СОДЕРЖАНИЕ

[Введение 5](#_Toc504649449)

[1. Aналитическая часть 6](#_Toc504649450)

[1.1. Обоснование актуальности задачи 6](#_Toc504649451)

[1.2. Обоснование выбора и описание методологии разработки программы решения задачи 8](#_Toc504649452)

[1.3. Обоснование выбора и описание инструментальных средств разработки программы решения задачи 9](#_Toc504649453)

[1.4. Содержательная постановка задачи 25](#_Toc504649454)

[2. Проектная часть 31](#_Toc504649455)

[2.1. Разработка структуры данных программы 31](#_Toc504649456)

[2.2. Логическая модель программы 38](#_Toc504649457)

[2.3. Разработка пользовательского интерфейса программы 41](#_Toc504649458)

[2.4. Листинг программы 55](#_Toc504649459)

[2.5. Разработка тестовых данных 55](#_Toc504649460)

[2.6. Результаты работы программы 62](#_Toc504649461)

[2.7. Руководство по использованию программы 69](#_Toc504649462)

[Заключение 85](#_Toc504649463)

[Приложение 1 87](#_Toc504649464)

[Приложение 2 88](#_Toc504649465)

[Приложение 3 89](#_Toc504649466)

[Приложение 4 90](#_Toc504649467)

[Приложение 5 92](#_Toc504649468)

[Приложение 6 93](#_Toc504649469)

[Приложение 7 94](#_Toc504649470)

[Приложение 8 95](#_Toc504649471)

[Приложение 9 96](#_Toc504649472)

[Приложение 10 98](#_Toc504649473)

[Приложение 11 99](#_Toc504649474)

[Приложение 12 100](#_Toc504649475)

ВВЕДЕНИЕ

Актуальность темы исследования определяется следующим. Одна из наиболее важных задач управления любым банком – обеспечение соответствующего уровня (т.е. способность банка платить по своим обязательствам), так как ее недостаточный уровень часто является первым признаком наличия у банка серьезных финансовых затруднений, в такой ситуации он обычно начинает терять свои депозиты, что уменьшает его наличные средства и заставляться от наиболее ликвидных активов. Кроме того, к важнейшим задачам деятельности банка относятся обеспечение достаточного уровни прибыльности и снижения уровня рисков. Названные факторы заставляют российские банки менять стратегию своего поведения: формировать портфель активов с учетом их риска, ориентироваться не столько на высокодоходные операции, сколько на приносящие постоянный надежный доход и т.п. В этих условиях управление коммерческим банком превращается во все более сложный технологический процесс, требующий учета совокупного влияния многочисленных факторов доходности и рисков, а также различных внешних ограничений и при этом направленный на получение гарантированной прибыли, оправдывающей вложение средств акционеров в банковский капитал.

Объектом разработки является автоматизация деятельности банка.

Предметом разработки является автоматизация банковских операций и отчетность по счетам клиента.

Целью данной разработки является проектирование и реализация программного обеспечения для автоматизации учета банковских пассивов.

# AНАЛИТИЧЕСКАЯ ЧАСТЬ

## Обоснование актуальности задачи

* + 1. Актуальность темы

Управление пассивами банка — составная часть банковского менеджмента, формирующая структуру пассивов банка. Основными задачами управления пассивами является обеспечение банка необходимыми кредитными ресурсами, слежение за обязательной прибыльностью средств банка. Важным средством управления пассивами является политика банка в части депозитного процента.

Исходя из этого было разработано программное обеспечение «Банк», которое учитывает необходимые условия для автоматизации работы банка.

* + 1. Анализ существующих разработок по теме

Банковские информационные системы (БИС) – особый класс программ, предназначенный для комплексной автоматизации банковских операций. Существует минимальный набор критериев, который бы позволял отнести ту или иную систему к уровню БИС:

* расчёт и распределение доходов и налогов между бюджетами;
* учёт расчётно–платёжных документов до наступления срока платежа и не оплаченных в срок;
* обеспечение правильных и своевременных расчётов между клиентами;
* начисление и списание процентов по текущим и расчётным счетам;
* учет ценностей и документов на вне балансовых счетов;
* учет срочных обязательств по ссудам;
* депонирование средств для выдачи чековых книжек, аккредитивов и акцептов платёжных поручений;
* организация и контроль операций межфилиального оборота;
* составление баланса.
  + - 1. Интегрированная банковская система «БИСКВИТ»

Интегрированная банковская система (ИБС) БИСКВИТ обеспечивает высокий уровень информационной поддержки банковских операций в условиях динамичного развития рынка финансовых услуг.

Функциональная полнота позволяет автоматизировать множество операций коммерческого банка.

Модульная структура дает возможность банку нести только обоснованные в данный момент сферой его деятельности и объемом документооборота затраты на программное обеспечение, легко наращивая функциональные возможности системы по мере расширения круга выполняемых операций.

Единая технология позволяет автоматически отражать операцию, выполняемую в любом модуле, в единой БД и использовать ее результаты во всех остальных модулях.

* + - 1. QBIS.Reporting

QBIS.Reporting – автоматизированная система отчетности, работающая в соответствии с российскими и международными стандартами. На протяжении многих лет считается лидером по количеству отчетных форм среди представленных на российском рынке аналогичных систем. Располагает широким функционалом, который включает в себя следующие опции:

* полный комплект регуляторной отчетности ЦБ РФ
* налоговый учет и отчетность
* отчетность по МСФО
* управленческая отчетность
* конструктор отчетов

Возможности и отличительные характеристики:

* Консолидация данных из различных информационных систем
* Хранение первичных данных и расчетных показателей, сквозная навигация по данным из любого отчета
* Технологии OLAP и Datawarehouse
* Реальная поддержка оперативного изменения методик
* Сопровождение изменений законодательства в банковской сфере и инструкций ЦБ РФ
  + - 1. Bank master

Bank master – это универсальная банковская система, однако существенная доля функциональных подсистем поддерживается за счет дополнительных продуктов производителя или третьих фирм. Допускается как автономная работа отделений (филиалов), так и совместная работа в режиме клиент–сервер. Функциональное развитие системы осуществлялось в форме проектов для отдельных банков.

Все приведенные выше программы обладают чрезмерным набором критериев, который бы позволил автоматизировать работу банка. Поэтому в данной работе разрабатывается новое программное обеспечение с учетом всех требований.

## Обоснование выбора и описание методологии разработки программы решения задачи

Важнейшим шагом на пути к совершенствованию языков программирования стало появление объектно-ориентированного подхода к программированию (или, сокращенно, ООП) и соответствующего класса языков. Наиболее известным примером объектно-ориентированного языка программирования является язык C++, развившейся из императивного языка С. Его прямым потомком и логическим продолжением является язык С# на основе которого и будет писаться клиентское приложение данной курсовой работы. При объектно-ориентированном подходе программа представляет собой описание объектов, их свойств (или атрибутов), совокупностей (или классов), отношений между ними, способов их взаимодействия и операций над объектами (или методов). Несомненным преимуществом данного подхода является концептуальная близость к предметной области произвольной структуры и назначения.

Механизм наследования атрибутов и методов позволяет строить производные понятия на основе базовых и таким образом создать модель сколь угодно сложной предметной области с заданными свойствами. В связи с тем, что объектно-ориентированный подход используется в основном для создания больших проектов и, когда важна управляемость проекта, а также скорость разработки и его модифицируемость, поэтому в данной работе он не используется в качестве основного.

Помимо этого, также будет использоваться и структурный. В его основе лежит представление программы в виде иерархической структуры блоков. В соответствии с данной методологией любая программа строится без использования оператора goto из трёх базовых управляющих структур: последовательность, ветвление, цикл; кроме того, используются подпрограммы. При этом разработка программы ведётся пошагово.

## Обоснование выбора и описание инструментальных средств разработки программы решения задачи

Для разработки данного проекта требуется программа для реализации клиентской части, которая будет предусматривать современный и удобный интерфейс для дальнейшего пользователя, в проекте для этой цели использована среда разработки MS Visual Studio 2015. Для реализации же серверной части, которая позволит хранить данные и управлять ими, выбрана база данных SQLite.

* + 1. Обоснование выбора средств разработки клиентской части

В качестве средства разработки клиентской части ПО автоматизации расчета заработной платы предприятия использована среда программирования Microsoft Visual Studio 2015. Среда разработана в соответствии с концепцией визуального программирования.

Microsoft Visual Studio 2015 — это набор инструментов для создания программного обеспечения: от планирования до разработки пользовательского интерфейса, написания кода, тестирования, отладки, анализа качества кода и производительности, развертывания в средах клиентов и сбора данных телеметрии по использованию. Эти инструменты предназначены для максимально эффективной совместной работы; все они доступны в интегрированной среде разработки (IDE) Visual Studio. Еще одним плюсом данной среды является, то, что она не нуждается в каких-либо настройках. По умолчанию Visual Studio обеспечивает поддержку C#, C и C++, JavaScript, F# и Visual Basic. В качестве языка программирования клиентской части данного проекта выбран язык C#.

Visual Studio можно использовать для создания различных типов приложений, от простых приложений для магазина и игр для мобильных телефонов до больших и сложных систем, обслуживающих предприятия и центры обработки данных. Visual Studio позволяет создавать:

* приложения и игры, которые выполняются не только на платформе Windows, но и на Android и iOS;
* веб-сайты и веб-службы на основе ASP.NET, JQuery, AngularJS и других популярных платформ;
* приложения для самых разных платформ и устройств, включая, но не ограничиваясь: Office, Sharepoint, Hololens, Kinect.

Для работы с Microsoft Visual Studio 2015 могут быть использованы следующие операционные системы: Windows7/8/8.1/10.

* + 1. Выбор языка программирования

В качестве языка программирования выбран язык C#.

C# является языком программирования, который разработан для создания множества приложений, работающих в среде .NET Framework. Язык C# прост, типобезопасен и объектно-ориентирован. Благодаря множеству нововведений C# обеспечивает возможность быстрой разработки приложений, но при этом сохраняет выразительность и элегантность, присущую С-подобным языков.

Visual C# — это реализация языка C# корпорацией Майкрософт. Поддержка Visual C# в Visual Studio обеспечивается с помощью полнофункционального редактора кода, компилятора, шаблонов проектов, конструкторов, мастеров кода, мощного и удобного отладчика и многих других средств. Библиотека классов .NET Framework предоставляет доступ ко многим службам операционной системы и к другим полезным, хорошо спроектированным классам, что существенно ускоряет цикл разработки.

C# упрощает разработку компонентов программного обеспечения благодаря нескольким инновационным конструкциям языка, в число которых входят следующие:

* инкапсулированные сигнатуры методов, называемые делегатами, которые поддерживают типобезопасные уведомления о событиях;
* свойства, выступающие в роли методов доступа для закрытых переменных-членов;
* атрибуты с декларативными метаданными о типах во время выполнения;
* встроенные комментарии XML-документации;
* LINQ, предлагающий встроенные возможности запросов в различных источниках данных.
  + 1. Общие сведения о языке программирования

Использованный язык C# является объектно-ориентированным языком программирования**.**

C# относится к семье языков с C-подобным синтаксисом, из них его синтаксис наиболее близок к C++ и Java. Язык имеет статическую типизацию, поддерживает полиморфизм, перегрузку операторов (в том числе операторов явного и неявного приведения типа), делегаты, атрибуты, события, свойства, обобщённые типы и методы, итераторы, анонимные функции с поддержкой замыканий, LINQ, исключения, комментарии в формате XML.

Переняв многое от своих предшественников — языков C++, Pascal, Модула, Smalltalk и, в особенности, Java — С#, опираясь на практику их использования, исключает некоторые модели, зарекомендовавшие себя как проблематичные при разработке программных систем, например, C# в отличие от C++ не поддерживает множественное наследование классов (между тем допускается множественное наследование интерфейсов).

* + - 1. Элементы языка

1) Переменные

Синтаксис объявления переменных в C# представлен на рисунке 1.

ТипДанных Идентификатор;

Рис. 1 – Объявление переменной

Пример объявления переменной представлен на рисунке 2.

int i;

Рис. 2 – Пример объявления переменной

Объявить можно переменную любого действительного типа. Важно подчеркнуть, что возможности переменной определяются ее типом. Например, переменную типа bool нельзя использовать для хранения числовых значений с плавающей точкой. Кроме того, тип переменной нельзя изменять в течение срока ее существования. В частности, переменную типа int нельзя преобразовать в переменную типа char.

Все переменные в C# должны быть объявлены до их применения. Это нужно для того, чтобы уведомить, компилятор о типе данных, хранящихся в переменной, прежде чем он попытается правильно скомпилировать любой оператор, в котором используется переменная. Это позволяет также осуществлять строгий контроль типов в C#.

Начиная с версии C# 3.0, компилятору предоставляется возможность самому определить тип локальной переменной. Такая переменная называется неявно типизированной и объявляется с помощью ключевого слова var. Пример объявления переменной представлен на рисунке 3.

var i = 12;

Рис. 3 – Объявление переменной

2) Операция присваивания

Задать значение переменной можно, в частности, с помощью оператора присваивания. Кроме того, задать начальное значение переменной можно при ее объявлении. Для этого после имени переменной указывается знак равенства и присваиваемое значение. Пример присваивания переменной представлен на рисунке 4.

int i = 42;

Рис. 4 – Присваивание переменной

3) Комментарии

Комментарий - строка, которая не выполняется программой. Служит для записи пояснений к коду, а также исключения (комментирования) строк кода, которые не надо выполнять, но и удалять нельзя (например, чтобы были видны проведенные изменения). В C# используются традиционные комментарии в стиле С — однострочные и многострочные. Запись комментариев изображена на рисунке 5.

//Однострочный комментарий

/\*Многострочный

Комментарий\*/

Рис. 5 - Комментарии

4) Конструкция перехода по условию

Если условие выполняется, то выполняется блок кода 1, а блок кода 2 игнорируется. Если условие не выполняется, то выполняется блок кода 2, а блок кода 1 игнорируется:

Условные конструкции - один из базовых компонентов многих языков программирования, которые направляют работу программы по одному из путей в зависимости от определенных условий.

В языке C# используются следующие условные конструкции: if..else и switch..case.

Конструкция if/else проверяет истинность некоторого условия и в зависимости от результатов проверки выполняет определенный код. Пример конструкции if/else представлен на рисунках 6-7.

int num1 = 8;int num2 = 6;

if(num1 > num2)

{

    Console.WriteLine("Число {0} большечисла {1}", num1, num2);

Рис. 6 – Конструкция if/else

|  |
| --- |
| }  else  {      Console.WriteLine("Число {0} меньшечисла {1}", num1, num2);  } |

Рис. 7 – Конструкция if/else

После ключевого слова if ставится условие. И если это условие выполняется, то срабатывает код, который помещен в блоке if после фигурных скобок. Если условие не выполняется, срабатывает кусок кода после слова else.

Конструкция switch/case аналогична конструкции if/else, так как позволяет обработать сразу несколько условий. Пример конструкции switch/case представлен на рисунке 8.

Console.WriteLine("Нажмите Y или N");

string selection = Console.ReadLine();

switch (selection)

{

    case "Y":

        Console.WriteLine("Вынажалибукву Y");

        break;

    case "N":

        Console.WriteLine("Вынажалибукву N");

        break;

    default:

        Console.WriteLine("Вынажалинеизвестнуюбукву");

        break;

}

Рис. 8 - Конструкция switch/case

После ключевого слова switch в скобках идет сравниваемое выражение. Значение этого выражения последовательно сравнивается со значениями, помещенными после оператора case. И если совпадение будет найдено, то будет выполняться определенный блок case.

В конце каждого блока case должен ставиться один из операторов перехода: break,goto case,return или throw. Как правило, используется оператор break. При его применении другие блоки case выполняться не будут.

Однако если нужно, чтобы, после выполнения текущего блока case выполнялся другой блок case, то можно использовать вместо break оператор goto case.

Если мы необходимо также обработать ситуацию, когда совпадения не будет найдено, то можно добавить блок default, как в примере выше.

Применение оператора return позволит выйти не только из блока case, но и из вызывающего метода. То есть, если в методе Main после конструкции switch..case, в которой используется оператор return, идут какие-либо операторы и выражения, то они выполняться не будут, а метод Main завершит работу.

Оператор throw применяется для выброса ошибок.

5) Циклы

Циклы также являются управляющими конструкциями, позволяя в зависимости от определенных условий выполнять некоторое действие множество раз. В C# имеются следующие виды циклов:

* for;
* foreach;
* while;
* do...while.

Цикл for представлен на рисунке 9.

for ([инициализация счетчика]; [условие]; [изменение счетчика])

{

    // действия

}

Рис. 9 – Цикл for

Цикл foreach:

Цикл foreach предназначен для перебора элементов в контейнерах. Конструкция цикла foreach представлена на рисунке 10.

foreach (тип\_данныхназвание\_переменнойin контейнер)

{

    // действия

}

Рис. 10 – Цикл foreach

Цикл do:

В цикле do сначала выполняется код цикла, а потом происходит проверка условия в инструкции while. И пока это условие истинно, цикл повторяется. Конструкция цикла do представлена на рисунке 11.

do

{// действия }

while ([условие]);

Рис. 11 – Цикл do

Цикл while:

В отличие от цикла do цикл while сразу проверяет истинность некоторого условия, и если условие истинно, то код цикла выполняется. Конструкция цикла while представлена на рисунке 12.

while ([условие])

{

    // действия

}

Рис. 12 – Цикл while

Иногда возникает ситуация, когда требуется выйти из цикла, не дожидаясь его завершения. В этом случае используется оператор break. Если нужно, чтобы при проверке цикл не завершался, а просто переходил к следующему элементу, используется оператор continue.

6) Операторы передачи управления

В C# есть пять операторов, изменяющих естественный порядок выполнения вычислений:

* оператор безусловного перехода goto;
* оператор выхода из цикла break;
* оператор перехода к следующей итерации цикла continue;
* оператор возврата из функции return;
* оператор генерации исключения throw.

Оператор goto-метка передает управление на помеченный оператор. Метка — это обычный идентификатор, областью видимости которого является функция, в теле которой он задан. Метка должна находиться в той же области видимости, что и оператор перехода.

Оператор break используется внутри операторов цикла или выбора для перехода в точку программы, находящуюся непосредственно за оператором, внутри которого находится оператор break.

Оператор перехода к следующей итерации текущего цикла continue пропускает все операторы, оставшиеся до конца тела цикла, и передает управление на начало следующей итерации.

Оператор возврата из функции return завершает выполнение функции и передает управление в точку ее вызова.

* + - 1. Способы структурирования программы

1) Процедура

Процедура или метод - часть программного модуля, предназначенная для выполнения определенной задачи.

Процедура может иметь входящие параметры, а может их не иметь – все зависит от ее назначения.

В C# определение метода состоит из любых модификаторов (таких как спецификация доступности), типа возвращаемого значения, за которым следует имя метода, затем список аргументов в круглых скобках и далее - тело метода в фигурных скобках. Конструкция процедуры представлена на рисунке 13.

[модификаторы] тип\_возвратаИмяМетода([параметры])

{

// Тело метода

}

Рис. 13 – Конструкция процедуры

Каждый параметр состоит из имени типа параметра и имени, по которому к нему можно обратиться в теле метода. Вдобавок, если метод возвращает значение, то для указания точки выхода должен использоваться оператор возврата return вместе с возвращаемым значением.

Если метод не возвращает ничего, то в качестве типа возврата указывается void. Если же он не принимает аргументов, то все равно после имени метода должны присутствовать пустые круглые скобки. При этом включать в тело метода оператор возврата не обязательно — метод возвращает управление автоматически по достижении закрывающей фигурной скобки.

В целом, возврат из метода может произойти при двух условиях. Во-первых, когда встречается фигурная скобка, закрывающая тело метода. И во-вторых, когда выполняется оператор return. Имеются две формы оператора return: одна — для методов типа void (возврат из метода), а другая — для методов, возвращающих конкретные значения (возврат значения).

2) Функция

Тоже что и процедура, но есть одна особенность – функция возвращает некий результат, что оформляется с помощью ключевого слова «Return».

Синтаксис функции представлен на рисунке 14.

[модификаторы] тип\_возвратаИмяФункции([параметры])

{

Return [Возвращаемое значение];// Тело функции

}

Рис. 14 – Конструкция функции

Программа на языке C# может состоять из одного или нескольких файлов. Каждый файл может содержать нуль или несколько пространств имен. Пространство имен может содержать типы, такие как классы, структуры, интерфейсы, перечисления и делегаты, а также другие пространства имен. На рисунке 15 приведена структура программы на C#.

usingSystem;

namespaceYourNamespace

{

classYourClass

{ }

interfaceIYourInterface

{ }

delegate int YourDelegate();

enumYourEnum

{ }

namespaceYourNestedNamespace

{

structYourStruct

{ }

}

classYourMainClass

{

static void Main(string[] args) //Программаначинаетсяздесь

{ } } }

Рис. 15 – Конструкция программы

* + - 1. Средства обмена данными

Средства обмена данными позволяют создавать общий проект на несколько пользователей через подключение к локальной сети. С помощью этих механизмов можно осуществлять разработку раздельную проекта, т.е. есть возможность разработки отдельной части независимо от других частей проекта.

Для возможности совместной разработки используется расширение TeamFoundationServer.

TeamFoundationServer (сокр. TFS) - продукт корпорации Microsoft, представляющий собой комплексное решение, объединяющее в себе систему управления версиями, сбор данных, построение отчётов, отслеживание статусов и изменений по проекту и предназначенное для совместной работы над проектами по разработке программного обеспечения. Данный продукт доступен как в виде отдельного приложения, так и в виде серверной платформы для Visual Studio Team System (VSTS).

Team Foundation Server работает по трёхуровневой архитектуре: клиентский уровень, прикладной уровень и уровень данных. Клиентский уровень используется для создания и управления проектами, а также для доступа к хранимым и управляемым элементам проекта. На этом уровне TFS не содержит никаких пользовательских интерфейсов, но предоставляетвеб-сервисы, которые могут быть использованы клиентскими приложениями для самостоятельной интеграции в функциональность TFS. Эти веб-сервисы используются такими приложениями, как Visual Studio Team System для применения TFS в качестве серверной инфраструктуры хранилища информации или выделенного TFS управления приложениями, наподобие включенного приложения Team Foundation Client. Сами веб-сервисы находятся на прикладном уровне. Прикладной уровень также включает в себя веб-портал и репозиторий (хранилище) документации, поддерживаемые Windows SharePoint Services. Веб-портал, называемый Team Project Portal (портал командного проекта), выступает в роли центра взаимодействия для проектов, управляемых TFS. Репозиторий документов используется как для элементов проекта, так и для отслеживания ревизий (документирование изменений), а также для накопления и обработки данных и генерации отчётов. Уровень данных, основывающийся в первую очередь на установленномSQL Server 2005 Standard Edition, обеспечивает сервисы постоянного хранения данных для репозитория документов. Уровень данных и уровень приложений могут существовать на различных физических или виртуальных серверах при использовании Windows Server 2003 или более специализированных версий. Уровень данных не взаимодействует с клиентским уровнем напрямую, только через прикладной уровень.

* + - 1. Встроенные элементы

Встроенные процедуры и функции представлены в таблице 1.

Таблица 1 – Встроенные процедуры и функции языка C#.

|  |  |
| --- | --- |
| Процедура/Функция | Описание процедуры/функции |
| Res = Math.Round(Cost\*0.5, 2); | Округление |
| Res = Math.Floor(Cost); | Возвращает наибольшее целое число, которое не больше заданной переменной |
| var DbResult = db.ProcedureName(); | Передача данных из процедуры |
| db.TableName.ToList() | Передача данных из таблицы БД |
| db.TableName.Add(data); | Добавляет заданную сущность к контексту, поддерживающему набор, в добавленном состоянии, в результате чего она будет вставлена в базу данных при вызове метода SaveChanges |
| db.Entry<TableName>(data).State = System.Data.Entity.EntityState.Modified; | Изменяет выбранную сущность в контексте, поддерживающему набор, в изменённом состоянии, в результате чего она будет изменена в базе данных при вызове метода SaveChanges |
| db.TableName.Remove(data); | Помечает заданную сущность как удалённую, в результате чего она будет удалена из базы данных при вызове метода SaveChanges |
| await db.SaveChangesAsync(); | Асинхронно сохраняет все изменения основной базы данных, произведённые в контексте |
| base.Dispose(); | Устраняет неуправляемые и, если указано, управляемые ресурсы |

* + - 1. Средства отладки программы

В случае если на этапе написания какого-либо кода программист допускает ошибку, которая может быть проанализирована ещё ДО компиляции и запуска программы – Visual Studio об этом сообщит в окне «Список ошибок». В основном на этапе разработки фиксируются, если можно так выразиться, элементарные ошибки: пропуск запятой, несоответствие видимости классов, полей и т.д., то ошибки, которые могут быть исправлены быстро.

