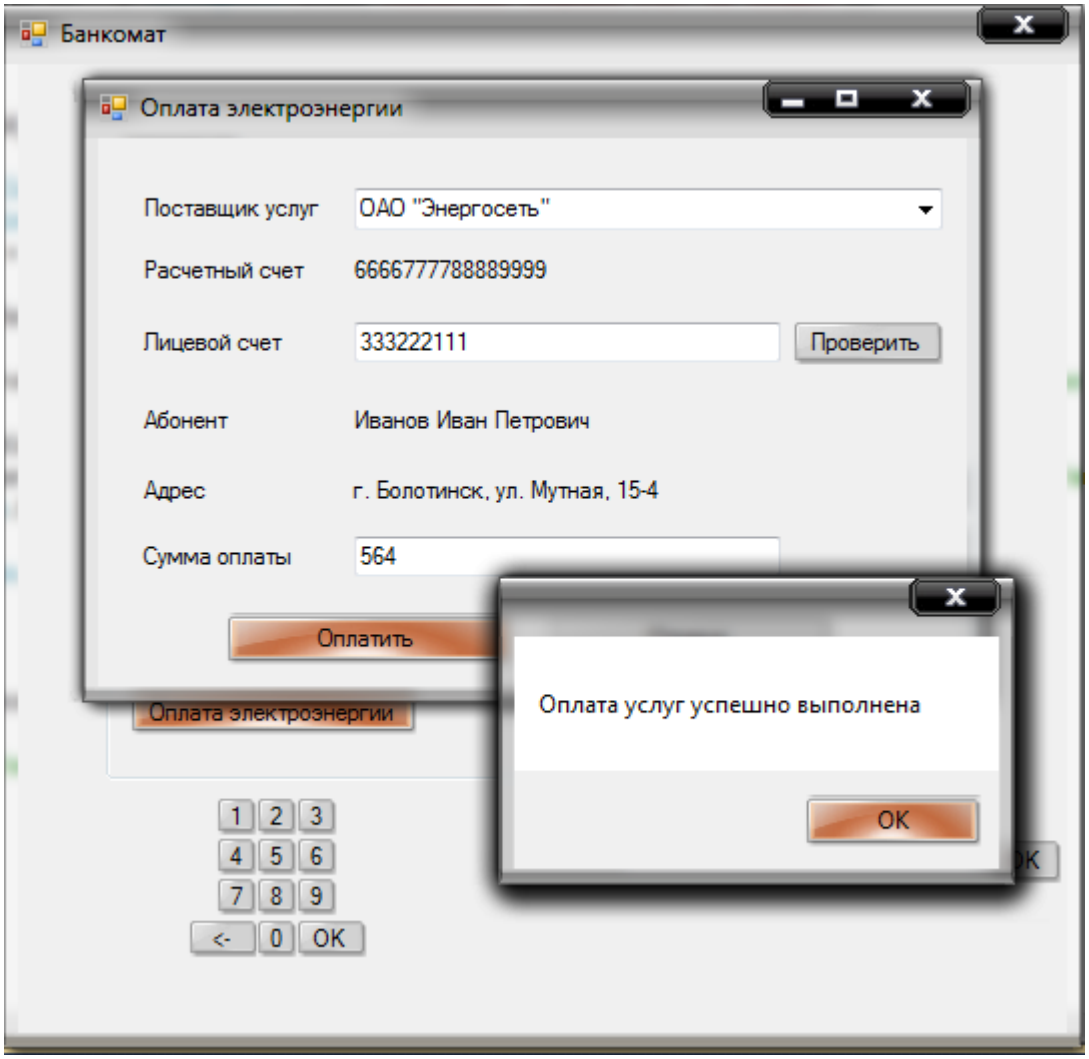

Рисунок 29 – Коммунальные платежи

При нажатии на кнопку «История операций» открывается форма «История операций о карте», на которой необходимо указать период и нажать кнопку «Поиск». На дисплее будет отображена история совершенных операций (рисунок 30).

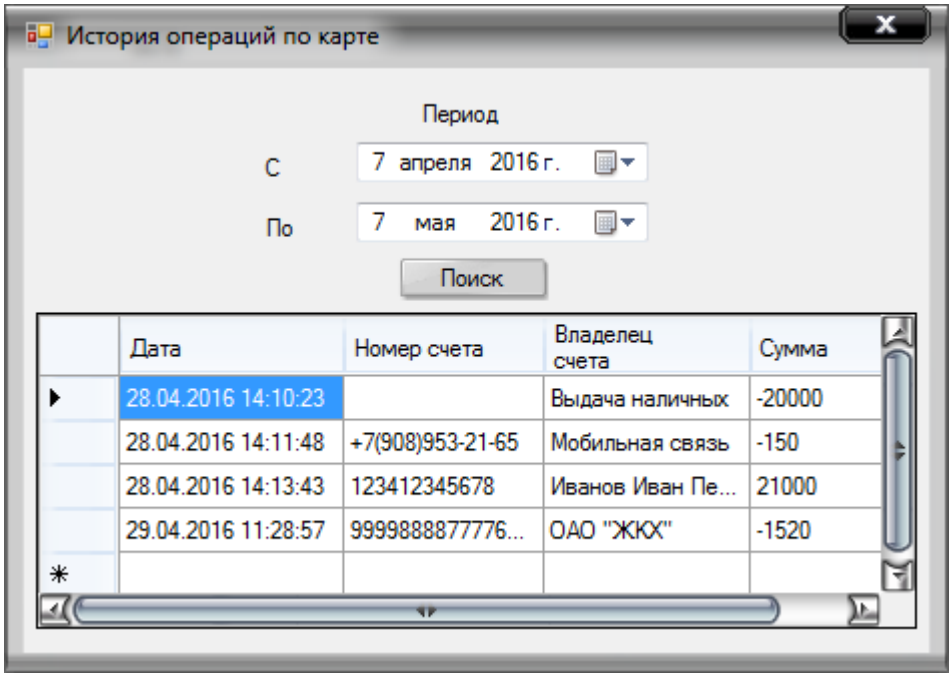

Рисунок 30 – История операций по карте

При нажатии на кнопку «Выход» выполняется очистка поля ввода номера карты, на дисплее отображается надпись «Вставьте карту».