Довольно часто могут возникать ситуации, когда на этапе выполнения программы в какой-либо метод передается неверный параметр, скажем, в качестве пути к локальной базе данных, может случайно передаться пустая строка – в этом случае обязательно возникнет исключительная ситуация. Такие моменты студия также фиксирует, если приложение запускается из среды IDE. Visual Studio выводит окно, в котором дается небольшое пояснение об исключительной ситуации. Здесь можно:

* просмотреть сведения об ошибке в справочных материалах;
* поискать решение проблемы в сети;
* просмотреть сведения об ошибке;
* скопировать сведения исключения в буфер обмена.

При выборе опции «Просмотр сведений» откроется ещё одно окно, в котором сдержится подробная информация об исключительной ситуации.

Точка останова или breakpoint – это преднамеренная остановка выполнения программы, при котором выполняется вывод отладчика.

Чтобы установить breakpoint, необходимо навести курсор на нужную строку, и выбрать в меню «Отладка – Точка останова» или нажать [F9]. При этом строка с точкой останова будет помечена. Как только выполнение программы дойдет до точки останова – выполнение прервется и запустится отладчик. Во время выполнения отладчика можно проверить значения переменных.

* + 1. Обоснование выбора СУБД

SQLite это реляционная база данных, запросы к которой можно осуществлять при помощи языка запросов SQL. База данных не поддерживает все особенности SQL и уступает в функциональности другим развитым СУБД, но вполне подходит для хранения и извлечения информации.

Отличия от аналогичных СУБД:

* Аналогичные СУБД состоят из отдельного сервера, поддерживающего работу базы данных и прослушивающих определённый порт, на предмет обращения клиентов. Движок SQLite и интерфейс к ней реализованы в одной библиотеке, что увеличивает скорость выполнения запросов. Такой сервер часто называют встроенным.
* Другое отличие - это бестиповой базой данных. Точнее, есть только два типа целочисленный "integer" и текстовый "text". Причём "integer" используется преимущественно для первичного ключа таблицы, а для остальных данных пойдёт "text". Длина строки, записываемой в текстовое поле, может быть любой

Особенности SQLite все базы данных хранятся в файлах, по одному файлу на базу. Количество баз данных, а также таблиц в них, ограниченно только свободным местом, имеющимся на сайте. А максимально возможный объём одной базы данных составляет 2 Тб. Так как все данные хранятся в файлах, проблем с переносом базы данных с одного хостинга на другой не существует достаточно лишь скопировать соответствующие файлы

* + 1. Общие сведения о языке запросов

SQL (Structured Query Language - язык структурированных запросов) является языком четвертого уровня, предназначенным для работы с реляционными базами данных.

В начале 1970-х годов в одной из исследовательских лабораторий компании IBM была разработана экспериментальная реляционная СУБД System R (англ.), для которой затем был создан специальный язык SEQUEL, позволявший относительно просто управлять данными в этой СУБД. Аббревиатура SEQUEL расшифровывалась как Structured English QUEry Language - структурированный английский язык запросов. Позже по юридическим соображениям язык SEQUEL был переименован в SQL и официальным произношением стало [, es kju:' el] - эс-кью-эл.

Первыми СУБД, поддерживающими новый язык, стали в 1979 году OracleV2 для машин VAX от компании Relational Software Inc. и System/38 от IBM, основанная на System/R.

Первый официальный стандарт языка SQL был принят ANSI в 1986 году и ISO в 1987 году (так называемый SQL-86) и несколько уточнён в 1989 году. Дальнейшее развитие языка поставщиками СУБД потребовало принятия в 1992 году нового расширенного стандарта (ANSI SQL-92 или просто SQL2). Следующим стандартом стал SQL:1999 (SQL3). В настоящее время действует стандарт, принятый в 2003 году (SQL:2003) с небольшими модификациями, внесёнными позже.

* + - 1. Элементы языка

1. Инструкция SELECT

В языке SQL имеется одна основная инструкция для выборки информации из базы данных - инструкция SELECT. Эта инструкция позволяет извлекать информацию из одной или нескольких таблиц базы данных и даже из нескольких баз данных. Результаты выполнения инструкции SELECT помещаются в еще одну таблицу, называемую результирующим набором (result set). Самая простая форма инструкции SELECT состоит из списка столбцов выборки и предложения FROM (Все прочие предложения являются необязательными). Форма инструкции SELECT представлена на рисунке 16.

|  |
| --- |
| SELECT [ALL | DISTINCT] column\_list  FROM {table1 [tab\_alias1]},... |

Рис. 16 – Конструкция запроса

1. Инструкция CREATE TABLE

Инструкция CREATE TABLE создает новую таблицу базы данных со всеми соответствующими столбцами требуемого типа данных. Инструкция создания новой таблицы представлена на рисунке 17.

|  |
| --- |
| CREATE TABLE table\_name ( col\_name1 type1 [NOT NULL | NULL]  [{, col\_name2 type2 [NOT NULL | NULL]}...]) |

Рис. 17

3) Создание базы данных

Базовый синтаксис sqlite команды создание базы данных представлен на рисунке 18.

|  |
| --- |
| $sqlite3 DatabaseName.sqlite |

Рис. 18 – Синтаксис создание БД

4) Вставка, изменение данных

Инструкция INSERT вставляет строки (части строк) в таблицу. Инструкции представлены на рисунке 19.

|  |
| --- |
| INSERT [INTO] tab\_name [(col\_list)]  DEFAULT VALUES | VALUES ({DEFAULT | NULL | expression} [,...n]) |

Рис. 19 – Синтаксис добавление записи

Инструкция UPDATE используется для модифицирования строк таблицы. Общая форма этой инструкции представлена на рисунке 20.

|  |
| --- |
| UPDATE tab\_name  {SET column\_1 = {expression | DEFAULT | NULL} [,...n]  [FROM tab\_name1 [,...n]]  [WHERE condition] |

Рис. 20 – Синтаксис обновления записи

* + - 1. Способы структурирования запросов

Все запросы на получение практически любых данных из одной или нескольких таблиц выполняются с помощью единственного предложения SELECT.В синтаксических конструкциях для обращения к БД используются следующие обозначения:

* звездочка (\*) для обозначения «все» - употребляется в обычном для программирования смысле, т.е. «все случаи, удовлетворяющие определению»;
* квадратные скобки ([]) - означают, что конструкции, заключенные в эти скобки, являются необязательными (т.е. могут быть опущены);
* фигурные скобки ({}) - означают, что конструкции, заключенные в эти скобки, должны рассматриваться как целые синтаксические единицы, т.е. они позволяют уточнить порядок разбора синтаксических конструкций, заменяя обычные скобки, используемые в синтаксисе SQL;
* многоточие (…) - указывает на то, что непосредственно предшествующая ему синтаксическая единица факультативно может повторяться один или более раз;
* прямая черта (|) - означает наличие выбора из двух или более возможностей.
* точка с запятой (;) - завершающий элемент предложений SQL;
* запятая (,) - используется для разделения элементов списков;
* пробелы ( ) - могут вводиться для повышения наглядности между любыми синтаксическими конструкциями предложений SQL;
* строчные буквы используются для написания конструкций, которые должны заменяться конкретными значениями, выбранными пользователем, причем для определенности отдельные слова этих конструкций связываются между собой символом подчеркивания (\_);
* термины «таблица» и «столбец» заменяют (с целью сокращения текста синтаксических конструкций) термины «имя\_таблицы», «имя\_столбца», …, соответственно;
  + - 1. Встроенные функции

Встроенные функции представлены в таблице 2.

Таблица 2 – Основные встроенные функции языка SQL

|  |  |
| --- | --- |
| Процедура/Функция | Описание процедуры/функции |
| MAX(поле) | Выводит максимальное значение поля |
| MIN(поле) | Выводит минимальное значение поля |
| AVG(поле) | Выводит среднее значение поля |
| SUM(поле) | Суммирует все значения поля |
| COUNT(\*) | Выводит количество записей в таблице |
| COUNT(поле) | Выводит количество всех значений поля |

## Содержательная постановка задачи

* + 1. Организационно-экономическая сущность задачи
       1. Назначение задачи

Задача предназначена для работы сотрудника банка при работе с клиентами.

* + - 1. Цели решения задачи

Целью решения задачи является автоматизация процессов:

* регистрация сотрудника;
* ведение справочников «Клиенты», «Паспорт», «Вклады», «Кредиты», «Операции», «Платежные поручения», «Вид вклада», «Счета»;
* вывод отчетности на печать.
  + - 1. Периодичность решения задачи

Данная задача решается ежедневно, а именно при необходимости сотрудников банка получить выписку по счету клиента или по обороту счета за период.

* + - 1. Информационная связь с другими задачами

Информационная связь между задачами осуществляется через информацию, поступающую в базу данных, хранящейся в таблице.

* + 1. Описательная модель задачи

Описательная модель задачи представлена в виде интеллектуальной карты.

Интеллектуальная карта представлена в приложении 1.

* + 1. Описание входной информации
       1. Описание входных документов

Входные документы представлены в приложении 2.

* + - 1. Описание входных данных

Описание структуры входных данных представлено в таблицах 3 - 11.

Таблица 3 – Входные данные «Паспорт»

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование поля | Идентификатор | Тип данных | Источник данных |
| Серия | Seriy | Числовой | Справочник «Паспорт» |
| Номер | Nomer | Текстовый | Справочник «Паспорт» |
| Кем\_выдан | Kem\_vydan | Текстовый | Справочник «Паспорт» |
| Дата\_выдачи | Date\_vyd | Текстовый | Справочник «Паспорт» |

Таблица 4 – Входные данные «Клиенты»

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование поля | Идентификатор | Тип данных | Источник данных |
| Код\_клиента | Id\_klienta | Числовой | Справочник «Клиент» |
| Серия | Seriy | Числовой | Справочник «Паспорт» |
| Фамилия | lastName | Текстовый | Справочник «Клиент» |
| Имя | Name | Текстовый | Справочник «Клиент» |
| Отчество | Otchestvo | Текстовый | Справочник «Клиент» |
| Дата\_рождения | Date\_rozh | Текстовый | Справочник «Клиент» |

Продолжение таблицы 4

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование поля | Идентификатор | Тип данных | Источник данных |
| Пол | Pol | Текстовый | Справочник «Клиент» |
| Адрес | Adres | Текстовый | Справочник «Клиент» |
| Телефон | Telephone | Текстовый | Справочник «Клиент» |

Таблица 5 – Входные данные «Счета»

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование поля | Идентификатор | Тип данных | Источник данных |
| Номер\_счета | Nomer\_scheta | Числовой | Справочник «Счета» |
| Код\_клиента | Id\_klienta | Числовой | Справочник «Клиент» |
| Дата\_открытия | date\_open | Текстовый | Справочник «Счета» |
| Дата\_закрытия | date\_close | Текстовый | Справочник «Счета» |
| Сумма\_на\_счете | Summa | Текстовый | Справочник «Счета» |

Таблица 6 – Входные данные «Пользователи»

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование поля | Идентификатор | Тип данных | Источник данных |
| Код\_пользователя | Id\_pol | Числовой | Справочник «Пользователи» |
| ФИО | FIO | Текстовый | Справочник «Пользователи» |
| Логин | Login | Текстовый | Справочник «Пользователи» |
| Пароль | Pass | Текстовый | Справочник «Пользователи» |
| Должность | Dolzhnost | Текстовый | Справочник «Пользователи» |

Таблица 7 – Входные данные «Вид\_вклада»

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование поля | Идентификатор | Тип данных | Источник данных |
| Код\_вида\_вклада | id\_vid\_vklada | Числовой | Справочник «Вид\_вклада» |
| Наименование | Name | Числовой | Справочник «Вид\_вклада» |
| Ставка | stavka | Текстовый | Справочник «Вид\_вклада» |
| Срок\_хранения | Srok\_xran | Числовой | Справочник «Вид\_вклада» |

Таблица 8 – Входные данные «Вклады»

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование поля | Идентификатор | Тип данных | Источник данных |
| Код\_вклада | Id\_vklada | Числовой | Справочник «Вклады» |
| Код\_вида\_вклада | id\_vid\_vklada | Числовой | Справочник «Вид\_вклада» |
| Номер\_договора | Name\_dog | Числовой | Справочник «Вклады» |
| Дата | Date | Текстовый | Справочник «Вклады» |
| Сумма | sum | Денежный | Справочник «Вклады» |

Таблица 9 – Входные данные «Кредит»

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование поля | Идентификатор | Тип данных | Источник данных |
| Код\_кредита | Id\_kredita | Числовой | Справочник «Кредит» |
| Номер\_договора | Nomer\_dogovora | Числовой | Справочник «Кредит» |
| Сумма | Summa | Текстовый | Справочник «Кредит» |
| Процентная\_ставка | Procenty | Текстовый | Справочник «Кредит» |
| Дата | Data | Текстовый | Справочник «Кредит» |
| Сумма в месяц | Summainmonth | Денежный | Справочник «Кредит» |
| Осталось погасить | Ostalos | Денежный | Справочник «Кредит» |

Таблица 10 – Входные данные «Платежные поручения»

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование поля | Идентификатор | Тип данных | Источник данных |
| Номер\_поручения | Nomer\_poruch | Числовой | Справочник «Платежные поручения» |
| Номер\_счета\_отправителя | nom\_otpravit | Текстовый | Справочник «Платежные поручения» |
| Номер\_счета\_получателя | Nom\_poluchat | Текстовый | Справочник «Платежные поручения» |
| Дата | Date | Текстовый | Справочник «Платежные поручения» |
| Сумма | Summa | Денежный | Справочник «Платежные поручения» |

Таблица 11 – Входные данные «Операции»

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование поля | Идентификатор | Тип данных | Источник данных |
| Код\_Операции | Id\_oper | Числовой | Справочник «Операции» |
| Код\_кредита | Id\_kredita | Числовой | Справочник «Кредит» |
| Номер\_поручения | Nomer\_poruch | Числовой | Справочник «Платежные поручения» |
| Код\_вклада | Id\_vklada | Числовой | Справочник «Вклады» |
| Код\_пользователя | Id\_pol | Числовой | Справочник «Пользователи» |
| Номер\_счета | Nomer\_scheta | Числовой | Справочник «Счета» |
| Наименование | Name | Текстовый | Справочник «Операции» |
| Дата | Date | Текстовый | Справочник «Операции» |

* + 1. Описание выходной информации
       1. Описание выходных документов

Выходные документы представлены в приложении 3.

* + - 1. Описание выходных данных

Выходные данные представлены в таблицах 12 - 13.

Таблица 12 – Выходные данные «Выписка по счету»

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование поля | Идентификатор | Тип данных | Источник данных |
| Номер\_счета | Nomer\_scheta | Числовой | Справочник «Счета» |
| Наименование | Name | Текстовый | Справочник «Операции» |
| Сумма\_кредита | Sum\_kred | Денежный | Справочник «Кредиты» |
| Сумма\_вклада | Sum\_vklad | Денежный | Справочник «Вклады» |
| Сумма\_на\_счете | Summa | Текстовый | Справочник «Счета» |

Таблица 13 – Выходные данные «Оборот по счету за период»

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование поля | Идентификатор | Тип данных | Источник данных |
| Номер\_счета | Nomer\_scheta | Числовой | Справочник «Счета» |
| Наименование | Name | Текстовый | Справочник «Операции» |
| Сумма\_кредита | Sum\_kred | Денежный | Справочник «Кредиты» |
| Сумма\_вклада | Sum\_vklad | Денежный | Справочник «Вклады» |

* + 1. Логическая модель задачи
       1. Функциональные диаграммы

Контекстная функциональная диаграмма и её детализация представлены в приложении 4.

* + - 1. Диаграммы потоков данных

Контекстная диаграмма потоков данных и ее детализация представлены в приложении 5.

* + 1. Требования к программе
       1. Функциональные требования

Данное программное обеспечение включает в себя следующие функциональные требования:

* хранение информации в базе данных;
* учет вкладов;
* учет кредитов;
* учет платежных поручений;
* формирование отчетов;
* печать документов в Microsoft Word.
  + - 1. Требования к надежности
* предусмотреть контроль вводимой информации;
* предусмотреть блокировку некорректных действий пользователя при работе с программой;
* предусмотреть авторизацию доступа в программе.
  + - 1. Требования к информативности
* наличие понятной справочной системы, всплывающих подсказок, подписей к элементам управления.
  + - 1. Требования к удобству эксплуатации

Программное обеспечение должно иметь дружественный пользовательский интерфейс со свободной навигацией, реализацией диалоговых окон при вводе и выводе информации.

# ПРОЕКТНАЯ ЧАСТЬ

## Разработка структуры данных программы

Структура данных программного обеспечения представлена в виде концептуальной модели, логической модели, физической модели.

Концептуальная модель данных приведена в приложении 6.

### Логическая схема данных

Логическая схема данных расширяет концептуальную путем определения для сущностей их атрибутов, описаний и ограничений, уточняет состав сущностей и взаимосвязей между ними.

Логическая схема данных БД представлена в приложении 7.

### Физическая схема данных

Физическая схема данных описывает реализацию объектов логической модели на уровне объектов конкретной базы данных.

Физическая схема данных БД представлена в приложении 8.

### Описание структуры таблиц БД

Структура таблицы прикладного решения представлена в таблицах 14 - 22.

Таблица 14 – Описание структуры таблицы «Паспорт»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Наименование поля | Идентификатор | Тип данных | Условия и ограничения | Описание |
| Серия | Seriy | Числовой | * целочисленный; * первичный ключ; * число из 4 знаков; * не могут быть отрицательными; * обязательное поле – да. | Используется для идентификации Паспорта |

Продолжение таблицы 14.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Наименование поля | Идентификатор | Тип данных | Условия и ограничения | Описание |
| Номер | Nomer | Текстовый | – обязательное поле – да. | Поле для ввода номера паспорта |
| Кем\_выдан | Kem\_vydan | Текстовый | – обязательное поле – да. | Поле для ввода кем выдан паспорт |
| Дата\_выдачи | Date\_vyd | Текстовый | – обязательное поле – да. | Поле для ввода даты выдачи паспорта |

Таблица 15 – Описание структуры таблицы «Клиенты»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Наименование поля | Идентификатор | Тип данных | Условия и ограничения | Описание |
| Код\_клиента | Id\_klienta | Числовой | * первичный ключ; * не могут быть отрицательными; * обязательное поле – да. | Используется для идентификации клиента |
| Серия | Seriy | Числовой | * целочисленный * внешний ключ; * число из 4 знаков; * не могут быть отрицательными; * обязательное поле – да. | Используется для идентификации паспорта |
| Фамилия | lastName | Текстовый | – обязательное поле – да. | Поле для ввода фамилии |
| Имя | Name | Текстовый | – обязательное поле – да. | Поле для ввода имени |
| Отчество | Otchestvo | Текстовый | – обязательное поле – да. | Поле для ввода отчества |
| Дата\_рождения | Date\_rozh | Текстовый | – обязательное поле – да. | Поле для ввода даты рождения |

Продолжение таблицы 15

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Наименование поля | Идентификатор | Тип данных | Условия и ограничения | Описание |
| Пол | Pol | Текстовый | –обязательное поле – да.  –ограничение («Муж», «Жен»). | Поле для ввода пола |
| Адрес | Adres | Текстовый | * обязательное поле – да. | Поле для ввода адреса |
| Телефон | Telephone | Текстовый | * обязательное поле – да. | Поле для ввода телефона |

Таблица 16 – Описание структуры таблицы «Счета»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Наименование поля | Идентификатор | Тип данных | Условия и ограничения | Описание |
| Номер\_счета | Nomer\_scheta | Числовой | * первичный ключ; * не могут быть отрицательными; * обязательное поле – да. | Используется для идентификации счета |
| Код\_клиента | Id\_klienta | Числовой | * целочисленный * внешний ключ; * длинное число; * не могут быть отрицательными; * обязательное поле – да. | Используется для идентификации клиента |
| Дата\_открытия | date\_open | Текстовый | * обязательное поле – да. | Поле для ввода даты открытия |
| Дата\_закрытия | date\_close | Текстовый | * обязательное поле – да. | Поле для ввода даты закрытия |
| Сумма\_на\_счете | Summa | Текстовый | * обязательное поле – да. | Поле для ввода суммы на счете |

Таблица 17 – Описание структуры таблицы «Пользователи»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Наименование поля | Идентификатор | Тип данных | Условия и ограничения | Описание |
| Код\_пользователя | Id\_pol | Числовой | * первичный ключ; * не могут быть отрицательными; * обязательное поле – да. | Используется для идентификации пользователя |
| ФИО | FIO | Текстовый | * обязательное поле – да | Поле для ввода Фамилии, имени и отчества |
| Логин | Login | Текстовый | – обязательное поле – да. | Поле для ввода логина |
| Пароль | Pass | Текстовый | – обязательное поле – да. | Поле для ввода пароля |
| Должность | Dolzhnost | Текстовый | – обязательное поле – да. | Поле для ввода должности |

Таблица 18 – Описание структуры таблицы «Вид\_вклада»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Наименование поля | Идентификатор | Тип данных | Условия и ограничения | Описание |
| Код\_вида\_вклада | id\_vid\_vklada | Числовой | * первичный ключ; * не могут быть отрицательными; * обязательное поле – да. | Используется для идентификации стационара |
| Наименование | Name | Числовой | * обязательное поле – да | Поле для ввода наименования вклада |
| Ставка | stavka | Текстовый | * обязательное поле – да | Поле для ввода процентной ставки вклада |
| Срок\_хранения | Srok\_xran | Числовой | * обязательное поле – да | Поле для ввода срока хранения вклада |

Таблица 19 – Описание структуры таблицы «Вклады»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Наименование поля | Идентификатор | Тип данных | Условия и ограничения | Описание |
| Код\_вклада | Id\_vklada | Числовой | * целочисленный * внешний ключ; * длинное число; * не могут быть отрицательными; * обязательное поле – да | Используется для идентификации вклада |
| Код\_вида\_вклада | id\_vid\_vklada | Числовой | * целочисленный * внешний ключ; * длинное число; * не могут быть отрицательными; * обязательное поле – да. | Используется для идентификации вида вклада |
| Номер\_договора | Name\_dog | Числовой | * обязательное поле – да. | Поле для ввода номера договора |
| Дата | Date | Текстовый | * обязательное поле – да. | Поле для ввода даты |
| Сумма | sum | Денежный | * обязательное поле – да. | Поле для ввода суммы вклада |

Таблица 20 – Описание структуры таблицы «Кредит»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Наименование поля | Идентификатор | Тип данных | Условия и ограничения | Описание |
| Код\_кредита | Id\_kredita | Числовой | * целочисленный * первичный ключ; * длинное число; * не могут быть отрицательными; * обязательное поле – да | Используется для идентификации кредита |

Продолжение таблицы 20

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Наименование поля | Идентификатор | Тип данных | Условия и ограничения | Описание |
| Номер\_договора | Nomer\_dogovora | Числовой | * обязательное поле – да. | Поле для ввода номера договора |
| Сумма | Summa | Текстовый | * обязательное поле – да. | Поле для ввода суммы |
| Процентная\_ставка | Procenty | Текстовый | * обязательное поле – да. | Поле для ввода процентной ставки |
| Дата | Data | Текстовый | * обязательное поле – да. | Поле для ввода даты |
| Сумма в месяц | Summainmonth | Денежный | * обязательное поле – да. | Поле для ввода суммы в месяц |
| Осталось погасить | Ostalos | Денежный | * обязательное поле – да. | Поле для ввода остатка погашения кредита |

Таблица 21 – Описание структуры таблицы «Платежные\_поручения»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Наименование поля | Идентификатор | Тип данных | Условия и ограничения | Описание |
| Номер\_поручения | Nomer\_poruch | Числовой | * целочисленный * первичный ключ; * длинное число; * не могут быть отрицательными; * обязательное поле – да. | Используется для идентификации платежного поручения |
| Номер\_счета\_отправителя | nom\_otpravit | Текстовый | * обязательное поле – да. | Поле для ввода номера счета отправителя |

Продолжение таблицы 21

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Наименование поля | Идентификатор | Тип данных | Условия и ограничения | Описание |
| Номер\_счета\_получателя | Nom\_poluchat | Текстовый | * обязательное поле – да. | Поле для ввода номера счета получателя |
| Дата | Date | Текстовый | * обязательное поле – да. | Поле для ввода даты |
| Сумма | Summa | Денежный | * обязательное поле – да. | Поле для ввода суммы |

Таблица 22 – Описание структуры таблицы «Операции»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Наименование поля | Идентификатор | Тип данных | Условия и ограничения | Описание |
| Код\_Операции | Id\_oper | Числовой | * целочисленный * внешний ключ; * длинное число; * не могут быть отрицательными; * обязательное поле – да. | Используется для идентификации операции |
| Код\_кредита | Id\_kredita | Числовой | * целочисленный * внешний ключ; * длинное число; * не могут быть отрицательными; * обязательное поле – нет. | Используется для идентификации кредита |
| Номер\_поручения | Nomer\_poruch | Числовой | * целочисленный * внешний ключ; * длинное число; * не могут быть отрицательными; * обязательное поле – нет. | Используется для идентификации платежного поручения |

Продолжение таблицы 22

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Наименование поля | Идентификатор | Тип данных | Условия и ограничения | Описание |
| Код\_вклада | Id\_vklada | Числовой | * целочисленный * внешний ключ; * длинное число; * не могут быть отрицательными; * обязательное поле – нет. | Используется для идентификации вклада |
| Код\_пользователя | Id\_pol | Числовой | * целочисленный * внешний ключ; * длинное число; * не могут быть отрицательными; * обязательное поле – да. | Используется для идентификации пользователя |
| Номер\_счета | Nomer\_scheta | Числовой | * целочисленный * внешний ключ; * длинное число; * не могут быть отрицательными; * обязательное поле – да. | Используется для идентификации номера счета |
| Наименование | Name | Текстовый | * обязательное поле – да. | Поле для ввода наименования операции |
| Дата | Date | Текстовый | * обязательное поле – да. | Поле для ввода даты операции |

### SQL-сценарий разработки БД

Листинг SQL запросов представлен в приложении 9.

## Логическая модель программы

### Диаграмма вариантов использования

Диаграмма вариантов использования представлена в приложении 10.

### Описание вариантов использования

Описание вариантов использования представлено в таблицах 23 - 30.

Таблица 23 – Описание сценария «Авторизация»

|  |  |
| --- | --- |
| Вариант использования | Авторизация |
| Актёры | Администратор, операционист |
| Краткое описание | Вход в программу |
| Цель | Авторизация пользователей, доступ к необходимому функционалу |
| Тип | Базовый |
| Ссылки на другие варианты использования | Включает в себя варианты использования: ввод данных, оформление операции, добавление сотрудника, формирование отчетов. |

Таблица 24 – Описание сценария «Ввод данных»

|  |  |
| --- | --- |
| Вариант использования | Ввод данных |
| Актёры | Администратор, операционист |
| Краткое описание | Добавление клиента в базу. |
| Цель | Формирование информации о клиенте |
| Тип | Базовый |
| Ссылки на другие варианты использования | Не включает |

Таблица 25 – Описание сценария «Оформление отчётов»

|  |  |
| --- | --- |
| Вариант использования | Формирование отчётов |
| Актёры | Администратор, операционист |
| Краткое описание | Формируются отчёты |
| Цель | Вывод в документ отчетов |
| Тип | Базовый |
| Ссылки на другие варианты использования | Включает в себя варианты использования:  выписка по счету клиента, оборот по счету за период. |

Таблица 26 – Описание сценария «Добавление сотрудника»

|  |  |
| --- | --- |
| Вариант использования | Добавление сотрудника |
| Актёры | Администратор |
| Краткое описание | Добавляют новый сотрудников банка |
| Цель | Добавить новых сотрудников |
| Тип | Базовый |
| Ссылки на другие варианты использования | Не включает |

Таблица 27 – Описание сценария «Оформление операции»

|  |  |
| --- | --- |
| Вариант использования | Оформление операции |
| Актёры | Администратор, операционист |
| Краткое описание | Добавляет операцию в базу |
| Цель | Добавить новую операцию |
| Тип | Базовый |
| Ссылки на другие варианты использования | Включает в себя варианты использования:  оформление кредита, оформление вклада, оформление платежного поручения. |

Таблица 28 – Описание сценария «Оформление кредита»

|  |  |
| --- | --- |
| Вариант использования | Оформление кредита |
| Актёры | Администратор, операционист |
| Краткое описание | Добавляет кредит в базу |
| Цель | Оформление кредита |
| Тип | Базовый |
| Ссылки на другие варианты использования | Не включает |

Таблица 29 – Описание сценария «Оформление вклада»

|  |  |
| --- | --- |
| Вариант использования | Оформление вклада |
| Актёры | Администратор, операционист |
| Краткое описание | Добавляет вклад в базу |
| Цель | Оформление вклада |
| Тип | Базовый |
| Ссылки на другие варианты использования | Не включает |

Таблица 30 – Описание сценария «Оформление платежного поручения»

|  |  |
| --- | --- |
| Вариант использования | Оформление платежного поручения |
| Актёры | Администратор, операционист |
| Краткое описание | Добавляет платежное поручение в базу |
| Цель | Оформление платежного поручения |
| Тип | Базовый |
| Ссылки на другие варианты использования | Не включает |

### Диаграмма последовательности

Диаграмма последовательности представлена в приложении 11.

## Разработка пользовательского интерфейса программы

### Разработка требований к интерфейсу

Основные требования к пользовательскому интерфейсу:

* правило доступности;
* правило эффективности;
* правило непрерывного развития;
* правило поддержки;
* правило соблюдения контекста.

Требования к расположению элементов управления:

* элементы меню следует группировать;
* группы следует разделять либо разделительными полосками, либо «визуальными паузами».

### Проектирование и описание элементов управления

Интерфейс приложения представлен на рисунках 21 - 33.

Компоненты и свойства приложения представлены в таблицах 31 - 43.

Форма «Авторизация»

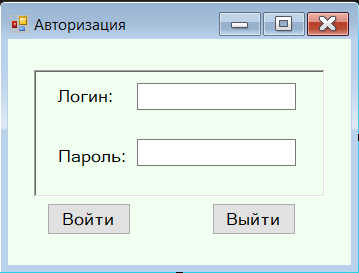


Рис. 21 – Форма «Авторизация»

Таблица 31 – Компоненты и свойства формы «Авторизация»

|  |  |  |
| --- | --- | --- |
| Компонент | Наименование компонента | Свойства компонента |
| Label | LblLogin | Text = “Логин:” |
| LblPass | Text = “Пароль:” |
| TextBox | TXTLogin | Text = “” |
| TXTPass | Text = “” |
| Button | BTNInt | Text = “Войти”, событие OnClick |
| BTNOut | Text = “Выйти”, событие OnClick |
| Panel | panel1 | BorderStyle= Fixed3D |

Форма «Главное меню»

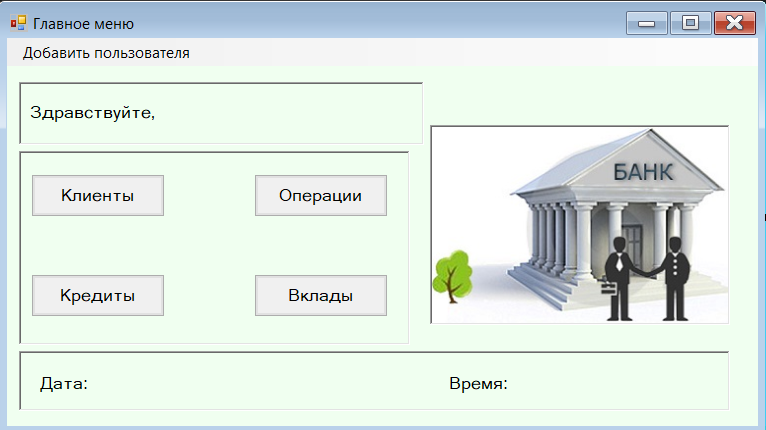


Рис. 22 –Форма «Главное меню»

Таблица 32 – Компоненты и свойства формы «Главное меню»

|  |  |  |
| --- | --- | --- |
| Компонент | Наименование компонента | Свойства компонента |
| Label | Label1 | Text = “Здравствуйте,” |
| Ldate | Text = “Дата:” |
| Ltime | Text = “Время:” |
| Button | BtnClient | Text = “Клиенты”  Событие OnClick |
| BtnOperacii | Text = “Операции”  Событие OnClick |
| Btnkredity | Text = “Кредиты”  Событие OnClick |
| BtnVklady | Text = “Вклады”  Событие OnClick |
| Panel | Panel1 | BorderStyle= Fixed3D |
| Panel2 | BorderStyle= Fixed3D |
| Panel3 | BorderStyle= Fixed3D |
| MenuStrip | MenuStrip1 | Items= «Добавить пользователя» |
| PictureBox | pictureBox1 | Image=«Bank.Properties.Resources.bank\_fin\_uch\_» |

Форма «Клиенты»

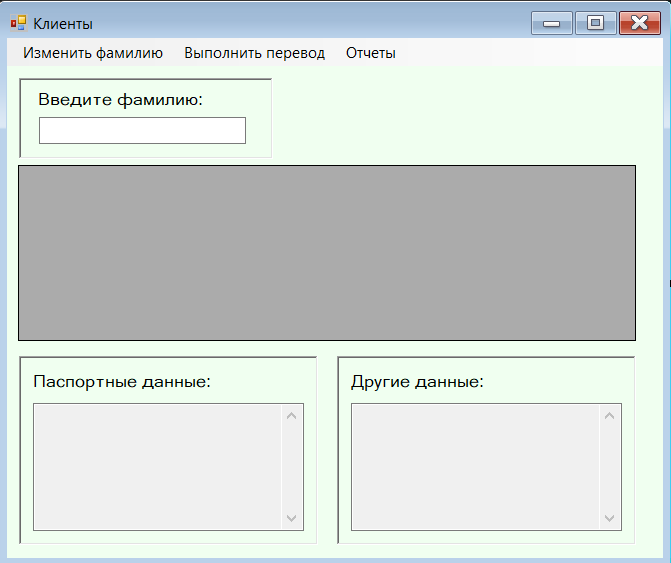


Рис. 23 –Форма «Клиенты»

Таблица 33 – Компоненты и свойства формы «Клиенты»

|  |  |  |
| --- | --- | --- |
| Компонент | Наименование компонента | Свойства компонента |
| Label | Lsearch | Text = “Введите фамилию:” |
| label1 | Text = “Паспортные данные:” |
| label2 | Text = “Другие данные:” |
| menuStrip | menuStrip1 | Items= «Изменить фамилию, выполнить перевод, Отчеты» |
| TextBox | Txtsearch | Text = “” |
| Txtpas | Text = “” |
| TxtDan | Text = “” |
| dataGridView | DGKlient | - |
| Panel | Panel1 | BorderStyle= Fixed3D |
| Panel2 | BorderStyle= Fixed3D |
| Panel3 | BorderStyle= Fixed3D |

Форма «Изменение фамилии»

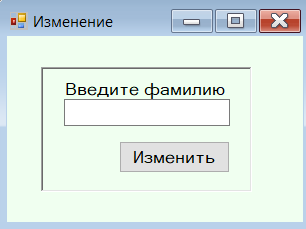


Рис. 24 – Форма «Изменение фамилии»

Таблица 34 – Компоненты и свойства формы «Изменение фамилии»

|  |  |  |
| --- | --- | --- |
| Компонент | Наименование компонента | Свойства компонента |
| Label | Label1 | Text = “Введите фамилию” |
| Button | BtnFinish | Text = “Изменить”  Событие OnClick |
| TextBox | TxtName | Text = “” |
| Panel | panel1 | BorderStyle= Fixed3D |

Форма «Перевод»

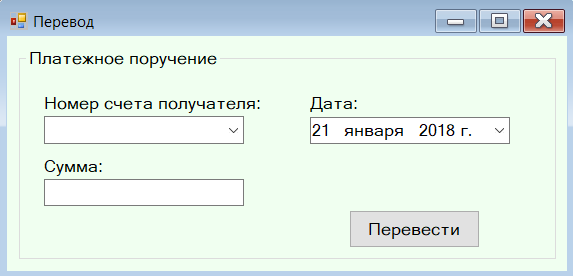


Рис. 25 – Форма «Перевод»

Таблица 35 – Компоненты и свойства формы «Перевод»

|  |  |  |
| --- | --- | --- |
| Компонент | Наименование компонента | Свойства компонента |
| Label | LlbNomer | Text = “Номер счета получателя:” |
| LblSum | Text = “Сумма:” |
| LblDate | Text = “Дата:” |
| Button | BtnNextPlat | Text = “Перевести”  Событие OnClick |
| TextBox | TxtSumperev | Text = “” |
| ComboBox | Cbnompol | Items = “” |
| DateTimePicker | DTperev | - |
| groupBox | Gbplat | Text = “Платежное поручение” |

Форма «Выписка по счету»

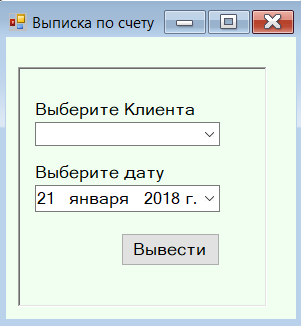


Рис. 26 – Форма «Выписка по счету»

Таблица 36 – Компоненты и свойства формы «Выписка по счету»

|  |  |  |
| --- | --- | --- |
| Компонент | Наименование компонента | Свойства компонента |
| Label | Label1 | Text = “Выберите дату” |
| Label2 | Text = “Выберите Клиента” |
| Button | BtnVyvod | Text = “Вывести”  Событие OnClick |
| ComboBox | Cbklient | Items=«» |
| dateTimePicker | DtDate | - |
| Panel | panel1 | BorderStyle= Fixed3D |

Форма «Оборот по счету»

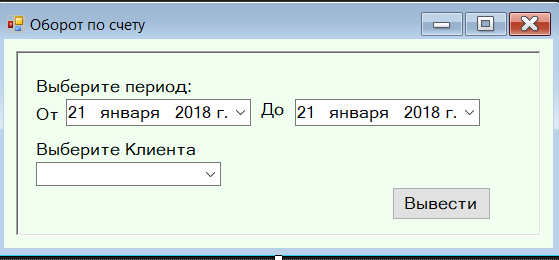


Рис. 27 – Форма «Оборот по счету»

Таблица 37 – Компоненты и свойства формы «Оборот по счету»

|  |  |  |
| --- | --- | --- |
| Компонент | Наименование компонента | Свойства компонента |
| Label | Label1 | Text = “Выберите период:” |
| Label2 | Text = “До” |
| Label3 | Text = “От” |
| Label4 | Text = “Выберите Клиента” |
| Button | BtnVyvod | Text = “Вывести”  Событие OnClick |
| ComboBox | Cbklient | Items=«» |
| dateTimePicker | Dtot | - |
| Dtdo | - |
| Panel | panel1 | BorderStyle= Fixed3D |

Форма «Вклады»

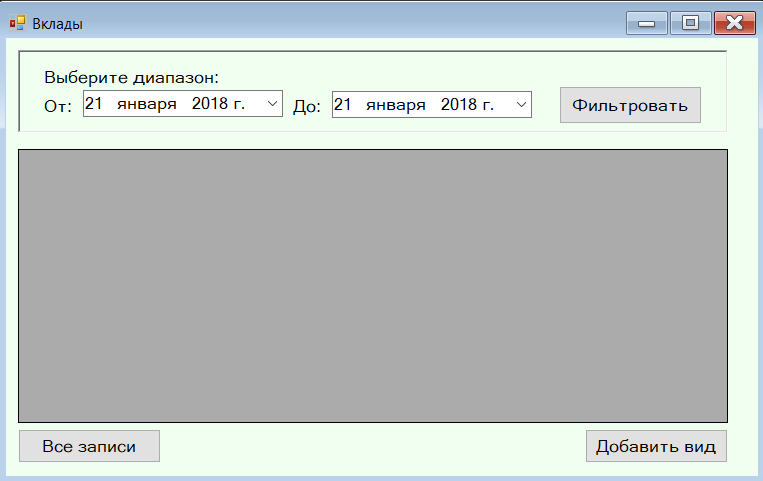


Рис. 28 –Форма «Вклады»

Таблица 38 – Компоненты и свойства формы «Вклады»

|  |  |  |
| --- | --- | --- |
| Компонент | Наименование компонента | Свойства компонента |
| Label | Label1 | Text = “Выберите диапазон:” |
| Label2 | Text = “От:” |
| Label3 | Text = “До:” |
| Button | Btnfilt | Text = “Фильтровать”  Событие OnClick |
| Btnallzapis | Text = “Все записи”  Событие OnClick |
| Frmaddvyd | Text = “Добавить вид”  Событие OnClick |
| dateTimePicker | DtpOt | - |
| DtpDo | - |
| dataGridView | Dtvklady | - |
| Panel | panel1 | BorderStyle= Fixed3D |

Форма «Операции»

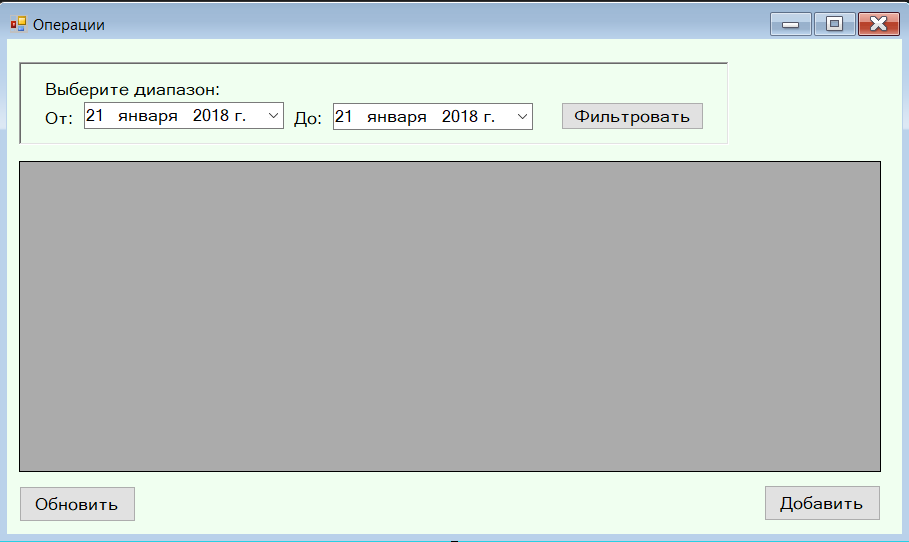


Рис. 29 – Форма «Операции»

Таблица 39 – Компоненты и свойства формы «Операции»

|  |  |  |
| --- | --- | --- |
| Компонент | Наименование компонента | Свойства компонента |
| Label | Label1 | Text = “Выберите диапазон:” |
| Label2 | Text = “От:” |
| Label3 | Text = “До:” |
| Button | Btnfilt | Text = “Фильтровать”  Событие OnClick |
| BtnReset | Text = “Обновить”  Событие OnClick |
| BtnAddoper | Text = “Добавить”  Событие OnClick |
| dateTimePicker | DtpOt | - |
| DtpDo | - |
| dataGridView | DtOper | - |
| Panel | panel1 | BorderStyle= Fixed3D |

Форма «Кредиты»

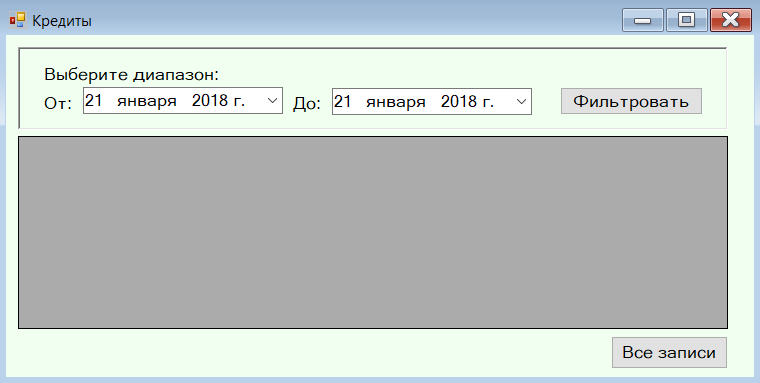


Рис. 30 – Форма «Кредиты»

Таблица 40 – Компоненты и свойства формы «Кредиты»

|  |  |  |
| --- | --- | --- |
| Компонент | Наименование компонента | Свойства компонента |
| Label | Label1 | Text = “Выберите диапазон:” |
| Label2 | Text = “От:” |
| Label3 | Text = “До:” |
| Button | Btnfilt | Text = “Фильтровать”  Событие OnClick |
| Btnobnov | Text = “Все записи”  Событие OnClick |
| dateTimePicker | DtpOt | - |
| DtpDo | - |
| dataGridView | Dtkredit | - |
| Panel | panel1 | BorderStyle= Fixed3D |

Форма «Добавление вида вклада»

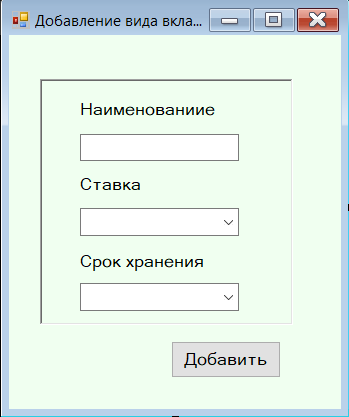


Рис. 31 – Форма «Добавление вида вклада»

Таблица 41 – Компоненты и свойства формы «Добавление вида вклада»

|  |  |  |
| --- | --- | --- |
| Компонент | Наименование компонента | Свойства компонента |
| Label | Label1 | Text = “Наименование” |
| Label2 | Text = “Ставка” |
| Label3 | Text = “Срок хранения” |
| Button | BtnAdd | Text = “Добавить”  Событие OnClick |
| TextBox | TxtNAme | Text = “” |
| ComboBox | Cbstavka | Items=«5,6,7,8,9,10» |
| Cbsrok | Items=«1,3,5,10» |
| Panel | panel1 | BorderStyle= Fixed3D |

Форма «Добавление»

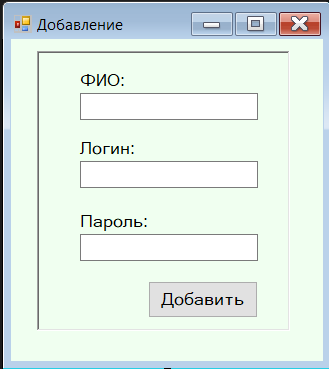


Рис. 32 – Форма «Добавление»

Таблица 42 – Компоненты и свойства формы «Добавление»

|  |  |  |
| --- | --- | --- |
| Компонент | Наименование компонента | Свойства компонента |
| Label | Label1 | Text = “ФИО:” |
| Label2 | Text = “Логин:” |
| Label3 | Text = “Пароль:” |
| Button | BtnAdd | Text = “Добавить”  Событие OnClick |
| TextBox | TxtFio | Text = “” |
| TxtLogin | Text = “” |
| TxtPass | Text = “” |
| Panel | panel1 | BorderStyle= Fixed3D |

Форма «Добавление операции»

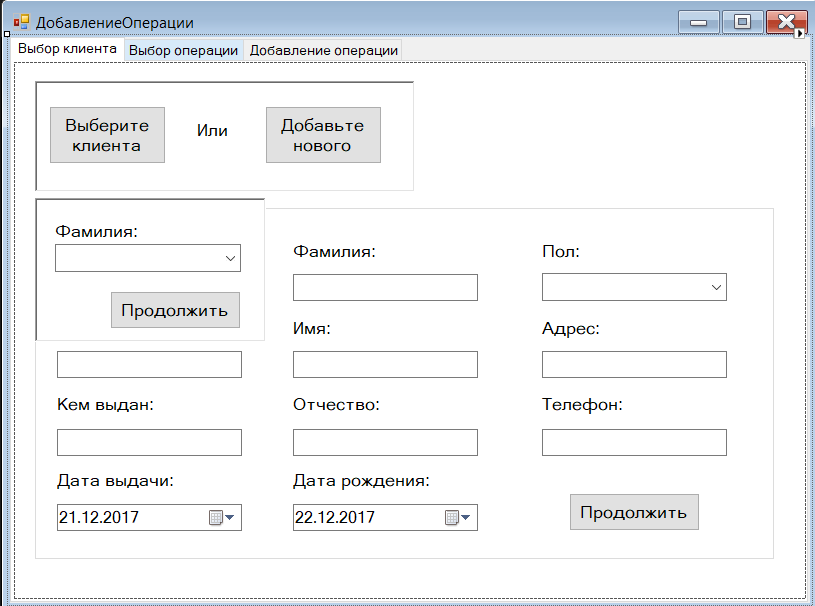


Рис. 33 – Форма «Добавление операции»

Таблица 43 – Компоненты и свойства формы «Добавление операции»

|  |  |  |
| --- | --- | --- |
| Компонент | Наименование компонента | Свойства компонента |
| Label | Label1 | Text = “Или” |
| Label2 | Text = “Серия паспорта:” |
| Label3 | Text = “Номер паспорта:” |
| Label4 | Text = “Кем выдан:” |
| Label5 | Text = “Дата выдачи:” |
| Label6 | Text = “Фамилия:” |
| Label7 | Text = “Имя:” |
| Label8 | Text = “Отчество:” |
| Label9 | Text = “Дата рождения:” |
| Label10 | Text = “Пол:” |
| Label11 | Text = “Адрес:” |
| Label12 | Text = “Телефон:” |

Продолжение таблицы 43

|  |  |  |
| --- | --- | --- |
| Компонент | Наименование компонента | Свойства компонента |
| Label | Label13 | Text = “Выберите операцию:” |
| Label23 | Text = “Номер договора:” |
| Label22 | Text = “Вид вклада:” |
| Label20 | Text = “Дата:” |
| Label19 | Text = “Сумма” |
| Label14 | Text = “Номер договора:” |
| Label15 | Text = “Сумма:” |
| Label16 | Text = “Процентная ставка:” |
| Label17 | Text = “Дата кредитования:” |
| Label18 | Text = “Количество месяцев:” |
| Label28 | Text = “Дата:” |
| Label30 | Text = “Наименование:” |
| Label31 | Text = “Сотрудник:” |
| Button | Btnselectklient | Text = “Выберите клиента”  Событие OnClick |
| Btnaddklient | Text = “Добавьте нового”  Событие OnClick |
| Btnselectoper | Text = “Продолжить”  Событие OnClick |
| BtnNext | Text = “Продолжить”  Событие OnClick |
| BtnNextselect | Text = “Продолжить”  Событие OnClick |
| BtnNextOPervklad | Text = “Продолжить”  Событие OnClick |
| BtnNextOperkredit | Text = “Продолжить”  Событие OnClick |
| BtnAddOperac | Text = “Завершить”  Событие OnClick |
| TextBox | Txtseria | Text = “” |
| Txtnomer | Text = “” |

Продолжение таблицы 43

|  |  |  |
| --- | --- | --- |
| Компонент | Наименование компонента | Свойства компонента |
| TextBox | Txtkemvyd | Text = “” |
| TxtLastname | Text = “” |
| TxtFirstname | Text = “” |
| TxtOtchestvo | Text = “” |
| TxtAdres | Text = “” |
| TxtTelephone | Text = “” |
| Txtnomvklada | Text = “” |
| Txtsumvklada | Text = “” |
| Txtnomdogkred | Text = “” |
| Txtsumkred | Text = “” |
| Panel | Pklient | BorderStyle= Fixed3D |
| Pselectfam | BorderStyle= Fixed3D |
| Pselectoper | BorderStyle= Fixed3D |
| GroupBox | Gbklient | Text = “Добавление клиента” |
| GbVklad | Text = “Вклад” |
| GbKredit | Text = “Кредит” |
| GpOper | Text = “Операция” |
| ComboBox | CBFamalyklienta | Items=«» |
| CBPol | Items=«Муж, Жен» |
| Cmbselectoper | Items=«Кредитование, Вклады» |
| CBStavkakredit | Items=«12,13,14,15,16,17,18,19,20,21,22,23» |
| CBKolmeskrdit | Items=«12,24,36,48» |
| Cbvydvklada | Items=«» |
| CbNameoper | Items=«Кредит, Вклад» |
| Cbsotrud | Items=«» |
| DateTimePicker | DtpDatevyd | - |
| DtpDatebirth | - |
| DtKredit | - |
| DtDatevklada | - |
| Dtdateoper | - |
| TabControl | TPclient | TabPages=«Выбор клиента» |

Продолжение таблицы 43

|  |  |  |
| --- | --- | --- |
| Компонент | Наименование компонента | Свойства компонента |
| TabControl | TPVyboper | TabPages=«Выбор операции» |
| TPaddoper | TabPages=«Добавление операции» |

## Листинг программы

Листинг приложения представлен в приложении 12.