Заключение

В процессе выполнения данной работы было спроектировано и реализовано приложение, имитирующее работу банкомата. Для достижения данной цели были решены следующие задачи:

- были сформулированы требования к системе, которые описаны в виде алгоритмов;

- спроектирована и разработана база данных приложения;

- спроектирована структура приложения, разработана диаграмма классов;

- разработан пользовательский интерфейс;

- разработан программный код;

- приложение было отлажено, все возникшие в ходе реализации ошибки были исправлены.

Приложение было реализовано в среде разработки программного обеспечения MicrosoftVisulStudio 2010, на языке программирования высокого уровня C#.

Список литературы

1. Балдин, К.В. Информационные системы в экономике: учеб.для вузов по спец. 351400 "Прикл. информатика" (по обл.) и др. междисциплинар. спец. / Балдин, К.В., Уткин, В.Б. - М.: Дашков и К, 2008. - 394 с.
2. Брауде, Э.Дж. Технология разработки программного обеспечения / Э.Дж. Брауде. - Питер, 2004. - 656 с.
3. Гагарина, Л.Г. Технология разработки программного обеспечения / Л.Г. Гагарина, Е.В. Кокорева, Б.Д. Виснадул. - М.: Инфра-М, 2008. - 400 с.
4. Нейгел Кристиан, Ивьен Билл, Глинн Джей, Уотсон Карл и Скиннер Морган. Н45 C# 2008 и платформа .NET 3.5 для профессионалов. : Пер. с англ. — М. : ООО "И.Д. Вильяме", 2009. - 1392 с.: ил. - Парал. тит. англ.
5. www.mdsn.com

Приложение 1. Листинг кода приложения

Листинг кода формы «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 Банкомат

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

ClassBankServer server; //экземпляр класса для связи с сервером

private void Form1_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: ";

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);

}

}

//проверка баланса карты

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;

pWireTransfer.Visible = false;

}

private void btGetCash_Click(object sender, EventArgs e) //получить наличные

{

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

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

gbSum.Visible = true;

pTransact.Visible = false;
```

```
pWireTransfer.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;

pWireTransfer.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;

pWireTransfer.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;

//отобразить панель для ввода номера карты и суммы для перевода
pTransact.Visible = true;
mtbMobile.Visible = false;
tbCardTansact.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
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;
//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;
pWireTransfer.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;
pWireTransfer.Visible = false;
FormHistory fh = new FormHistory();
fh.numAccount = numAccount;
fh.Show();
}
}
}
```

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 Банкомат

{

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();

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";

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;
}
public bool CheckCard(string numCard) //проверка карты
{
bool access = false;
dr = SqlSelectCard(numCard);
while (dr.Read()) access = true;

return access;
}
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;
}
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;

}

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();

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();

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;

}

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;
}

}
}
```

Листинг кода формы «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 Банкомат

{

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 фому

}

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 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 поставщиков»

```
using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Data.OleDb;
```

```
using System.Windows.Forms;

namespace Банкомат

{

class ClassProviders

{

OleDbDataReader dr;

public OleDbDataReader FillProviders(string type) //список поставщиков услуг по типу: коммунальные услуги или электроэнергия

{

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 Providers.idProvider, Providers.nameProvider " +

"FROM Providers WHERE Providers.typeProvider=@type";

OleDbCommand cmd = new OleDbCommand(sqlFindCard, conn);

OleDbParameter[] prms = new OleDbParameter[1];

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

prms[0].Value = type;

cmd.Parameters.AddRange(prms);

dr = cmd.ExecuteReader();

return dr;

}

public string FindAccountProvider(int idProvider) //получить номер счета поставщика
```

```
{

string account="";

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 Providers.accountProvider " +

"FROM Providers WHERE Providers.idProvider=@idProvider";

OleDbCommand cmd = new OleDbCommand(sqlFindCard, conn);

OleDbParameter[] prms = new OleDbParameter[1];

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

prms[0].Value = idProvider;

cmd.Parameters.AddRange(prms);

dr = cmd.ExecuteReader();

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

return account;

}

public OleDbDataReader CheckPersonalAccount(string personalAccount) //проверить
персональнай счет клиента

{

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 ClientsOfProviders.nameClient,
ClientsOfProviders.addressClient " +

"FROM ClientsOfProviders WHERE
ClientsOfProviders.personalAccount=@personalAccount";

OleDbCommand cmd = new OleDbCommand(sqlFindCard, conn);

OleDbParameter[] prms = new OleDbParameter[1];

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

prms[0].Value = personalAccount;

cmd.Parameters.AddRange(prms);

dr = cmd.ExecuteReader();

return dr;

}

}

}
```

Листинг кода формы «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 Банкомат

{

public partial class FormHistory : Form

{

public FormHistory()

{

InitializeComponent();

}

public string numAccount; //номер счета

private void btOK_Click(object sender, EventArgs e)

{

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

DateTime dateStart = dtDateStart.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

}

}

}

}

}
```