## Разработка тестовых данных

Тестирования существования БД представлено в таблице 44.

Таблица 44 - Тестирование существования БД

|  |  |
| --- | --- |
| Наименование | Описание |
| Test Case # | TC\_UI\_1 |
| Приоритет теста | Высокий |
| Название тестирования/Имя | Проверка существования БД при запуске приложения с существующей БД |
| Резюме испытания | Проверка существования БД |
| Шаги тестирования | Запустить приложение |
| Данные тестирования | Запустить приложение, БД находится вместе с exe файлом |
| Ожидаемый результат | Произойдет запуск приложения без создания новых файлов. |
| Фактический результат | Запуск приложения |
| Предпосылки | - |

Продолжение таблицы 44

|  |  |
| --- | --- |
| Наименование | Описание |
| Постусловия | Откроется окно авторизации |
| Статус  (Pass/Fail) | Pass |
| Комментарии | - |

Тестирования формы «Авторизация» представлены в таблицах 45 - 46.

Таблица 45 – Тестирование формы «Авторизация»

|  |  |
| --- | --- |
| Наименование | Описание |
| Test Case # | TC\_UI\_2 |
| Приоритет теста | Высокий |
| Название тестирования/Имя | Проверка формы авторизации пользователей при вводе неправильного пароля |
| Резюме испытания | Проверка входа в приложение |
| Шаги тестирования | 1. Запустить приложение; 2. ввести логин пользователя; 3. ввести пароль; 4. нажать на кнопку «Войти». |
| Данные тестирования | Логин= polsovatel, Пароль = polsovatel |
| Ожидаемый результат | Сообщение об ошибке «Неверно введен логин/пароль» |
| Фактический результат | Приложение выдало сообщение об ошибке «Неправильно введен логин/пароль» |
| Предпосылки | - |
| Постусловия | Запрет на вход в приложение |
| Статус  (Pass/Fail) | Pass |
| Комментарии | - |

Таблица 46 – Тестирование формы «Авторизация»

|  |  |
| --- | --- |
| Наименование | Описание |
| Test Case # | TC\_UI\_3 |
| Приоритет теста | Высокий |
| Название тестирования/Имя | Проверка формы авторизации пользователей при вводе верных данных |
| Резюме испытания | Проверка входа в приложение |
| Шаги тестирования | 1. Запустить приложение; 2. ввести логин пользователя; 3. ввести пароль; 4. нажать на кнопку «Войти». |
| Данные тестирования | Логин= admin  Пароль = admin |
| Ожидаемый результат | Переход на форму «Главное меню» |
| Фактический результат | Открылась форма «Главное меню» |
| Предпосылки | - |
| Постусловия | Запрет на вход в приложение |
| Статус  (Pass/Fail) | Pass |
| Комментарии | - |

Тестирование формы «Добавление операции» представлено в таблице 47-48.

Таблица 47 - Тестирование формы «Добавление операции»

|  |  |
| --- | --- |
| Наименование | Описание |
| Test Case # | TC\_UI\_4 |
| Приоритет теста | Высокий |
| Название тестирования/Имя | Добавление клиента в базу |
| Резюме испытания | Проверка добавления клиента |

Продолжение таблицы 47

|  |  |
| --- | --- |
| Наименование | Описание |
| Шаги тестирования | 1. Нажать на кнопку «добавить клиента»; 2. Ввести данные; 3. Нажать кнопку продолжить. |
| Данные тестирования | Серия= «1234»  Номер= «123432»  Кем выдан= «Серпухов»  Дата выдачи= «12.02.2012»  Фамилия= «Иванов»  Имя= «Иван»  Отчество= «Иванович»  Дата рождения= «12.02.1996»  Пол= «Мужской»  Адрес= «Серпухов»  Телефон= «89647321824» |
| Ожидаемый результат | Переход на вкладку «Выбор операции» |
| Фактический результат | Перешла на вкладку «Выбор операции» |
| Предпосылки | Производен вход в приложение, нажат пункт меню «Операции» и нажата кнопка добавить |
| Постусловия | - |
| Статус  (Pass/Fail) | Pass |
| Комментарии | - |

Таблица 48 - Тестирование формы «Добавление операции»

|  |  |
| --- | --- |
| Наименование | Описание |
| Test Case # | TC\_UI\_5 |
| Приоритет теста | Высокий |
| Название тестирования/Имя | Добавление клиента в базу |
| Резюме испытания | Проверка добавления клиента |
| Шаги тестирования | 1. Нажать на кнопку «добавить клиента»; 2. Ввести данные; 3. Нажать кнопку продолжить. |
| Данные тестирования | Серия= «1234»  Номер= «123432»  Кем выдан= «Серпухов»  Дата выдачи= «12.02.2012»  Фамилия= «Иванов»  Имя= «Иван»  Отчество= «Иванович»  Дата рождения= «12.02.1996»  Пол= «Мужской»  Адрес= «Серпухов»  Телефон= «89647321824» |
| Ожидаемый результат | Сообщение об ошибке «Клиент с такими паспортными данными уже есть в базе» |
| Фактический результат | Приложение выдало сообщение об ошибке «Клиент с такими паспортными данными уже есть в базе» |
| Предпосылки | Производен вход в приложение, нажат пункт меню «Операции» и нажата кнопка добавить |
| Постусловия | - |

Продолжение таблицы 48

|  |  |
| --- | --- |
| Наименование | Описание |
| Статус  (Pass/Fail) | Pass |
| Комментарии | - |

Тестирование формы «Добавление сотрудника» представлено в таблице 49.

Таблица 49 - Тестирование формы «Список пациентов»

|  |  |
| --- | --- |
| Наименование | Описание |
| Test Case # | TC\_UI\_6 |
| Приоритет теста | Высокий |
| Название тестирования/Имя | Добавление сотрудника в базу |
| Резюме испытания | Проверка добавления сотрудника |
| Шаги тестирования | 1. Нажать кнопку в главном меню «Добавить сотрудника»; 2. Ввести данные; 3. Нажать кнопку «Добавить». |
| Данные тестирования | ФИО= «Петров»  Логин= «Petrov»  Пароль= «Petrov» |
| Ожидаемый результат | Сообщение «Пользователь добавлен» |
| Фактический результат | Приложение выдало сообщение «Пользователь добавлен» |
| Предпосылки | Производен вход в приложение |
| Постусловия | - |
| Статус  (Pass/Fail) | Pass |
| Комментарии | - |

Тестирование формы «Перевод» представлено в таблице 50.

Таблица 50 - Тестирование формы «Перевод»

|  |  |
| --- | --- |
| Наименование | Описание |
| Test Case # | TC\_UI\_7 |
| Приоритет теста | Высокий |
| Название тестирования/Имя | Добавление платежного поручения в базу |
| Резюме испытания | Проверка добавление платежного поручения |
| Шаги тестирования | 1. Выбрать номер счета получателя; 2. Ввести Сумма; 3. Выбрать дату; 4. Нажать кнопку перевести. |
| Данные тестирования | Номер счета = 2  Сумма = 5000  Дата = 21.12.2017 |
| Ожидаемый результат | Сообщение пользователю «Перевод выполнен» |
| Фактический результат | Сообщение пользователю «Перевод выполнен» |
| Предпосылки | Производен вход в приложение, нажат пункт меню «Клиенты» и нажата кнопка «Выполнить перевод» |
| Постусловия | Открытие формы «Клиенты» |
| Статус (Pass/Fail) | Pass |
| Комментарии |  |

Тестирование формы «Оборот по счету» представлено в таблице 51.

Таблица 51 - Тестирование формы «Оборот по счету»

|  |  |
| --- | --- |
| Наименование | Описание |
| Test Case # | TC\_UI\_8 |
| Приоритет теста | Высокий |
| Название тестирования/Имя | Проверка создание документа «Оборот по счету» из существующего файла шаблона документа |
| Резюме испытания | Проверка создание документа «Оборот по счету» |

Продолжение таблицы 51

|  |  |
| --- | --- |
| Наименование | Описание |
| Шаги тестирования | 1. Открыть форму «Клиенты»; 2. Навести на пункт меню «Отчеты» и выбрать «Оборот по счету»; 3. Выбрать период; 4. Выбрать клиента. |
| Данные тестирования | Нажать на кнопку «Печать» файл шаблона документа находиться рядом с exe файлом |
| Ожидаемый результат | В директории с exe файлом приложения создастся файл «Оборот по счету.docx» |
| Фактический результат | В директории с exe файлом приложения создастся файл «Оборот по счету.docx» |
| Предпосылки | Производен вход в приложение |
| Постусловия | - |
| Статус  (Pass/Fail) | Pass |
| Комментарии | - |

## Результаты работы программы

Результаты работы программы представлены на рисунках 34 - 46.

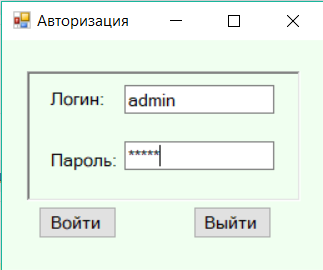


Рис. 34 – Окно «Авторизация»

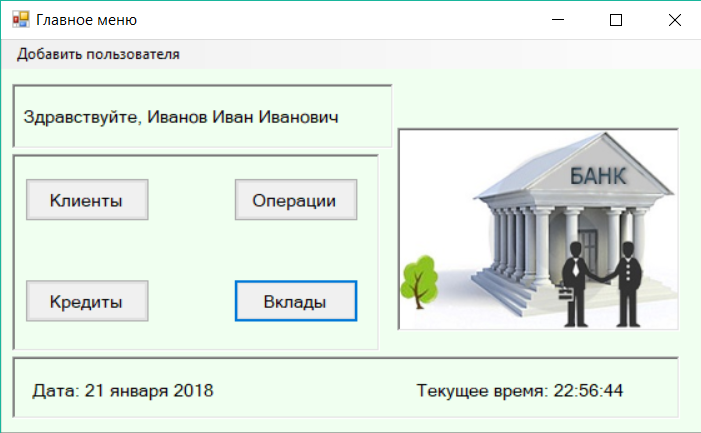


Рис. 35 – Окно «Главное меню»

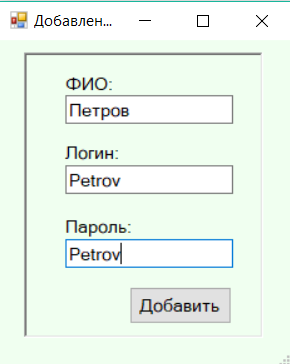


Рис. 36 – Окно «Добавление болезни»

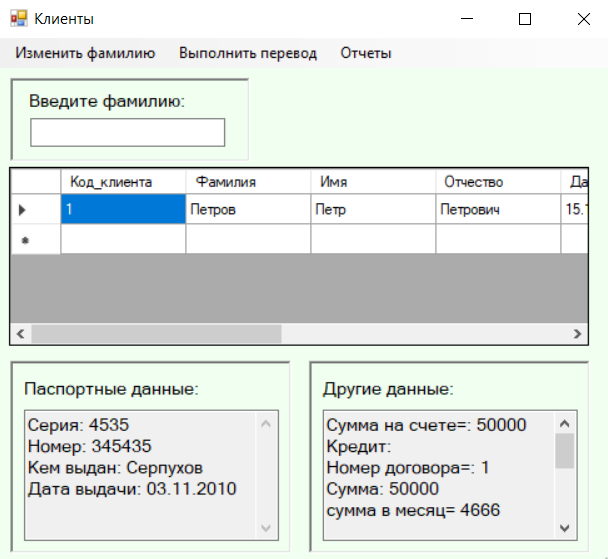


Рис. 37 – Окно «Болезни»

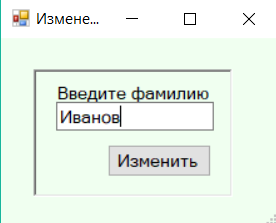


Рис. 38 – Окно «Список сотрудников»

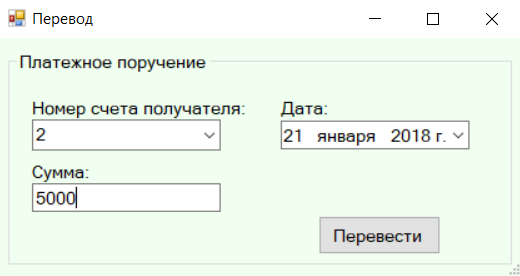


Рис. 39 – Окно «Добавление сотрудника»

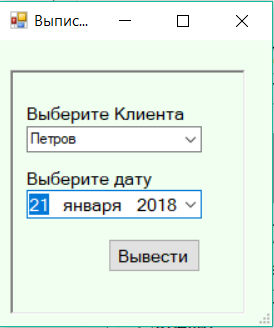


Рис. 40 – Окно «Добавление пациента»

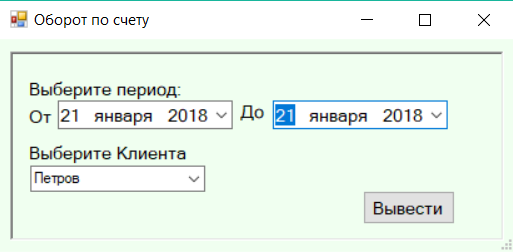


Рис. 41 – Окно «Запись на прием»

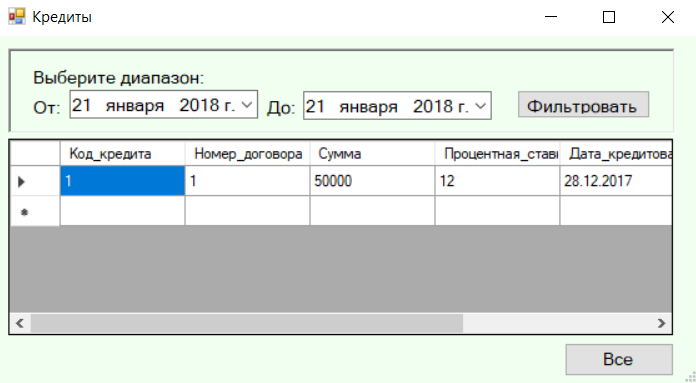


Рис. 42 – Окно «Список пациентов»

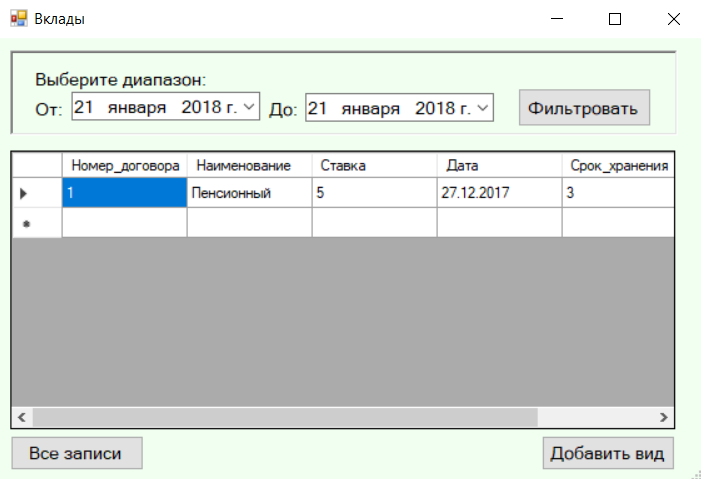


Рис. 43 – Окно «Редактирование пациента»

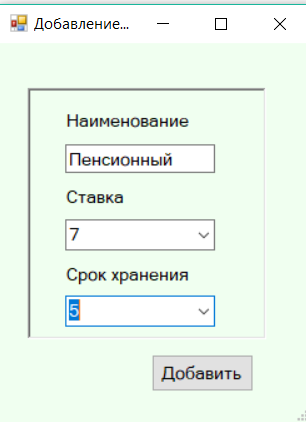


Рис. 44 – Окно «Поиск пациента»

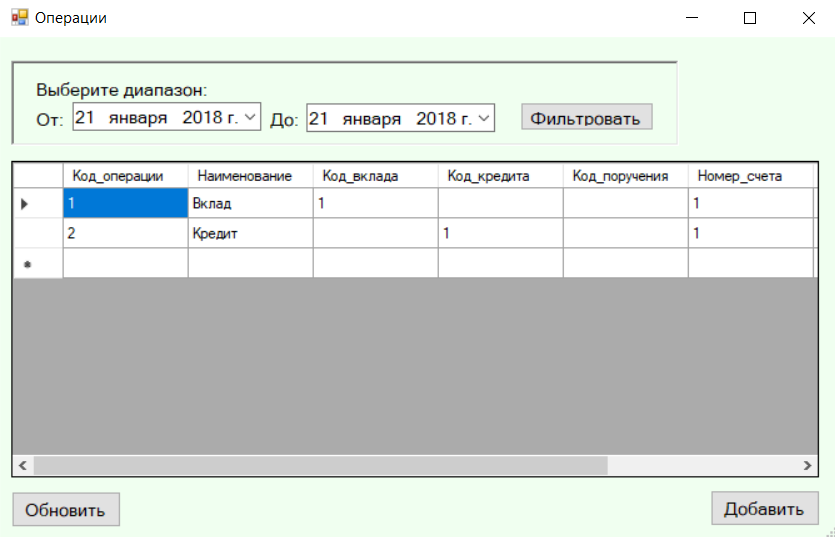


Рис. 45 – Окно «Список отделений»

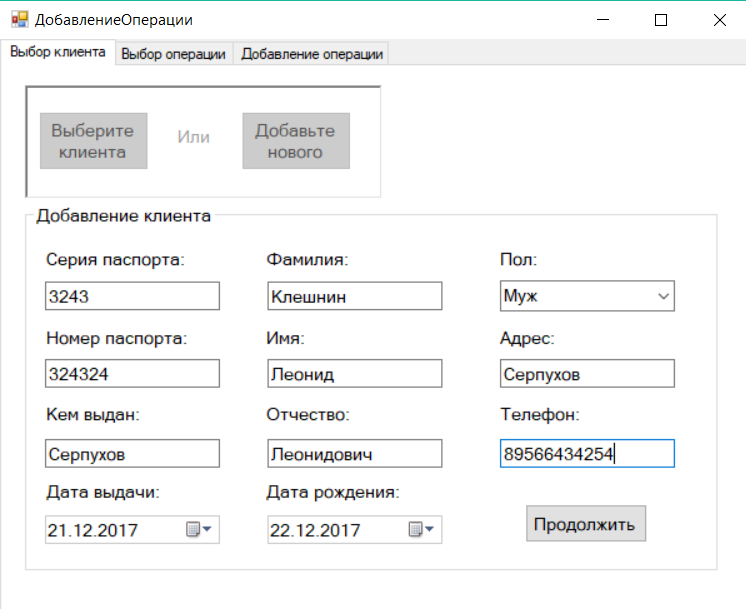


Рис. 46 – Окно «Приемы»

Результаты отчетов представлены на рисунках 47 – 48.

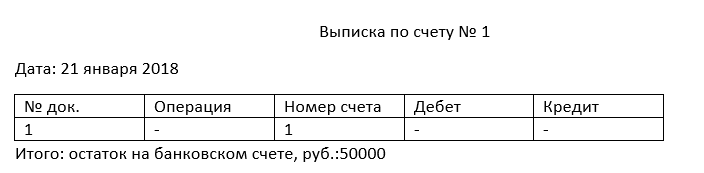


Рис. 47 – Отчет «Выписка по счету»

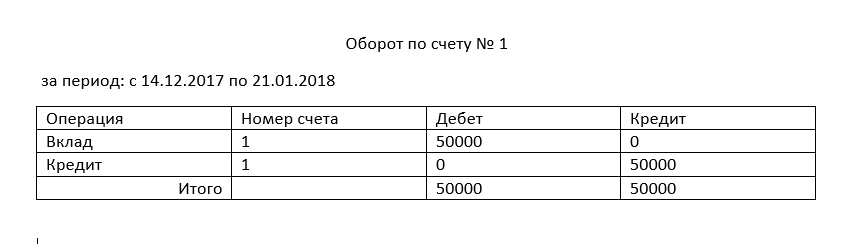


Рис. 48 – Отчет «Оборот по счету»

## Руководство по использованию программы

### Руководство системного программиста

* + - 1. Общие сведения о программе

Данная программа предназначена для автоматизации учета банковских пассивов и создания отчетных документов. Включает в себя такие функции, как: сохранение, редактирование и удаление данных в программе; вывод выписки по счету и вывод оборота по счету.

Системные требование к программе:

1. процессор: Pentium 4 с частотой 1 ГГц;
2. оперативная память: 256 МБ;
3. память на жестком диске: минимум 50 МБ свободного места на диске;
4. операционная система: минимум Windows XP.
   * + 1. Структура программы

Структура программы представлена на рисунке 39.

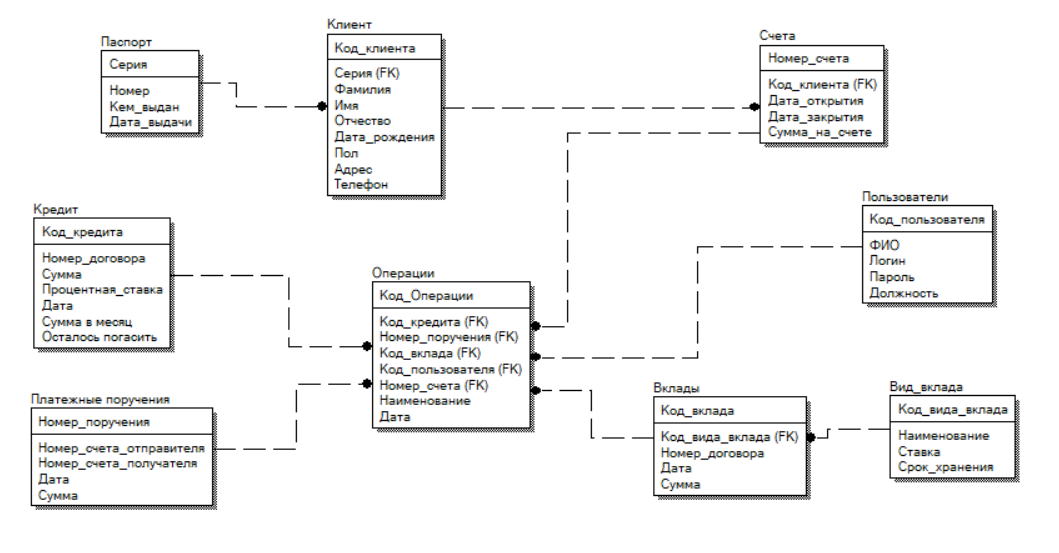


Рис. 49 – Структура программы

* + - 1. Настройка программы

Данное программное обеспечение не требует никаких настроек.

* + - 1. Проверка программы

Тестирования существования БД представлено в таблице 52.

Таблица 52 - Тестирование существования БД

|  |  |
| --- | --- |
| Наименование | Описание |
| Test Case # | TC\_UI\_1 |
| Приоритет теста | Высокий |
| Название тестирования/Имя | Проверка существования БД при запуске приложения с существующей БД |
| Резюме испытания | Проверка существования БД |
| Шаги тестирования | Запустить приложение |
| Данные тестирования | БД находится вместе с exe файлом |

Продолжение таблицы 52

|  |  |
| --- | --- |
| Наименование | Описание |
| Ожидаемый результат | Произойдет запуск приложения без создания новых файлов. |
| Фактический результат | Запуск приложения |
| Предпосылки | - |
| Постусловия | Откроется окно авторизации |
| Статус  (Pass/Fail) | Pass |
| Комментарии | - |

Тестирования формы «Авторизация» представлены в таблицах 53 - 54.

Таблица 53 – Тестирование формы «Авторизация»

|  |  |
| --- | --- |
| Наименование | Описание |
| Test Case # | TC\_UI\_2 |
| Приоритет теста | Высокий |
| Название тестирования/Имя | Проверка формы авторизации пользователей при вводе неправильного пароля |
| Резюме испытания | Проверка входа в приложение |
| Шаги тестирования | 1. Запустить приложение; 2. ввести логин пользователя; 3. ввести пароль; 4. нажать на кнопку «Войти». |
| Данные тестирования | Логин= polsovatel  Пароль = polsovatel |
| Ожидаемый результат | Сообщение об ошибке «Неверно введен логин/пароль» |
| Фактический результат | Приложение выдало сообщение об ошибке «Неправильно введен логин/пароль» |

Продолжение таблицы 53

|  |  |
| --- | --- |
| Наименование | Описание |
| Предпосылки | - |
| Постусловия | Запрет на вход в приложение |
| Статус  (Pass/Fail) | Pass |
| Комментарии | - |

Таблица 54 – Тестирование формы «Авторизация»

|  |  |
| --- | --- |
| Наименование | Описание |
| Test Case # | TC\_UI\_3 |
| Приоритет теста | Высокий |
| Название тестирования/Имя | Проверка формы авторизации пользователей при вводе верных данных |
| Резюме испытания | Проверка входа в приложение |
| Шаги тестирования | 1. Запустить приложение; 2. ввести логин пользователя; 3. ввести пароль; 4. нажать на кнопку «Войти». |
| Данные тестирования | Логин= admin  Пароль = admin |
| Ожидаемый результат | Переход на форму «Главное меню» |
| Фактический результат | Открылась форма «Главное меню» |
| Предпосылки | - |
| Постусловия | Запрет на вход в приложение |
| Статус  (Pass/Fail) | Pass |
| Комментарии | - |

Тестирование формы «Добавление операции» представлено в таблице 55-56.

Таблица 55 - Тестирование формы «Добавление операции»

|  |  |
| --- | --- |
| Наименование | Описание |
| Test Case # | TC\_UI\_4 |
| Приоритет теста | Высокий |
| Название тестирования/Имя | Добавление клиента в базу |
| Резюме испытания | Проверка добавления клиента |
| Шаги тестирования | 1. Нажать на кнопку «добавить клиента»; 2. Ввести данные; 3. Нажать кнопку продолжить. |
| Данные тестирования | Серия= «1234»  Номер= «123432»  Кем выдан= «Серпухов»  Дата выдачи= «12.02.2012»  Фамилия= «Иванов»  Имя= «Иван»  Отчество= «Иванович»  Дата рождения= «12.02.1996»  Пол= «Мужской»  Адрес= «Серпухов»  Телефон= «89647321824» |
| Ожидаемый результат | Переход на вкладку «Выбор операции» |
| Фактический результат | Перешла на вкладку «Выбор операции» |
| Предпосылки | Производен вход в приложение, нажат пункт меню «Операции» и нажата кнопка добавить |
| Постусловия | - |

Продолжение таблицы 55

|  |  |
| --- | --- |
| Наименование | Описание |
| Статус  (Pass/Fail) | Pass |
| Комментарии | - |

Таблица 56 - Тестирование формы «Добавление операции»

|  |  |
| --- | --- |
| Наименование | Описание |
| Test Case # | TC\_UI\_4 |
| Приоритет теста | Высокий |
| Название тестирования/Имя | Добавление клиента в базу |
| Резюме испытания | Проверка добавления клиента |
| Шаги тестирования | 1. Нажать на кнопку «добавить клиента»; 2. Ввести данные; 3. Нажать кнопку продолжить. |
| Данные тестирования | Серия= «1234»  Номер= «123432»  Кем выдан= «Серпухов»  Дата выдачи= «12.02.2012»  Фамилия= «Иванов»  Имя= «Иван»  Отчество= «Иванович»  Дата рождения= «12.02.1996»  Пол= «Мужской»  Адрес= «Серпухов»  Телефон= «89647321824» |

Продолжение таблицы 56

|  |  |
| --- | --- |
| Наименование | Описание |
| Ожидаемый результат | Сообщение об ошибке «Клиент с такими паспортными данными уже есть в базе» |
| Фактический результат | Приложение выдало сообщение об ошибке «Клиент с такими паспортными данными уже есть в базе» |
| Предпосылки | Производен вход в приложение, нажат пункт меню «Операции» и нажата кнопка добавить |
| Постусловия | - |
| Статус  (Pass/Fail) | Pass |
| Комментарии | - |

Тестирование формы «Добавление сотрудника» представлено в таблице 57.

Таблица 57 - Тестирование формы «Список пациентов»

|  |  |
| --- | --- |
| Наименование | Описание |
| Test Case # | TC\_UI\_5 |
| Приоритет теста | Высокий |
| Название тестирования/Имя | Добавление сотрудника в базу |
| Резюме испытания | Проверка добавления сотрудника |
| Шаги тестирования | 1. Нажать кнопку в главном меню «Добавить сотрудника»; 2. Ввести данные; 3. Нажать кнопку «Добавить». |
| Данные тестирования | ФИО= «Петров»  Логин= «Petrov»  Пароль= «Petrov» |
| Ожидаемый результат | Сообщение «Пользователь добавлен» |

Продолжение таблицы 57

|  |  |
| --- | --- |
| Наименование | Описание |
| Фактический результат | Приложение выдало сообщение «Пользователь добавлен» |
| Предпосылки | Производен вход в приложение |
| Постусловия | - |
| Статус  (Pass/Fail) | Pass |
| Комментарии | - |

Тестирование формы «Перевод» представлено в таблице 58.

Таблица 58 - Тестирование формы «Перевод»

|  |  |
| --- | --- |
| Наименование | Описание |
| Test Case # | TC\_UI\_6 |
| Приоритет теста | Высокий |
| Название тестирования/Имя | Добавление платежного поручения в базу |
| Резюме испытания | Проверка добавление платежного поручения |
| Шаги тестирования | 1. Выбрать номер счета получателя; 2. Ввести Сумма; 3. Выбрать дату; 4. Нажать кнопку перевести. |
| Данные тестирования | Номер счета = 2  Сумма = 5000  Дата = 21.12.2017 |
| Ожидаемый результат | Сообщение пользователю «Перевод выполнен» |
| Фактический результат | Сообщение пользователю «Перевод выполнен» |
| Предпосылки | Производен вход в приложение, нажат пункт меню «Клиенты» и нажата кнопка «Выполнить перевод» |
| Постусловия | Открытие формы «Клиенты» |

Продолжение таблицы 58

|  |  |
| --- | --- |
| Наименование | Описание |
| Статус (Pass/Fail) | Pass |
| Комментарии |  |

Тестирование формы «Оборот по счету» представлено в таблице 59.

Таблица 59 - Тестирование формы «Оборот по счету»

|  |  |
| --- | --- |
| Наименование | Описание |
| Test Case # | TC\_UI\_5 |
| Приоритет теста | Высокий |
| Название тестирования/Имя | Проверка создание документа «Оборот по счету» из существующего файла шаблона документа |
| Резюме испытания | Проверка создание документа «Оборот по счету» |
| Шаги тестирования | 1. Открыть форму «Клиенты»; 2. Навести на пункт меню «Отчеты» и выбрать «Оборот по счету»; 3. Выбрать период; 4. Выбрать клиента. |
| Данные тестирования | Нажать на кнопку «Печать» файл шаблона документа находиться рядом с exe файлом |
| Ожидаемый результат | В директории с exe файлом приложения создастся файл «Оборот по счету.docx» |
| Фактический результат | В директории с exe файлом приложения создастся файл «Оборот по счету.docx» |
| Предпосылки | Производен вход в приложение |
| Постусловия | - |
| Статус  (Pass/Fail) | Pass |
| Комментарии | - |

* + - 1. Дополнительные возможности

Дополнительные возможности не предусмотрены.

* + - 1. Сообщения системному программисту

Сообщения системному программисту отсутствуют.

### Руководство программиста

* + - 1. Назначения и условия применения программы

Приложение разработано для автоматизации учета банковских пассивов. Программа предназначена для 2-х типов пользователей: администратор, операционист. Каждый пользователь имеет свои определенные функции.

* + - 1. Характеристики программы

Запуск программы должен быть не более 10 секунд.

Системные требование к программе:

1. процессор: Pentium 4 с частотой 1 ГГц;
2. оперативная память: 256 МБ;
3. память на жестком диске: минимум 50 МБ свободного места на диске;
4. операционная система: минимум Windows XP.
   * + 1. Обращение к программе

Для запуска следует щелкнуть на ярлычок под именем «Bank» два раза, откроется окно входа в программу.

* + - 1. Входные и выходные данные

Таблица 60 – Входные данные «Паспорт»

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование поля | Идентификатор | Тип данных | Источник данных |
| Серия | Seriy | Числовой | Справочник «Паспорт» |
| Номер | Nomer | Текстовый | Справочник «Паспорт» |
| Кем\_выдан | Kem\_vydan | Текстовый | Справочник «Паспорт» |
| Дата\_выдачи | Date\_vyd | Текстовый | Справочник «Паспорт» |

Таблица 61 – Входные данные «Клиенты»

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование поля | Идентификатор | Тип данных | Источник данных |
| Код\_клиента | Id\_klienta | Числовой | Справочник «Клиент» |
| Серия | Seriy | Числовой | Справочник «Паспорт» |
| Фамилия | lastName | Текстовый | Справочник «Клиент» |

Продолжение таблицы 61

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование поля | Идентификатор | Тип данных | Источник данных |
| Имя | Name | Текстовый | Справочник «Клиент» |
| Отчество | Otchestvo | Текстовый | Справочник «Клиент» |
| Дата\_рождения | Date\_rozh | Текстовый | Справочник «Клиент» |
| Пол | Pol | Текстовый | Справочник «Клиент» |
| Адрес | Adres | Текстовый | Справочник «Клиент» |
| Телефон | Telephone | Текстовый | Справочник «Клиент» |

Таблица 62 – Входные данные «Счета»

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование поля | Идентификатор | Тип данных | Источник данных |
| Номер\_счета | Nomer\_scheta | Числовой | Справочник «Счета» |
| Код\_клиента | Id\_klienta | Числовой | Справочник «Клиент» |
| Дата\_открытия | date\_open | Текстовый | Справочник «Счета» |
| Дата\_закрытия | date\_close | Текстовый | Справочник «Счета» |
| Сумма\_на\_счете | Summa | Текстовый | Справочник «Счета» |

Таблица 63 – Входные данные «Пользователи»

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование поля | Идентификатор | Тип данных | Источник данных |
| Код\_пользователя | Id\_pol | Числовой | Справочник «Пользователи» |
| ФИО | FIO | Текстовый | Справочник «Пользователи» |
| Логин | Login | Текстовый | Справочник «Пользователи» |
| Пароль | Pass | Текстовый | Справочник «Пользователи» |
| Должность | Dolzhnost | Текстовый | Справочник «Пользователи» |

Таблица 64 – Входные данные «Вид\_вклада»

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование поля | Идентификатор | Тип данных | Источник данных |
| Код\_вида\_вклада | id\_vid\_vklada | Числовой | Справочник «Вид\_вклада» |
| Наименование | Name | Числовой | Справочник «Вид\_вклада» |
| Ставка | stavka | Текстовый | Справочник «Вид\_вклада» |
| Срок\_хранения | Srok\_xran | Числовой | Справочник «Вид\_вклада» |

Таблица 65 – Входные данные «Вклады»

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование поля | Идентификатор | Тип данных | Источник данных |
| Код\_вклада | Id\_vklada | Числовой | Справочник «Вклады» |
| Код\_вида\_вклада | id\_vid\_vklada | Числовой | Справочник «Вид\_вклада» |

Продолжение таблицы 65

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование поля | Идентификатор | Тип данных | Источник данных |
| Номер\_договора | Name\_dog | Числовой | Справочник «Вклады» |
| Дата | Date | Текстовый | Справочник «Вклады» |
| Сумма | sum | Денежный | Справочник «Вклады» |

Таблица 66 – Входные данные «Кредит»

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование поля | Идентификатор | Тип данных | Источник данных |
| Код\_кредита | Id\_kredita | Числовой | Справочник «Кредит» |
| Номер\_договора | Nomer\_dogovora | Числовой | Справочник «Кредит» |
| Сумма | Summa | Текстовый | Справочник «Кредит» |
| Процентная\_ставка | Procenty | Текстовый | Справочник «Кредит» |
| Дата | Data | Текстовый | Справочник «Кредит» |
| Сумма в месяц | Summainmonth | Денежный | Справочник «Кредит» |
| Осталось погасить | Ostalos | Денежный | Справочник «Кредит» |

Таблица 67 – Входные данные «Платежные поручения»

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование поля | Идентификатор | Тип данных | Источник данных |
| Номер\_поручения | Nomer\_poruch | Числовой | Справочник «Платежные поручения» |
| Номер\_счета\_отправителя | nom\_otpravit | Текстовый | Справочник «Платежные поручения» |
| Номер\_счета\_получателя | Nom\_poluchat | Текстовый | Справочник «Платежные поручения» |
| Дата | Date | Текстовый | Справочник «Платежные поручения» |
| Сумма | Summa | Денежный | Справочник «Платежные поручения» |

Таблица 68 – Входные данные «Операции»

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование поля | Идентификатор | Тип данных | Источник данных |
| Код\_Операции | Id\_oper | Числовой | Справочник «Операции» |
| Код\_кредита | Id\_kredita | Числовой | Справочник «Кредит» |
| Номер\_поручения | Nomer\_poruch | Числовой | Справочник «Платежные поручения» |
| Код\_вклада | Id\_vklada | Числовой | Справочник «Вклады» |

Продолжение таблицы 68

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование поля | Идентификатор | Тип данных | Источник данных |
| Код\_пользователя | Id\_pol | Числовой | Справочник «Пользователи» |
| Номер\_счета | Nomer\_scheta | Числовой | Справочник «Счета» |
| Наименование | Name | Текстовый | Справочник «Операции» |
| Дата | Date | Текстовый | Справочник «Операции» |

Таблица 69 – Выходные данные «Выписка по счету»

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование поля | Идентификатор | Тип данных | Источник данных |
| Номер\_счета | Nomer\_scheta | Числовой | Справочник «Счета» |
| Наименование | Name | Текстовый | Справочник «Операции» |
| Сумма\_кредита | Sum\_kred | Денежный | Справочник «Кредиты» |
| Сумма\_вклада | Sum\_vklad | Денежный | Справочник «Вклады» |
| Сумма\_на\_счете | Summa | Текстовый | Справочник «Счета» |

Таблица 70 – Выходные данные «Оборот по счету за период»

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование поля | Идентификатор | Тип данных | Источник данных |
| Номер\_счета | Nomer\_scheta | Числовой | Справочник «Счета» |
| Наименование | Name | Текстовый | Справочник «Операции» |
| Сумма\_кредита | Sum\_kred | Денежный | Справочник «Кредиты» |
| Сумма\_вклада | Sum\_vklad | Денежный | Справочник «Вклады» |

* + - 1. Сообщения

Сообщения программисту в программе не присутствуют.

### Руководство оператора

* + - 1. Назначение программы

Приложение разработано для автоматизации учета банковских пассивов. Программа предназначена для 2-х типов пользователей: администратор, операционист. Каждый пользователь имеет свои определенные функции.

* + - 1. Условия выполнения программы

Для корректного выполнения данного ПО, аппаратное средство должно соответствовать данным характеристикам:

1. процессор: Pentium 4 с частотой 1 ГГц;
2. оперативная память: 256 МБ;
3. память на жестком диске: минимум 50 МБ свободного места на диске;
4. операционная система: минимум Windows XP.
   * + 1. Выполнение программы

После запуска программы необходимо выполнить вход. Зависимо под какой учетной записью Вы войдете, у Вас будет определенные функции. Ниже будут представлены ограничения по функциям:

* Администратор: ограничения отсутствуют;
* Операционист: отсутствует возможность добавления нового сотрудника.

После авторизации вы попадаете в главное меню, где будут представлены все справочники.

Для того чтобы добавить кредит или вклад следует выполнить следующие действия:

* + - * 1. Нажать кнопку на главном меню «Операции»;
        2. На форме «Операции» нажать кнопку «Добавить»;
        3. Выбрать клиента или добавить нового;
        4. Выбрать вид операции и заполнить поля;
        5. Завершить добавление операции.

Для просмотра информации о клиенте необходимо нажать на главном меню кнопку, после перехода выбрать клиента.

Для просмотра выписки по счету требуется перейти на форму клиенты, навести на вкладку «Отчеты», выбрать «выписка по счету», выбрать дату и клиента, нажать кнопку «Вывести».

Для просмотра оборота по счету требуется перейти на форму клиенты, навести на вкладку «Отчеты», выбрать «оборот по счету», выбрать необходимый период и клиента, нажать кнопку «Вывести».

Для добавления нового сотрудника необходимо зайти как Администратор и на окне «Главное меню» выбрать «Добавить сотрудника», заполнить все необходимые данные и нажать кнопку «Добавить».

Для просмотра выданных кредитов необходимо на главном меню нажать на кнопку «Кредиты».

Для просмотра добавленных вкладов необходимо на главном меню нажать на кнопку «Вклады».

Для выполнения перевода необходимо на главном меню нажать на кнопку «Клиенты», выбрать клиента, который выполняет перевод, затем в меню выбрать «Выполнить перевод», выбрать получателя, ввести сумму перевода выбрать дату и нажать кнопку перевести.

Для добавления вида вклада необходимо на главном меню нажать кнопку «Вклады», затем «Добавить вид», заполнить все поля и нажать кнопку «Добавить».

* + - 1. Сообщения оператору

Сообщения представлены на рисунках 50 – 54.

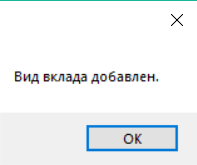


Рис. 50 – Сообщение оператору

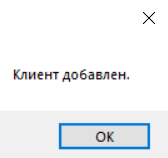


Рис. 51 – Сообщение оператору

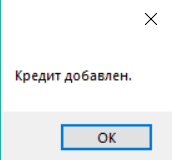


Рис. 52 – Сообщение оператору

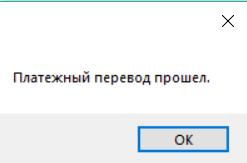


Рис. 53 – Сообщение оператору

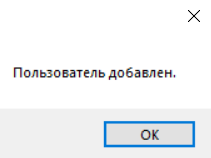


Рис. 54 – Сообщение оператору

# ЗАКЛЮЧЕНИЕ

Данное приложение разработано в учебных целях в рамках курсовой работы по дисциплине «Технология разработки программного обеспечения». Предполагается, что при внедрении приложения в банк сотрудники организации отметят существенное сокращение времени на учет банковских операций и отчетной документации, а также появилась возможность хранить в базе уже занесенную информацию о клиенте, его кредитах и вкладов. Очень удобно выводятся из программы все необходимые документы. Следует отметить, что до внедрения программы оборот по счету и выписка по счету составлялись вручную. Следовательно, представленное прикладное решение может использоваться небольшими банками. При разработке данного прикладного решения учитывались актуальность предметной области, огромное количество организаций, занимающихся учетом банковских пассивов.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. ГОСТ 19.001-77 ЕСПД. Общие положения.
2. ГОСТ 19.005-85 ЕСПД. Р-схемы алгоритмов и программ. Обозначения условные графические и правила выполнения.
3. ГОСТ 19.104-78 ЕСПД. Основные надписи.
4. ГОСТ 19.106-78 ЕСПД. Требования к программным документам, выполненным печатным способом.
5. ГОСТ 19.503-79 ЕСПД. Руководство системного программиста. Требования к содержанию и оформлению.
6. ГОСТ 19.504-79 ЕСПД. Руководство программиста. Требования к содержанию и оформлению.
7. ГОСТ 19.505-79 ЕСПД. Руководство оператора. Требования к содержанию и оформлению.
8. Сайт о программировании [Электронный ресурс] URL: <https://metanit.com>

ПРИЛОЖЕНИЕ 1



Рис. 55 – Интеллектуальная карта

ПРИЛОЖЕНИЕ 2



Рис. 56 – Паспорт пациента

ПРИЛОЖЕНИЕ 3

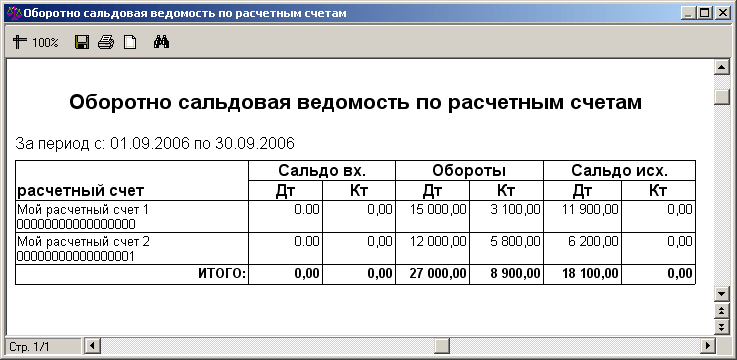


Рис. 57 – Оборот по счету за период



Рис. 58 – Выписка по счету

ПРИЛОЖЕНИЕ 4

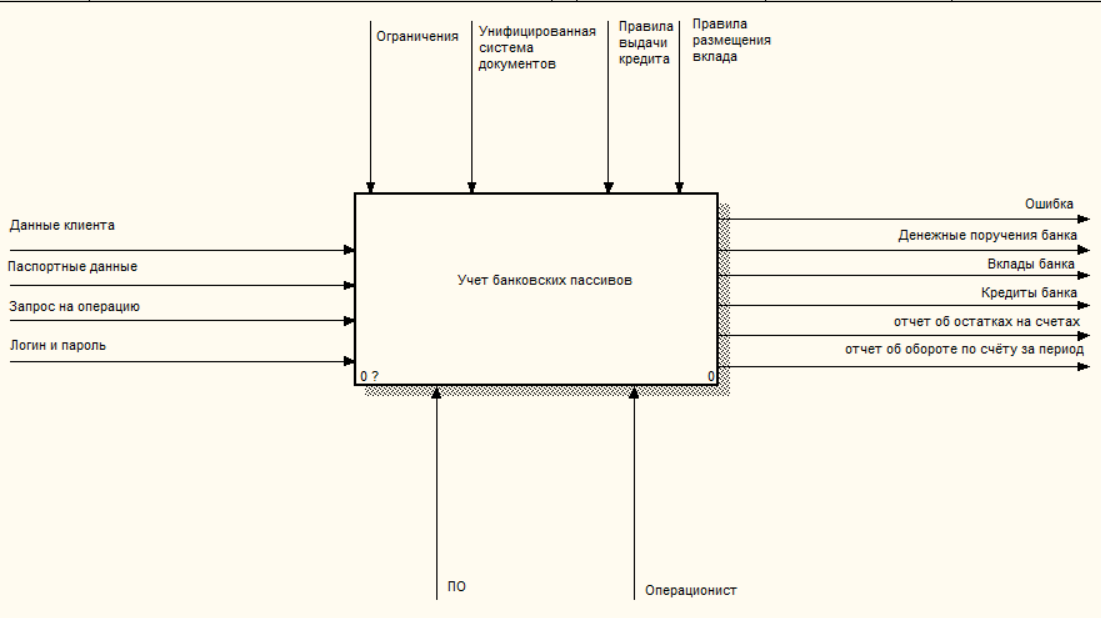


Рис. 59 – Функциональная диаграмма

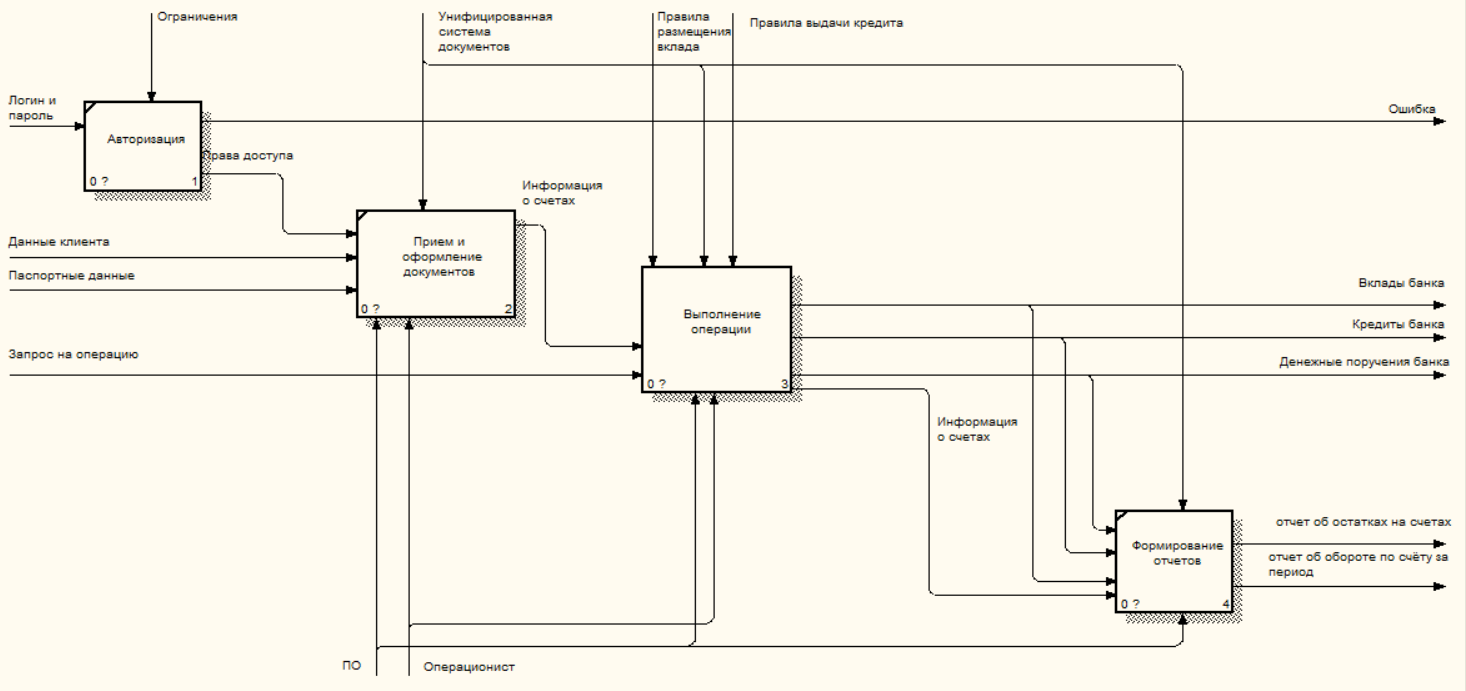


Рис. 60 – Детализация функциональной диаграммы

Продолжение приложения 4

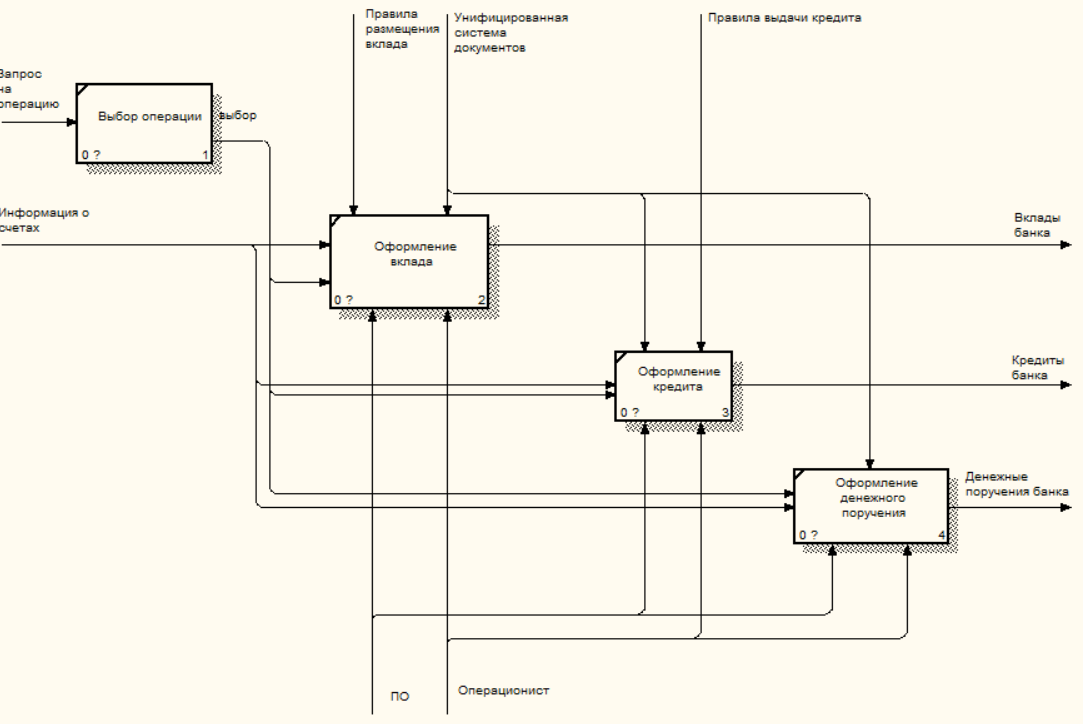


Рис. 61 – Детализация блока «Выполнение операции»

ПРИЛОЖЕНИЕ 5

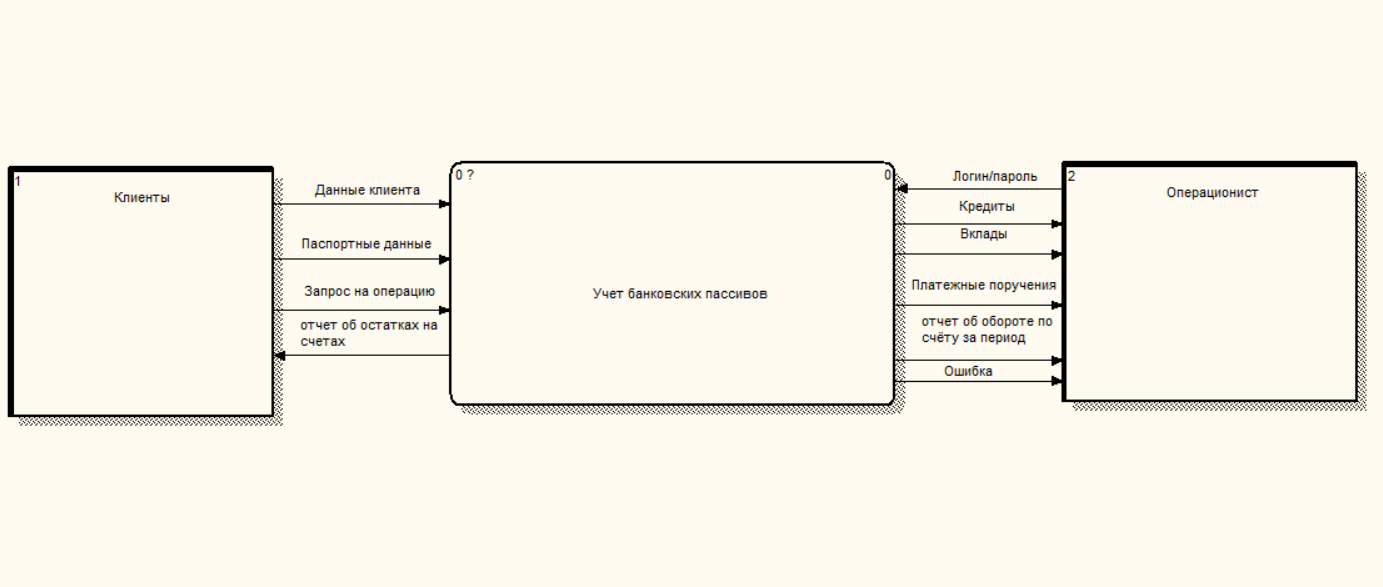


Рис. 62 – Диаграмма поток данных

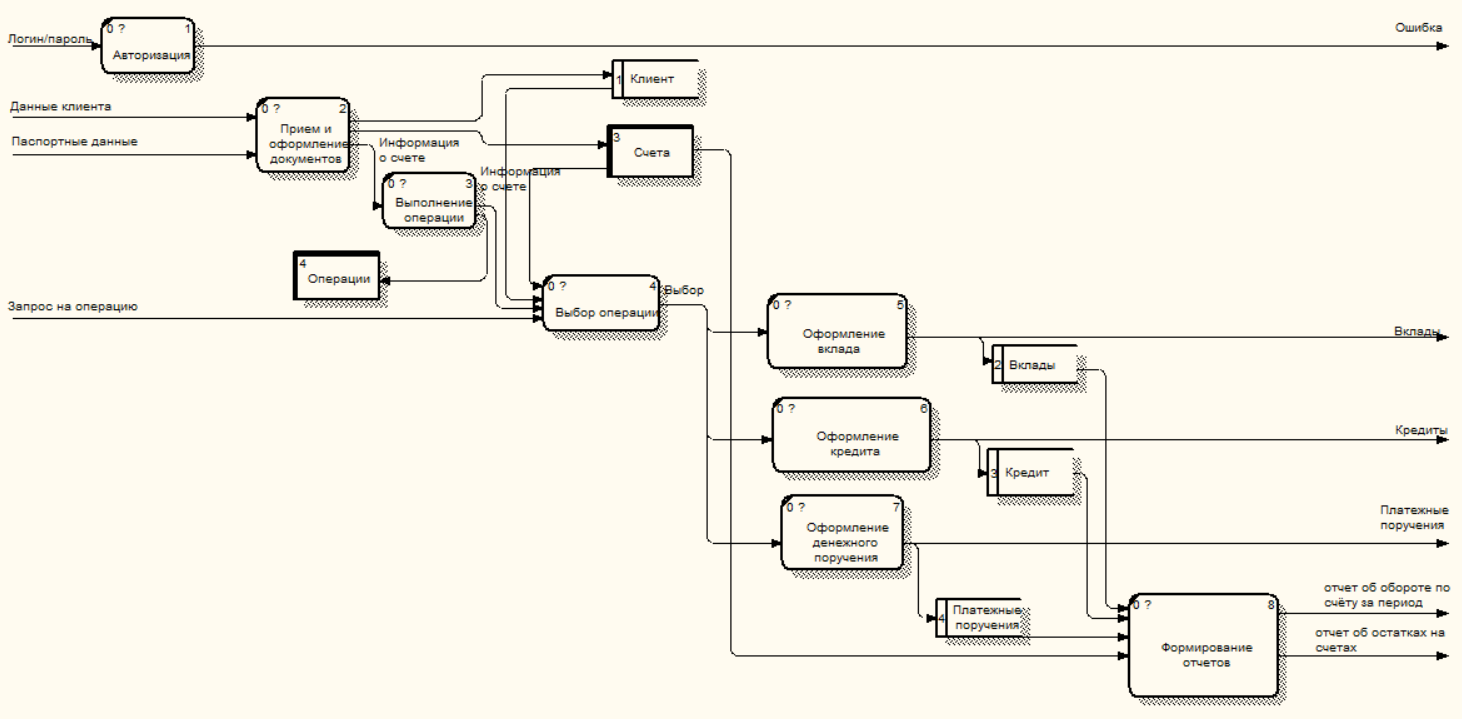


Рис. 63 – Детализации диаграммы потоков данных

ПРИЛОЖЕНИЕ 6

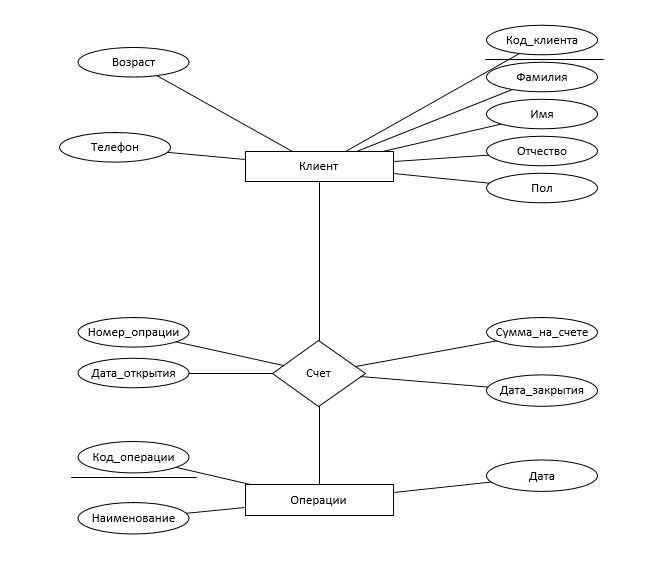


Рис. 64 – Концептуальная модель

ПРИЛОЖЕНИЕ 7

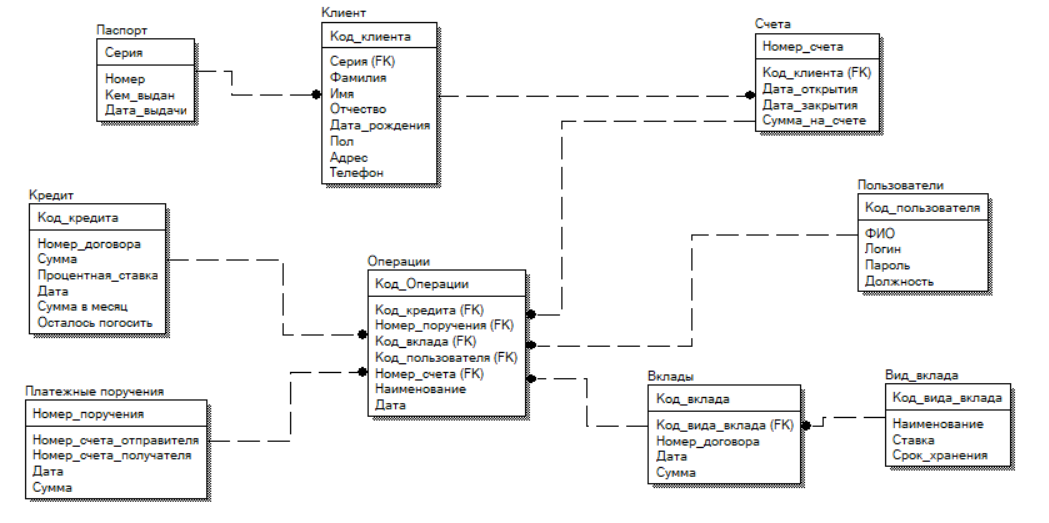


Рис. 65 – Логическая модель

ПРИЛОЖЕНИЕ 8

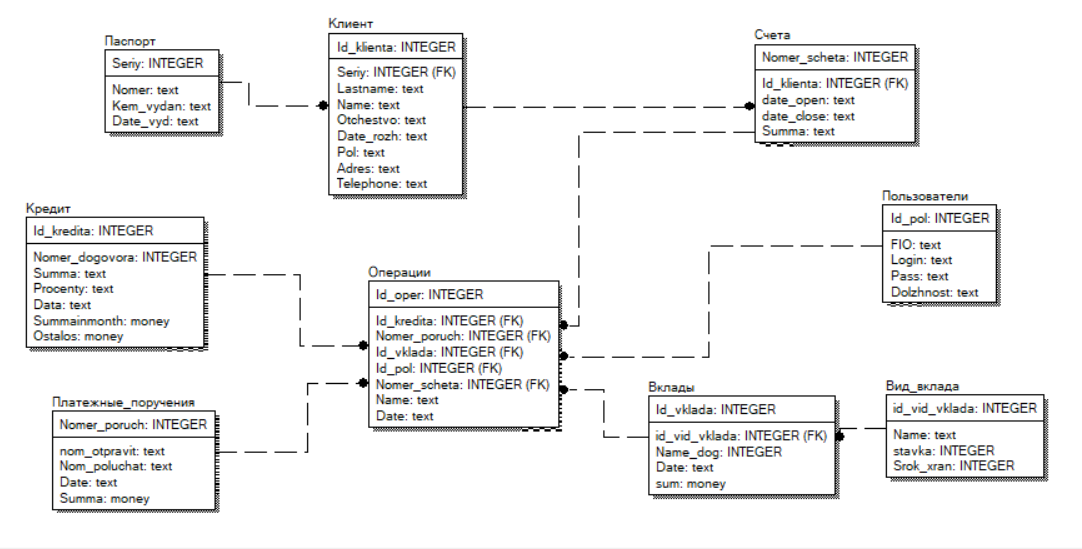


Рис. 66 – Физическая модель

ПРИЛОЖЕНИЕ 9

Листинг SQL-сценариев представлен на рисунках 67 – 68.

|  |
| --- |
| create table Паспорт (Серия Text not null primary key, Номер integer not null, Кем\_выдан TEXT not null, Дата\_выдачи Text not null)  create table Клиент (Код\_клиента integer primary key AUTOINCREMENT not null, Фамилия TEXT not null, Имя TEXT not null, Отчество TEXT not null, Дата\_рождения Text not null, Пол TEXT check (Пол in ('Муж', 'Жен')) not null, Серия\_паспорта Text not null, Адрес TEXT not null, Телефон TEXT not null, foreign key (Серия\_паспорта) references Паспорт(Серия))  create table Счета (Номер\_счета integer primary key AUTOINCREMENT not null, Код\_клиента integer, Дата\_открытия Text not null, Дата\_закрытия Text, Сумма\_на\_счете money not null, foreign key (Код\_клиента) references Клиент(Код\_клиента))  create table Кредит (Код\_кредита integer primary key AUTOINCREMENT not null, Номер\_договора integer not null, Сумма money not null, Процентная\_ставка Text not null, Дата\_кредитования Text not null, Сумма\_в\_месяц money not null, Осталось\_погасить money not null)  create table Вид\_вклада (Код\_вида\_вклада integer primary key AUTOINCREMENT not null, Наименование Text not null, Cтавка integer not null, Срок\_хранения Text not null)  create table Вклады (Код\_вклада integer primary key AUTOINCREMENT not null, Код\_вида\_вклада integer, Номер\_договора integer not null, Дата Text not null, Сумма money not null, foreign key (Код\_вида\_вклада) references Вид\_вклада(Код\_вида\_вклада))  create table Платежные\_поручения (Номер\_поручения integer primary key AUTOINCREMENT not null, Номер\_счета\_отправителя integer not null, Номер\_счета\_получателя integer not null, Дата Text not null, Сумма money not null)  create table Пользователи (Код\_пользователя integer primary key AUTOINCREMENT not null, ФИО Text not null, Логин Text not null, Пароль Text not null, Должность Text not null)  create table Операции(Код\_операции integer primary key AUTOINCREMENT not null,Наименование Text not null,Код\_вклада integer,Код\_кредитаinteger,Код\_поручения integer,Номер\_счета integer, Код\_сотрудника integer,Дата Text not null, foreign key (Код\_вклада) references Вклады(Код\_вклада),foreign key (Код\_кредита) references Кредит(Код\_кредита),foreign key (Код\_поручения) references Платежные\_поручения(Номер\_поручения),foreign key (Номер\_счета) references Счета(Номер\_счета),foreign key (Код\_сотрудника) references Пользователи(Код\_пользователя)) |

Рис. 67 – Листинг SQL-сценариев

Продолжение приложения 9

INSERT INTO Пользователи (ФИО,Логин,Пароль,Должность) values ('Иванов Иван Иванович','admin','admin','Администратор')

Рис. 68 – Листинг SQL-сценариев

ПРИЛОЖЕНИЕ 10

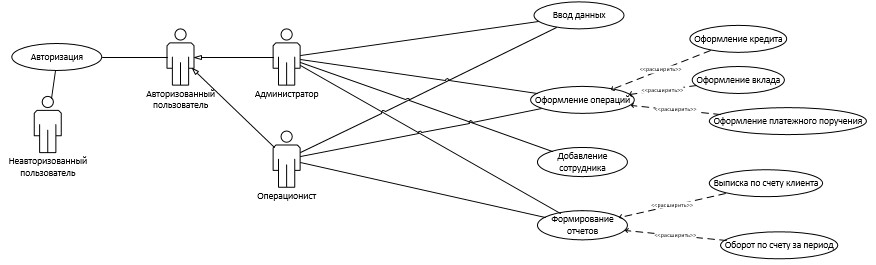


Рис. 69 – Диаграмма вариантов использования

ПРИЛОЖЕНИЕ 11

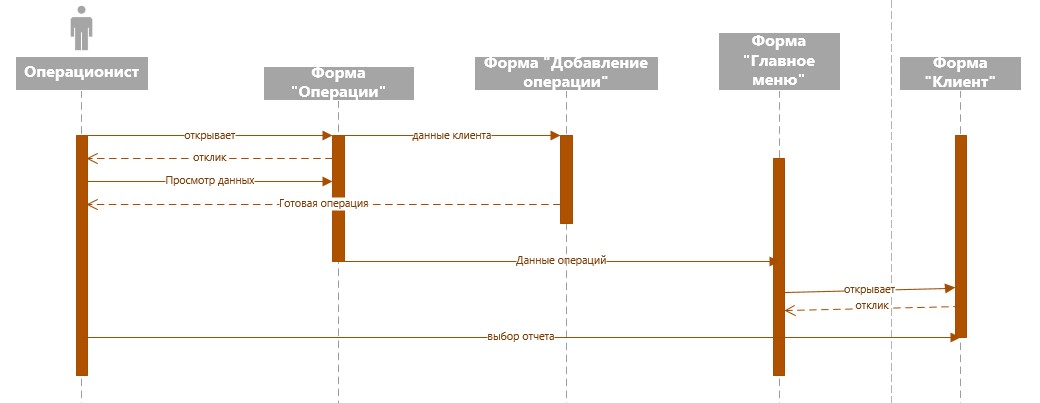


Рис. 70 – Диаграмма последовательности

ПРИЛОЖЕНИЕ 12

Листинг модуля «Program.cs» представлен на рисунке 71.

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Bank

{

static class Program

{

/// <summary>

/// Главная точка входа для приложения.

/// </summary>

[STAThread]

static void Main()

{

Application.EnableVisualStyles();

Application.SetCompatibleTextRenderingDefault(false);

Application.Run(new FRMautorization());

}

}

}

Рисунок 71 - Листинг модуля «Program.cs»

Листинг модуля «Autorization.cs» представлен на рисунках 72-74.

using System;

using System.Data;

using System.Windows.Forms;

using System.Data.SQLite;

using System.IO;

namespace Bank

{

public partial class FRMautorization : Form

{

public static string kod;

private String dbFileName;

private SQLiteConnection m\_dbConn;

private SQLiteCommand m\_sqlCmd;

public FRMautorization()

{

InitializeComponent();

}

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

{

m\_dbConn = new SQLiteConnection();

m\_sqlCmd = new SQLiteCommand();

dbFileName = "bank.sqlite";

if (!File.Exists(dbFileName))

{

SQLiteConnection.CreateFile(dbFileName);

Рисунок 72 - Листинг модуля «Autorization.cs»

Продолжение приложение 12

try

{

m\_dbConn = new SQLiteConnection("Data Source=" + dbFileName + ";Version=3;");

m\_dbConn.Open();

m\_sqlCmd.Connection = m\_dbConn;

m\_sqlCmd.CommandText = "create table Паспорт (Серия Text not null primary key, Номер integer not null, Кем\_выдан TEXT not null,Дата\_выдачи Text not null)";

m\_sqlCmd.ExecuteNonQuery();

m\_sqlCmd.CommandText = "create table Клиент (Код\_клиента integer primary key AUTOINCREMENT not null, Фамилия TEXT not null, Имя TEXT not null, Отчество TEXT not null, Дата\_рождения Text not null, Пол TEXT check (Пол in ('Муж', 'Жен')) not null, Серия\_паспорта Text not null, Адрес TEXT not null, Телефон TEXT not null, foreign key (Серия\_паспорта) references Паспорт(Серия))";

m\_sqlCmd.ExecuteNonQuery();

m\_sqlCmd.CommandText = "create table Счета (Номер\_счета integer primary key AUTOINCREMENT not null, Код\_клиента integer, Дата\_открытия Text not null, Дата\_закрытия Text, Сумма\_на\_счете money not null, foreign key (Код\_клиента) references Клиент(Код\_клиента))";

m\_sqlCmd.ExecuteNonQuery();

m\_sqlCmd.CommandText = "create table Кредит (Код\_кредита integer primary key AUTOINCREMENT not null, Номер\_договора integer not null, Сумма money not null, Процентная\_ставка Text not null, Дата\_кредитования Text not null, Сумма\_в\_месяц money not null, Осталось\_погасить money not null)";

m\_sqlCmd.ExecuteNonQuery();

m\_sqlCmd.CommandText = "create table Вид\_вклада (Код\_вида\_вклада integer primary key AUTOINCREMENT not null, Наименование Text not null, Cтавка integer not null, Срок\_хранения Text not null)";

m\_sqlCmd.ExecuteNonQuery();

m\_sqlCmd.CommandText = "create table Вклады (Код\_вклада integer primary key AUTOINCREMENT not null, Код\_вида\_вклада integer, Номер\_договора integer not null, Дата Text not null, Сумма money not null,foreign key (Код\_вида\_вклада) references Вид\_вклада(Код\_вида\_вклада))";

m\_sqlCmd.ExecuteNonQuery();

m\_sqlCmd.CommandText = "create table Платежные\_поручения (Номер\_поручения integer primary key AUTOINCREMENT not null, Номер\_счета\_отправителя integer not null, Номер\_счета\_получателя integer not null, Дата Text not null, Сумма money not null)";

m\_sqlCmd.ExecuteNonQuery();

m\_sqlCmd.CommandText = "create table Пользователи (Код\_пользователя integer primary key AUTOINCREMENT not null, ФИО Text not null, Логин Text not null, Пароль Text not null, Должность Text not null)";

m\_sqlCmd.ExecuteNonQuery();

m\_sqlCmd.CommandText = "create table Операции(Код\_операции integer primary key AUTOINCREMENT not null,Наименование Text not null,Код\_вклада integer,Код\_кредита integer,Код\_поручения integer,Номер\_счета integer, Код\_сотрудника integer,Дата Text not null,foreign key (Код\_вклада) references Вклады(Код\_вклада),foreign key (Код\_кредита) references Кредит(Код\_кредита),foreign key (Код\_поручения) references Платежные\_поручения(Номер\_поручения),foreign key (Номер\_счета) references Счета(Номер\_счета),foreign key (Код\_сотрудника) references Пользователи(Код\_пользователя))";

m\_sqlCmd.ExecuteNonQuery();

Рисунок 73 - Листинг модуля «Autorization.cs»

Продолжение приложение 12

m\_sqlCmd.CommandText = "INSERT INTO Пользователи (ФИО,Логин,Пароль,Должность) values ('Иванов Иван Иванович','admin','admin','Администратор')";

m\_sqlCmd.ExecuteNonQuery();

MessageBox.Show("Успешно созданы ");

}

catch (SQLiteException ex)

**{**

MessageBox.Show("Error: " + ex.Message);

}

m\_dbConn.Close();

} }

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

{

m\_dbConn = new SQLiteConnection();

DataTable dTable = new DataTable();

String sqlQuery;

dbFileName = "bank.sqlite";

if (m\_dbConn.State != ConnectionState.Open)

{

m\_dbConn = new SQLiteConnection("Data Source=" + dbFileName + ";Version=3;");

m\_dbConn.Open();

m\_sqlCmd.Connection = m\_dbConn;

}

try

{

sqlQuery = "SELECT \* FROM Пользователи where Логин='"+ TXTLogin.Text + "' and Пароль='" + TXTPass.Text + "'";

SQLiteDataAdapter adapter = new SQLiteDataAdapter(sqlQuery, m\_dbConn);

adapter.Fill(dTable);

if (dTable.Rows.Count > 0)

{

kod= dTable.Rows[0][0].ToString();

Properties.Settings.Default.Role = dTable.Rows[0][4].ToString();

this.Hide();

FrmGlavnoemenu glavnoemenu = new FrmGlavnoemenu();

glavnoemenu.FIO = dTable.Rows[0][1].ToString();

glavnoemenu.ShowDialog();

}

else

MessageBox.Show("Неправильно введен логин/пароль.");

}

catch (SQLiteException ex)

{

MessageBox.Show("Error: " + ex.Message);

}

finally

{

m\_dbConn.Close();

}

}

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

{

Application.Exit();}

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

{

Application.Exit();}}}

Рисунок 74 - Листинг модуля «Autorization.cs»

Продолжение приложение 12

Листинг модуля «Vklady.cs» представлен на рисунках 75-76.

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.Data.SQLite;

using System.IO;

namespace Bank

{

public partial class Frmvklady : Form

{

private String dbFileName;

private SQLiteConnection m\_dbConn;

public Frmvklady()

{

InitializeComponent();

}

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

{

FrmAddvydvklada addvyd = new FrmAddvydvklada();

addvyd.ShowDialog();

}

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

{

m\_dbConn = new SQLiteConnection();

DataTable dTable = new DataTable();

String sqlQuery;

dbFileName = "bank.sqlite";

m\_dbConn = new SQLiteConnection("Data Source=" + dbFileName + ";Version=3;");

m\_dbConn.Open();

try

{

sqlQuery = "SELECT Вклады.Номер\_договора, Вид\_вклада.Наименование, Вид\_вклада.Cтавка,Вклады.Дата,Вид\_вклада.Срок\_хранения,Вклады.Сумма FROM Вклады inner join Вид\_вклада on Вид\_вклада.Код\_вида\_вклада=Вклады.Код\_вида\_вклада";

SQLiteDataAdapter adapter = new SQLiteDataAdapter(sqlQuery, m\_dbConn);

adapter.Fill(dTable);

Dtvklady.DataSource = dTable;

}

catch (SQLiteException ex)

{

MessageBox.Show("Error: " + ex.Message);

}

finally

{

m\_dbConn.Close();

}

}

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

Рисунок 75 - Листинг модуля «Vklady.cs»

Продолжение приложение 12

{

m\_dbConn = new SQLiteConnection();

DataTable dTable = new DataTable();

String sqlQuery;

dbFileName = "bank.sqlite";

m\_dbConn = new SQLiteConnection("Data Source=" + dbFileName + ";Version=3;");

m\_dbConn.Open();

try

{

sqlQuery = $"SELECT \* FROM Вклады where Дата BETWEEN '{DtpOt.Value.ToString("dd.MM.yyyy")}' and '{DtpDo.Value.ToString("dd.MM.yyyy")}'";

SQLiteDataAdapter adapter = new SQLiteDataAdapter(sqlQuery, m\_dbConn);

adapter.Fill(dTable);

Dtvklady.DataSource = dTable;

}

catch (SQLiteException ex)

{

MessageBox.Show("Error: " + ex.Message);

}

finally

{

m\_dbConn.Close();

}

}

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

{

m\_dbConn = new SQLiteConnection();

DataTable dTable = new DataTable();

String sqlQuery;

dbFileName = "bank.sqlite";

m\_dbConn = new SQLiteConnection("Data Source=" + dbFileName + ";Version=3;");

m\_dbConn.Open();

try

{

sqlQuery = "SELECT Вклады.Номер\_договора, Вид\_вклада.Наименование, Вид\_вклада.Cтавка,Вклады.Дата,Вид\_вклада.Срок\_хранения,Вклады.Сумма FROM Вклады inner join Вид\_вклада on Вид\_вклада.Код\_вида\_вклада=Вклады.Код\_вида\_вклада";

SQLiteDataAdapter adapter = new SQLiteDataAdapter(sqlQuery, m\_dbConn);

adapter.Fill(dTable);

Dtvklady.DataSource = dTable;

}

catch (SQLiteException ex)

{

MessageBox.Show("Error: " + ex.Message);

}

finally

{

m\_dbConn.Close();

}

}

}

}

Рисунок 76 - Листинг модуля «Vklady.cs»

Продолжение приложение 12

Листинг модуля «Perevod.cs» представлен на рисунках 77-80.

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.Data.SQLite;

using System.IO;

namespace Bank

{

public partial class FrmPerevod : Form

{

private String dbFileName;

private SQLiteConnection m\_dbConn;

private SQLiteCommand m\_sqlCmd;

public FrmPerevod()

{

InitializeComponent();

}

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

{

m\_dbConn = new SQLiteConnection();

m\_sqlCmd = new SQLiteCommand();

string sqlQuery;

string sqlQuery1;

string sqlQuery2;

string sqlQuery3;

DataTable dTable = new DataTable();

DataTable dTable1 = new DataTable();

DataTable dTable2 = new DataTable();

DataTable dTable3 = new DataTable();

DataTable dTable4 = new DataTable();

DataTable dTable5 = new DataTable();

dbFileName = "bank.sqlite";

m\_dbConn = new SQLiteConnection("Data Source=" + dbFileName + ";Version=3;");

m\_dbConn.Open();

m\_sqlCmd.Connection = m\_dbConn;

//Добавление перевода

if (TxtSumperev.Text == "" | Cbnompol.SelectedIndex ==-1)

{

MessageBox.Show("Заполните все поля");

m\_dbConn.Close();

return;

}

else

{

sqlQuery = "SELECT Код\_клиента FROM Счета where (Сумма\_на\_счете > " + TxtSumperev.Text + " and Код\_клиента='"+ FrmKlient.kod\_klienta + "')";

Рисунок 77 - Листинг модуля «Perevod.cs»

Продолжение приложение 12

SQLiteDataAdapter adapter = new SQLiteDataAdapter(sqlQuery, m\_dbConn);

adapter.Fill(dTable1);

if (dTable1.Rows.Count > 0)

{

try

{

sqlQuery1 = "SELECT Счета.Номер\_счета,Счета.Сумма\_на\_счете FROM Счета inner join Клиент on Клиент.Код\_клиента=Счета.Код\_клиента where Код\_клиента='" + FrmKlient.kod\_klienta + "'";

SQLiteDataAdapter adapter2 = new SQLiteDataAdapter(sqlQuery, m\_dbConn);

adapter2.Fill(dTable2);

m\_dbConn.Close();

m\_dbConn.Open();

m\_sqlCmd.CommandText = $"INSERT INTO Платежные\_поручения values (null,'{dTable2.Rows[0][0].ToString()}','{Cbnompol.SelectedValue}','{DTperev.Value.ToString("dd.MM.yyyy")}','{TxtSumperev.Text}')";

m\_sqlCmd.ExecuteNonQuery();

m\_dbConn.Close();

m\_dbConn.Open();

sqlQuery2 = $"SELECT Номер\_поручения FROM Платежные\_поручения where Номер\_счета\_отправителя='{dTable2.Rows[0][0].ToString()}' and Номер\_счета\_получателя='{Cbnompol.SelectedValue}' and Дата='{DTperev.Value.ToString("dd.MM.yyyy")}' and Сумма='{TxtSumperev.Text}'";

SQLiteDataAdapter adapter3 = new SQLiteDataAdapter(sqlQuery, m\_dbConn);

adapter3.Fill(dTable3);

m\_dbConn.Close();

m\_dbConn.Open();

m\_sqlCmd.CommandText = $"INSERT INTO Операции values (null,'Платежное поручение',null,null,'{dTable3.Rows[0][0].ToString()}','{dTable2.Rows[0][0].ToString()}','{FrmKlient.kod\_klienta}','{DTperev.Value.ToString("dd.MM.yyyy")}')";

m\_sqlCmd.ExecuteNonQuery();

m\_dbConn.Close();

m\_dbConn.Open();

sqlQuery1 = "SELECT Сумма\_на\_счете FROM Счета inner join Клиент on Клиент.Код\_клиента=Счета.Код\_клиента where Код\_клиента='" + Cbnompol.SelectedValue + "'";

SQLiteDataAdapter adapter5 = new SQLiteDataAdapter(sqlQuery, m\_dbConn);

adapter5.Fill(dTable5);

m\_dbConn.Close();

m\_dbConn.Open();

m\_sqlCmd.CommandText = $"Update Счета set Сумма\_на\_счете=Сумма\_на\_счете + {TxtSumperev.Text} where Код\_клиента='{Cbnompol.SelectedValue}'";

m\_sqlCmd.ExecuteNonQuery();

m\_dbConn.Close();

m\_dbConn.Open();

sqlQuery3 = "SELECT Счета.Сумма\_на\_счете FROM Счета inner join Клиент on Клиент.Код\_клиента=Счета.Код\_клиента where Код\_клиента='" + FrmKlient.kod\_klienta + "'";

Рисунок 78 - Листинг модуля «Perevod.cs»

Продолжение приложение 12

SQLiteDataAdapter adapter4 = new SQLiteDataAdapter(sqlQuery, m\_dbConn);

adapter4.Fill(dTable4);

m\_dbConn.Close();

m\_dbConn.Open();

m\_sqlCmd.CommandText = $"Update Счета set Сумма\_на\_счете=Сумма\_на\_счете - {TxtSumperev.Text} where Номер\_счета='{dTable2.Rows[0][0].ToString()}'";

m\_sqlCmd.ExecuteNonQuery();

m\_dbConn.Close();

MessageBox.Show("Платежный перевод прошел.");

m\_dbConn.Close();

}

catch (SQLiteException ex)

{

MessageBox.Show("Error: " + ex.Message);

}

finally

{

m\_dbConn.Close();

}

}

else

{

MessageBox.Show("Недостаточно средств.");

}

}

}

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

{

m\_dbConn = new SQLiteConnection();

DataTable dTable = new DataTable();

String sqlQuery;

dbFileName = "bank.sqlite";

m\_dbConn = new SQLiteConnection("Data Source=" + dbFileName + ";Version=3;");

m\_dbConn.Open();

sqlQuery = "Select \* from Счета";

SQLiteDataAdapter adapter = new SQLiteDataAdapter(sqlQuery, m\_dbConn);

adapter.Fill(dTable);

Cbnompol.DataSource = dTable;

Cbnompol.ValueMember = "Номер\_счета";

Cbnompol.DisplayMember = "Номер\_счета";

if (dTable.Rows.Count > 0)

{

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

{

Cbnompol.SelectedIndex = i;

}

}

m\_dbConn.Close();

}

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

{

Рисунок 79 - Листинг модуля «Perevod.cs»

Продолжение приложение 12

this.Close();

}

private void TxtSumperev\_KeyPress(object sender, KeyPressEventArgs e)

{

char text = e.KeyChar;

if (!Char.IsDigit(text) && text != 8)

{

e.Handled = true;

}

}

}

}

Рисунок 80 - Листинг модуля «Perevod.cs»

Листинг модуля «otchet.cs» представлен на рисунках 81-83.

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.Data.SQLite;

using System.IO;

using Word = Microsoft.Office.Interop.Word;

namespace Bank

{

public partial class FrmVypotchet : Form

{

private String dbFileName;

private SQLiteConnection m\_dbConn;

public FrmVypotchet()

{

InitializeComponent();

}

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

{

Close();

}

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

{

Word.Document wDoc = null;

try

{

dbFileName = "bank.sqlite";

m\_dbConn = new SQLiteConnection("Data Source=" + dbFileName + ";Version=3;");

DataTable dTable = new DataTable();

String sqlQuery;

Рисунок 81 - Листинг модуля «otchet.cs»

Продолжение приложение 12

Word.Application wApp = new Word.Application();

wApp.Visible = true;

string source = $@"{Directory.GetCurrentDirectory().ToString()}\Выписка по счету.docx";

wDoc = wApp.Documents.Open(source);

wDoc.Activate();

Word.Bookmarks wMarks = wDoc.Bookmarks;

m\_dbConn.Open();

sqlQuery = $"SELECT Операции.Код\_операции,Операции.Наименование,Операции.Номер\_счета,Вклады.Сумма,Кредит.Сумма,Счета.Сумма\_на\_счете FROM Операции inner join Вклады on Вклады.Код\_вклада=Операции.Код\_вклада inner join Счета on Счета.Номер\_счета=Операции.Номер\_счета inner join Кредит on Кредит.Код\_кредита=Операции.Код\_кредита inner join Клиент on Клиент.Код\_клиента=Счета.Код\_клиента where Операции.Дата='{DtDate.Value.ToString("dd.MM.yyyy")}' and Клиент.Код\_клиента={Cbklient.SelectedValue}";

using (SQLiteCommand cmd = new SQLiteCommand(sqlQuery, m\_dbConn))

{

using (SQLiteDataReader reader = cmd.ExecuteReader())

{

if (reader.HasRows)

{

reader.Read();

{

wMarks["nomer"].Range.Text = reader.GetValue(2).ToString();

wMarks["date"].Range.Text = DateTime.Now.ToString("dd MMMM yyyy");

wMarks["number"].Range.Text = reader.GetValue(0).ToString();

wMarks["operation"].Range.Text = reader.GetValue(1).ToString();

wMarks["schet"].Range.Text = reader.GetValue(2).ToString();

if (reader.GetValue(3).ToString() == "")

wMarks["debet"].Range.Text = reader.GetValue(3).ToString();

else wMarks["debet"].Range.Text = "0";

if (reader.GetValue(4).ToString() == "")

wMarks["kredit"].Range.Text = reader.GetValue(4).ToString();

else wMarks["kredit"].Range.Text = "0";

wMarks["Itogo"].Range.Text = reader.GetValue(5).ToString();

}

}

else

{

sqlQuery = $"SELECT Счета.Номер\_счета,Счета.Сумма\_на\_счете from Счета inner join Клиент on Клиент.Код\_клиента=Счета.Код\_клиента where Клиент.Код\_клиента={Cbklient.SelectedValue}";

SQLiteDataAdapter adapter = new SQLiteDataAdapter(sqlQuery, m\_dbConn);

adapter.Fill(dTable);

if (dTable.Rows.Count > 0)

Рисунок 82 - Листинг модуля «otchet.cs»

Продолжение приложение 12

{

wMarks["nomer"].Range.Text = dTable.Rows[0][0].ToString();

wMarks["date"].Range.Text = DateTime.Now.ToString("dd MMMM yyyy");

wMarks["number"].Range.Text = "1";

wMarks["operation"].Range.Text = "-";

wMarks["schet"].Range.Text = dTable.Rows[0][0].ToString();

wMarks["debet"].Range.Text = "-";

wMarks["kredit"].Range.Text = "-";

wMarks["Itogo"].Range.Text = dTable.Rows[0][1].ToString();

}

}

}

}

m\_dbConn.Close();

wDoc.SaveAs2($@"{Directory.GetCurrentDirectory().ToString()}\1.docx");

wDoc.Close();

wDoc = null;

wApp.Quit();

}

catch (Exception ex)

{

MessageBox.Show(ex.ToString());

wDoc.Close();

wDoc = null;

}

}

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

{

dbFileName = "bank.sqlite";

m\_dbConn = new SQLiteConnection("Data Source=" + dbFileName + ";Version=3;");

DataTable dTable1 = new DataTable();

String sqlQuery;

m\_dbConn.Open();

sqlQuery = "Select \* from Клиент";

SQLiteDataAdapter adapter1 = new SQLiteDataAdapter(sqlQuery, m\_dbConn);

adapter1.Fill(dTable1);

Cbklient.DataSource = dTable1;

Cbklient.ValueMember = "Код\_клиента";

Cbklient.DisplayMember = "Фамилия";

if (dTable1.Rows.Count > 0)

{

for (int i = 0; i < dTable1.Rows.Count; i++)

{

Cbklient.SelectedIndex = i;

}

}

m\_dbConn.Close();

}

}

}

Рисунок 83 - Листинг модуля «otchet.cs»

Продолжение приложение 12

Листинг модуля «Operacii.cs» представлен на рисунках 84-86.

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.Data.SQLite;

using System.IO;

namespace Bank

{

public partial class FrmOperacii : Form

{

private String dbFileName;

private SQLiteConnection m\_dbConn;

private void view()

{

m\_dbConn = new SQLiteConnection();

DataTable dTable = new DataTable();

String sqlQuery;

dbFileName = "bank.sqlite";

m\_dbConn = new SQLiteConnection("Data Source=" + dbFileName + ";Version=3;");

m\_dbConn.Open();

try

{

sqlQuery = "SELECT \* FROM Операции";

SQLiteDataAdapter adapter = new SQLiteDataAdapter(sqlQuery, m\_dbConn);

adapter.Fill(dTable);

DtOper.DataSource = dTable;

}

catch (SQLiteException ex)

{

MessageBox.Show("Error: " + ex.Message);

}

finally

{

m\_dbConn.Close();

}

}

public FrmOperacii()

{

InitializeComponent();

}

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

{

this.Close();

}

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

Рисунок 84 - Листинг модуля «Operacii.cs»

Продолжение приложение 12

{

view();

}

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

{

m\_dbConn = new SQLiteConnection();

DataTable dTable = new DataTable();

String sqlQuery;

dbFileName = "bank.sqlite";

m\_dbConn = new SQLiteConnection("Data Source=" + dbFileName + ";Version=3;");

m\_dbConn.Open();

try

{

sqlQuery = "SELECT \* FROM Вид\_вклада";

SQLiteDataAdapter adapter = new SQLiteDataAdapter(sqlQuery, m\_dbConn);

adapter.Fill(dTable);

if (dTable.Rows.Count > 0)

{

Frmaddoper addoper = new Frmaddoper();

addoper.ShowDialog();

}

else

{

MessageBox.Show("Добавьте вклад.");

FrmAddvydvklada vyd = new FrmAddvydvklada();

vyd.ShowDialog();

}

}

catch (SQLiteException ex)

{

MessageBox.Show("Error: " + ex.Message);

}

finally

{

m\_dbConn.Close();

}

}

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

{

view();

}

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

{

m\_dbConn = new SQLiteConnection();

DataTable dTable = new DataTable();

String sqlQuery;

dbFileName = "bank.sqlite";

m\_dbConn = new SQLiteConnection("Data Source=" + dbFileName + ";Version=3;");

m\_dbConn.Open();

try

{

sqlQuery = $"SELECT \* FROM Операции where Дата BETWEEN '{DtpOt.Value.ToString("dd.MM.yyyy")}' and '{DtpDo.Value.ToString("dd.MM.yyyy")}'";

Рисунок 85 - Листинг модуля «Operacii.cs»

Продолжение приложение 12

SQLiteDataAdapter adapter = new SQLiteDataAdapter(sqlQuery, m\_dbConn);

adapter.Fill(dTable);

DtOper.DataSource = dTable;

}

catch (SQLiteException ex)

{

MessageBox.Show("Error: " + ex.Message);

}

finally

{

m\_dbConn.Close();

}

}

}

}

Рисунок 86 - Листинг модуля «Operacii.cs»

Листинг модуля «Obototchet.cs» представлен на рисунках 87-89.

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.Data.SQLite;

using System.IO;

using Word = Microsoft.Office.Interop.Word;

namespace Bank

{

public partial class FrmObototchet : Form

{

private String dbFileName;

private SQLiteConnection m\_dbConn;

public FrmObototchet()

{

InitializeComponent();

}

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

{

Close();

}

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

{

Word.Document wDoc = null;

try

{

dbFileName = "bank.sqlite";

m\_dbConn = new SQLiteConnection("Data Source=" + dbFileName + ";Version=3;");

DataTable dTable = new DataTable();

Рисунок 87 - Листинг модуля «Obototchet.cs»

Продолжение приложение 12

Word.Application wApp = new Word.Application();

wApp.Visible = true;

string source = $@"{ Directory.GetCurrentDirectory().ToString()}\Оборот по счету за период.docx";

wDoc = wApp.Documents.Open(source);

wDoc.Activate();

Word.Bookmarks wMarks = wDoc.Bookmarks;

Word.Range wRange;

using (SQLiteCommand cmd = new SQLiteCommand($"SELECT Операции.Наименование,Операции.Номер\_счета,Вклады.Сумма,Кредит.Сумма FROM Операции inner join Вклады on Вклады.Код\_вклада=Операции.Код\_вклада inner join Счета on Счета.Номер\_счета=Операции.Номер\_счета inner join Кредит on Кредит.Код\_кредита=Операции.Код\_кредита inner join Клиент on Клиент.Код\_клиента=Счета.Код\_клиента where (Операции.Дата BETWEEN '{Dtot.Value.ToString("dd.MM.yyyy")}' and '{DtDo.Value.ToString("dd.MM.yyyy")}') and Клиент.Код\_клиента={Cbklient.SelectedValue}", m\_dbConn))

{

m\_dbConn.Open();

SQLiteDataAdapter da = new SQLiteDataAdapter(cmd);

DataTable dt = new DataTable();

cmd.ExecuteNonQuery();

da.Fill(dt);

wMarks["dateot"].Range.Text = Dtot.Value.ToString("dd.MM.yyyy");

wMarks["datedo"].Range.Text = DtDo.Value.ToString("dd.MM.yyyy");

m\_dbConn.Close();

wRange = wMarks["tab"].Range;

var table = wDoc.Tables.Add(wRange, dt.Rows.Count + 1, dt.Columns.Count, 4);

table.Cell(1, 1).Range.Text = "Операция";

table.Cell(1, 2).Range.Text = "Номер счета";

table.Cell(1, 3).Range.Text = "Дебет";

table.Cell(1, 4).Range.Text = "Кредит";

for (int j = 1; j < dt.Rows.Count + 1; j++)

for (int k = 0; k < dt.Columns.Count; k++)

{

table.Cell(j + 1, k).Range.Text = dt.Rows[j - 1][k].ToString();

}

wDoc.SaveAs2($@"{Directory.GetCurrentDirectory().ToString()}\2.docx");

wDoc.Close();

wDoc = null;

wApp.Quit();

}

}

catch (Exception ex)

{

MessageBox.Show(ex.ToString());

wDoc.Close();

wDoc = null;

}

}

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

{

dbFileName = "bank.sqlite";

m\_dbConn = new SQLiteConnection("Data Source=" + dbFileName + ";Version=3;");

DataTable dTable1 = new DataTable();

Рисунок 88 - Листинг модуля «Obototchet.cs»

Продолжение приложение 12

String sqlQuery;

m\_dbConn.Open();

sqlQuery = "Select \* from Клиент";

SQLiteDataAdapter adapter1 = new SQLiteDataAdapter(sqlQuery, m\_dbConn);

adapter1.Fill(dTable1);

Cbklient.DataSource = dTable1;

Cbklient.ValueMember = "Код\_клиента";

Cbklient.DisplayMember = "Фамилия";

if (dTable1.Rows.Count > 0)

{

for (int i = 0; i < dTable1.Rows.Count; i++)

{

Cbklient.SelectedIndex = i;

}

}

m\_dbConn.Close();

}

}

}

Рисунок 89 - Листинг модуля «Obototchet.cs»

Листинг модуля «Kredit.cs» представлен на рисунках 90-92.

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.Data.SQLite;

using System.IO;

namespace Bank

{

public partial class Frmkredit : Form

{

private String dbFileName;

private SQLiteConnection m\_dbConn;

public Frmkredit()

{

InitializeComponent();

}

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

{

m\_dbConn = new SQLiteConnection();

DataTable dTable = new DataTable();

String sqlQuery;

dbFileName = "bank.sqlite";

m\_dbConn = new SQLiteConnection("Data Source=" + dbFileName + ";Version=3;");

m\_dbConn.Open();

try

{

sqlQuery = "SELECT \* FROM Кредит";

Рисунок 90 - Листинг модуля «Kredit.cs»

Продолжение приложение 12

SQLiteDataAdapter adapter = new SQLiteDataAdapter(sqlQuery, m\_dbConn);

adapter.Fill(dTable);

Dtkredit.DataSource = dTable;

}

catch (SQLiteException ex)

{

MessageBox.Show("Error: " + ex.Message);

}

finally

{

m\_dbConn.Close();

}

}

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

{

this.Close();

}

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

{

m\_dbConn = new SQLiteConnection();

DataTable dTable = new DataTable();

String sqlQuery;

dbFileName = "bank.sqlite";

m\_dbConn = new SQLiteConnection("Data Source=" + dbFileName + ";Version=3;");

m\_dbConn.Open();

try

{

sqlQuery = $"SELECT \* FROM Кредит where Дата\_кредитования BETWEEN '{DtpOt.Value.ToString("dd.MM.yyyy")}' and '{DtpDo.Value.ToString("dd.MM.yyyy")}'";

SQLiteDataAdapter adapter = new SQLiteDataAdapter(sqlQuery, m\_dbConn);

adapter.Fill(dTable);

Dtkredit.DataSource = dTable;

}

catch (SQLiteException ex)

{

MessageBox.Show("Error: " + ex.Message);

}

finally

{

m\_dbConn.Close();

}

}

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

{

m\_dbConn = new SQLiteConnection();

DataTable dTable = new DataTable();

String sqlQuery;

dbFileName = "bank.sqlite";

m\_dbConn = new SQLiteConnection("Data Source=" + dbFileName + ";Version=3;");

m\_dbConn.Open();

Рисунок 91 - Листинг модуля «Kredit.cs»

Продолжение приложение 12

try

{

sqlQuery = "SELECT \* FROM Кредит";

SQLiteDataAdapter adapter = new SQLiteDataAdapter(sqlQuery, m\_dbConn);

adapter.Fill(dTable);

Dtkredit.DataSource = dTable;

}

catch (SQLiteException ex)

{

MessageBox.Show("Error: " + ex.Message);

}

finally

{

m\_dbConn.Close();

}

}

}

}

Рисунок 92 - Листинг модуля «Kredit.cs»

Листинг модуля «Klient.cs» представлен на рисунках 93-96.

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.Data.SQLite;

using System.IO;

namespace Bank

{

public partial class FrmKlient : Form

{

private String dbFileName;

private void view()

{

m\_dbConn = new SQLiteConnection();

DataTable dTable = new DataTable();

String sqlQuery;

dbFileName = "bank.sqlite";

m\_dbConn = new SQLiteConnection("Data Source=" + dbFileName + ";Version=3;");

m\_dbConn.Open();

try

{

sqlQuery = "SELECT \* FROM Клиент";

SQLiteDataAdapter adapter = new SQLiteDataAdapter(sqlQuery, m\_dbConn);

adapter.Fill(dTable);

DGKlient.DataSource = dTable;

}

Рисунок 93 - Листинг модуля «Klient.cs»

Продолжение приложение 12

catch (SQLiteException ex)

{

MessageBox.Show("Error: " + ex.Message);

}

finally

{

m\_dbConn.Close();

}

}

private SQLiteConnection m\_dbConn;

public static string kod\_klienta;

public FrmKlient()

{

InitializeComponent();

}

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

{

this.Close();

}

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

{

view();

}

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

{

m\_dbConn = new SQLiteConnection();

String sqlQuery;

DataTable dKredit = new DataTable();

DataTable dVklad = new DataTable();

dbFileName = "bank.sqlite";

TxtDan.Text = "";

m\_dbConn = new SQLiteConnection("Data Source=" + dbFileName + ";Version=3;");

m\_dbConn.Open();

sqlQuery = "select \* from Паспорт where Серия='" + DGKlient.CurrentRow.Cells[6].Value.ToString() + "'";

SQLiteCommand pass = new SQLiteCommand(sqlQuery, m\_dbConn);

SQLiteDataReader reader = pass.ExecuteReader();

if (reader.HasRows)

{

reader.Read();

Txtpas.Text = $"Серия: {reader.GetValue(0).ToString()}\r\nНомер: {reader.GetValue(1).ToString()}\r\nКем выдан: {reader.GetValue(2)}\r\nДата выдачи: {reader.GetValue(3).ToString()}";

}

sqlQuery = "select Счета.Сумма\_на\_счете from Счета inner join Клиент on Клиент.Код\_клиента=Счета.Код\_клиента where Счета.Код\_клиента='" + DGKlient.CurrentRow.Cells[0].Value.ToString() + "'";

SQLiteCommand dannye = new SQLiteCommand(sqlQuery, m\_dbConn);

SQLiteDataReader reader1 = dannye.ExecuteReader();

if (reader1.HasRows)

{

reader1.Read();

Рисунок 94 - Листинг модуля «Klient.cs»

Продолжение приложение 12

TxtDan.Text = TxtDan.Text + $"Сумма на счете=: {reader1.GetValue(0).ToString()}\r\n";

}

sqlQuery = "select Кредит.Номер\_договора,Кредит.Сумма,Кредит.Сумма\_в\_месяц,Кредит.Осталось\_погасить from Кредит inner join Операции on Операции.Код\_кредита=Кредит.Код\_кредита inner join Счета on Счета.Номер\_счета=Операции.Номер\_счета inner join Клиент on Клиент.Код\_клиента=Счета.Код\_клиента where Клиент.Код\_клиента='" + DGKlient.CurrentRow.Cells[0].Value.ToString() + "'";

SQLiteDataAdapter adapter = new SQLiteDataAdapter(sqlQuery, m\_dbConn);

adapter.Fill(dKredit);

if (dKredit.Rows.Count > 0)

{

SQLiteCommand kredit = new SQLiteCommand(sqlQuery, m\_dbConn);

SQLiteDataReader reader2 = kredit.ExecuteReader();

if (reader2.HasRows)

{

reader2.Read();

TxtDan.Text = TxtDan.Text + $"Кредит:\r\nНомер договора=: {reader2.GetValue(0).ToString()}\r\nСумма: {reader2.GetValue(1).ToString()}\r\nсумма в месяц= {reader2.GetValue(2).ToString()}\r\nОсталось погасить: {reader2.GetValue(3).ToString()}\r\n";

}

}

else

TxtDan.Text = TxtDan.Text + "Кредит:\r\nКредита нету.\r\n";

sqlQuery = "select Вклады.Номер\_договора,Вклады.Сумма,Вид\_вклада.Наименование,Вид\_вклада.Срок\_хранения from Вклады inner join Вид\_вклада on Вид\_вклада.Код\_вида\_вклада=Вклады.Код\_вида\_вклада inner join Операции on Операции.Код\_вклада=Вклады.Код\_вклада inner join Счета on Счета.Номер\_счета=Операции.Номер\_счета inner join Клиент on Клиент.Код\_клиента=Счета.Код\_клиента where Клиент.Код\_клиента='" + DGKlient.CurrentRow.Cells[0].Value.ToString() + "'";

SQLiteDataAdapter adapter1 = new SQLiteDataAdapter(sqlQuery, m\_dbConn);

adapter1.Fill(dVklad);

if (dVklad.Rows.Count > 0)

{

SQLiteCommand Vklad = new SQLiteCommand(sqlQuery, m\_dbConn);

SQLiteDataReader reader3 = Vklad.ExecuteReader();

if (reader3.HasRows)

{

reader3.Read();

TxtDan.Text = TxtDan.Text + $"Вклад:\r\nНомер договора=: {reader3.GetValue(0).ToString()}\r\nСумма: {reader3.GetValue(1).ToString()}\r\nВид вклада= {reader3.GetValue(2).ToString()}\r\nСрок хранения: {reader3.GetValue(3).ToString()} года\r\n";

}

}

else

TxtDan.Text = TxtDan.Text + "Вклад:\r\nВклада нету.\r\n";

m\_dbConn.Close();

kod\_klienta=DGKlient.CurrentRow.Cells[0].Value.ToString();

}

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

{

Рисунок 95 - Листинг модуля «Klient.cs»

Продолжение приложение 12

FrmEditLastname edit = new FrmEditLastname();

edit.ShowDialog();

}

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

{

FrmPerevod perevod = new FrmPerevod();

perevod.ShowDialog();

}

private void textBox1\_TextChanged(object sender, EventArgs e)

{

if (!string.IsNullOrEmpty(Txtsearch.Text) && !string.IsNullOrWhiteSpace(Txtsearch.Text))

{

m\_dbConn = new SQLiteConnection();

dbFileName = "bank.sqlite";

m\_dbConn = new SQLiteConnection("Data Source=" + dbFileName + ";Version=3;");

m\_dbConn.Open();

try

{

DataTable dTable = new DataTable();

String sqlQuery = $"SELECT \* from Клиент where Фамилия like '{Txtsearch.Text}%'";

using (SQLiteDataAdapter adapter = new SQLiteDataAdapter(sqlQuery, m\_dbConn))

{

adapter.Fill(dTable);

DGKlient.DataSource = dTable;

}

}

catch (SQLiteException ex)

{

MessageBox.Show("Error: " + ex.Message);

}

}

else

{

view();

}

}

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

{

FrmVypotchet vyp = new FrmVypotchet();

vyp.ShowDialog();

}

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

{

FrmObototchet ob = new FrmObototchet();

ob.ShowDialog();

}

}

}

Рисунок 96 - Листинг модуля «Klient.cs»

Продолжение приложение 12

Листинг модуля «Glavnoemenu.cs» представлен на рисунках 97-98.

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.Data.SQLite;

using System.IO;

namespace Bank

{

public partial class FrmGlavnoemenu : Form

{

public string FIO {

set { label1.Text = $"Здравствуйте, {value}"; }

}

public FrmGlavnoemenu()

{

InitializeComponent();

}

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

{

this.Hide();

FRMautorization Autorization = new FRMautorization();

Autorization.Show();

}

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

{

timer.Enabled = true;

if (Properties.Settings.Default.Role == "Администратор")

{

menuStrip1.Visible = true;

}

if (Properties.Settings.Default.Role == "Операционист")

{

menuStrip1.Visible = false;

}

timer.Start();

Ldate.Text = "Дата: " + DateTime.Now.ToString("dd MMMM yyyy");

}

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

{

Ltime.Text = string.Format("Текущее время: {0}", DateTime.Now.ToString("HH:mm:ss"));

}

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

{

FrmKlient klient = new FrmKlient();

Рисунок 97 - Листинг модуля «Glavnoemenu.cs»

Продолжение приложение 12

klient.ShowDialog();

}

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

{

FrmOperacii oper = new FrmOperacii();

oper.ShowDialog();

}

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

{

Frmkredit vklady = new Frmkredit();

vklady.ShowDialog();

}

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

{

Frmvklady vklad = new Frmvklady();

vklad.ShowDialog();

}

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

{

FrmAddusers user = new FrmAddusers();

user.ShowDialog();

}

}

}

Рисунок 98 - Листинг модуля «Glavnoemenu.cs»

Листинг модуля «EditLastname.cs» представлен на рисунках 99-100.

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.Data.SQLite;

using System.IO;

namespace Bank

{

public partial class FrmEditLastname : Form

{

private String dbFileName;

private SQLiteConnection m\_dbConn;

private SQLiteCommand m\_sqlCmd;

public FrmEditLastname()

{

InitializeComponent();

}

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

{

Рисунок 99 - Листинг модуля «EditLastname.cs»

Продолжение приложение 12

m\_dbConn = new SQLiteConnection();

m\_sqlCmd = new SQLiteCommand();

DataTable dTable = new DataTable();

String sqlQuery;

dbFileName = "bank.sqlite";

m\_dbConn = new SQLiteConnection("Data Source=" + dbFileName + ";Version=3;");

m\_dbConn.Open();

m\_sqlCmd.Connection = m\_dbConn;

if (TxtName.Text == "")

{

MessageBox.Show("Заполните все поля.");

return;

}

else

{

try

{

sqlQuery = "SELECT \* FROM Клиент where Код\_клиента='" + FrmKlient.kod\_klienta + "'";

SQLiteDataAdapter adapter = new SQLiteDataAdapter(sqlQuery, m\_dbConn);

adapter.Fill(dTable);

if (dTable.Rows.Count > 0)

{

m\_sqlCmd.CommandText = $"UPDATE Клиент SET Фамилия='" + TxtName.Text + "'";

m\_sqlCmd.ExecuteNonQuery();

MessageBox.Show("фамилия изменена.");

}

else

{

MessageBox.Show("Выберите клиента.");

return;

}

}

catch (SQLiteException ex)

{

MessageBox.Show("Error: " + ex.Message);

}

finally

{

m\_dbConn.Close();

this.Close();

}

}

}

private void TxtName\_KeyPress(object sender, KeyPressEventArgs e)

{

char text = e.KeyChar;

if (Char.IsDigit(text) && text != 8)

{

e.Handled = true;

}

}

}

}

Рисунок 100 - Листинг модуля «EditLastname.cs»

Продолжение приложение 12

Листинг модуля «Addvydvklada.cs» представлен на рисунках 101-102.

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.Data.SQLite;

using System.IO;

namespace Bank

{

public partial class FrmAddvydvklada : Form

{

private String dbFileName;

private SQLiteConnection m\_dbConn;

private SQLiteCommand m\_sqlCmd;

public FrmAddvydvklada()

{

InitializeComponent();

}

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

{

m\_dbConn = new SQLiteConnection();

m\_sqlCmd = new SQLiteCommand();

DataTable dTable = new DataTable();

dbFileName = "bank.sqlite";

m\_dbConn = new SQLiteConnection("Data Source=" + dbFileName + ";Version=3;");

m\_dbConn.Open();

m\_sqlCmd.Connection = m\_dbConn;

try

{

m\_sqlCmd.CommandText = $"INSERT INTO Вид\_вклада values (null,'{TxtNAme.Text}','{Cbstavka.Text}','{Cbsrok.Text}')";

m\_sqlCmd.ExecuteNonQuery();

MessageBox.Show("Вид вклада добавлен.");

this.Close();

}

catch (SQLiteException ex)

{

MessageBox.Show("Error: " + ex.Message);

}

finally

{

m\_dbConn.Close();

}

}

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

{

Рисунок 101 - Листинг модуля «Addvydvklada.cs»

Продолжение приложение 12

this.Close();

}

private void TxtNAme\_KeyPress(object sender, KeyPressEventArgs e)

{

char text = e.KeyChar;

if (Char.IsDigit(text) && text != 8)

{

e.Handled = true;

}

}

}

}

Рисунок 102 - Листинг модуля «Addvydvklada.cs»

Листинг модуля «Addusers.cs» представлен на рисунках 103-104.

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.Data.SQLite;

using System.IO;

namespace Bank

{

public partial class FrmAddusers : Form

{

private String dbFileName;

private SQLiteConnection m\_dbConn;

private SQLiteCommand m\_sqlCmd;

public FrmAddusers()

{

InitializeComponent();

}

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

{

m\_dbConn = new SQLiteConnection();

m\_sqlCmd = new SQLiteCommand();

String sqlQuery;

DataTable dTable = new DataTable();

dbFileName = "bank.sqlite";

m\_dbConn = new SQLiteConnection("Data Source=" + dbFileName + ";Version=3;");

m\_dbConn.Open();

m\_sqlCmd.Connection = m\_dbConn;

if (TxtFio.Text == "" | TxtLogin.Text == "" | TxtPass.Text == "")

{

MessageBox.Show("Заполните все поля!");

}

Рисунок 103 - Листинг модуля «Addusers.cs»

Продолжение приложение 12

else

{

sqlQuery = "SELECT \* FROM Пользователи where Логин='" + TxtLogin.Text + "' and Пароль='" + TxtPass.Text + "'";

SQLiteDataAdapter adapter = new SQLiteDataAdapter(sqlQuery, m\_dbConn);

adapter.Fill(dTable);

if (dTable.Rows.Count == 0)

{

try

{

m\_sqlCmd.CommandText = $"INSERT INTO Пользователи values (null,'{TxtFio.Text}','{TxtLogin.Text}','{TxtPass.Text}','Операционист')";

m\_sqlCmd.ExecuteNonQuery();

MessageBox.Show("Пользователь добавлен.");

this.Close();

}

catch (SQLiteException ex)

{

MessageBox.Show("Error: " + ex.Message);

}

finally

{

m\_dbConn.Close();

this.Close();

}

}

else

{

MessageBox.Show("Такой пользователь существует!");

m\_dbConn.Close();

}

}

}

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

{

Close();

}

private void TxtFio\_KeyPress(object sender, KeyPressEventArgs e)

{

char text = e.KeyChar;

if (Char.IsDigit(text) && text != 8)

{

e.Handled = true;

}

}

}

}

Рисунок 104 - Листинг модуля «Addusers.cs»

Листинг модуля «Addorer.cs» представлен на рисунках 105-120.

using System;

using System.Collections.Generic;

using System.ComponentModel;

Рисунок 105 - Листинг модуля «Addorer.cs»

Продолжение приложение 12

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.Data.SQLite;

using System.IO;

namespace Bank

{

public partial class Frmaddoper : Form

{

private String dbFileName;

private SQLiteConnection m\_dbConn;

private SQLiteCommand m\_sqlCmd;

public int s;

public int p;

public Frmaddoper()

{

InitializeComponent();

}

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

{

this.Close();

}

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

{

Gbklient.Visible = true;

Pklient.Enabled = false;

Gbklient.Enabled = true;

}

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

{

Рисунок 106 - Листинг модуля «Addorer.cs»

Продолжение приложение 12

if (Cmbselectoper.SelectedIndex == 0)

{

GbKredit.Visible = true;

Pselectoper.Enabled = false;

GbVklad.Visible = false;

}

if (Cmbselectoper.SelectedIndex == 1)

{

GbKredit.Visible = false;

Pselectoper.Enabled = false;

GbVklad.Visible = true;

}

if (Cmbselectoper.SelectedIndex == 2)

{

GbKredit.Visible = false;

Pselectoper.Enabled = false;

GbVklad.Visible = false;

}

}

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

{

m\_dbConn = new SQLiteConnection();

m\_sqlCmd = new SQLiteCommand();

string sqlQuery;

DataTable dTable = new DataTable();

DataTable dTable1 = new DataTable();

dbFileName = "bank.sqlite";

m\_dbConn = new SQLiteConnection("Data Source=" + dbFileName + ";Version=3;");

m\_dbConn.Open();

m\_sqlCmd.Connection = m\_dbConn;

if (Cmbselectoper.SelectedIndex == 0)

{

if (Txtnomdogkred.Text == "" | Txtsumkred.Text == "" | CBStavkakredit.Text == "" | CBKolmeskrdit.Text == "")

{

MessageBox.Show("Заполните все поля");

m\_dbConn.Close();

Рисунок 107 - Листинг модуля «Addorer.cs»

Продолжение приложение 12

return;

}

else

{

sqlQuery = "SELECT \* FROM Кредит where Номер\_договора='" + Txtnomdogkred.Text + "'";

SQLiteDataAdapter adapter1 = new SQLiteDataAdapter(sqlQuery, m\_dbConn);

adapter1.Fill(dTable);

if (dTable.Rows.Count == 0)

{

if (s == 1)

{

try

{

sqlQuery = "SELECT Код\_клиента FROM Клиент where Серия\_паспорта='" + Txtseria.Text + "'";

SQLiteDataAdapter adapter = new SQLiteDataAdapter(sqlQuery, m\_dbConn);

adapter.Fill(dTable1);

if (dTable1.Rows.Count > 0)

{

m\_sqlCmd.CommandText = $"INSERT INTO Счета values (null,'{dTable1.Rows[0][0]}','{DateTime.Now.ToString("dd MMMM yyyy")}',null,{Txtsumkred.Text})";

m\_sqlCmd.ExecuteNonQuery();

}

}

catch (SQLiteException ex)

{

MessageBox.Show("Error: " + ex.Message);

}

}

if (s == 0)

{

try

{

m\_sqlCmd.CommandText = $"INSERT INTO Счета values (null,'{CBFamalyklienta.SelectedValue}','{DateTime.Now.ToString("dd MMMM

Рисунок 108 - Листинг модуля «Addorer.cs»

Продолжение приложение 12

yyyy")}',null,{Txtsumkred.Text})";

m\_sqlCmd.ExecuteNonQuery();

}

catch (SQLiteException ex)

{

MessageBox.Show("Error: " + ex.Message);

}

}

try

{

m\_sqlCmd.CommandText = $"INSERT INTO Кредит values (null,'{Txtnomdogkred.Text}',{Txtsumkred.Text},'{CBStavkakredit.Text}','{DtKredit.Value.ToString("dd.MM.yyyy")}',({Txtsumkred.Text} + {Txtsumkred.Text}\*{CBStavkakredit.Text}/100)/'{CBKolmeskrdit.Text}',{Txtsumkred.Text} + {Txtsumkred.Text}\*{CBStavkakredit.Text}/100 )";

m\_sqlCmd.ExecuteNonQuery();

p = 0;

MessageBox.Show("Кредит добавлен.");

}

catch (SQLiteException ex)

{

MessageBox.Show("Error: " + ex.Message);

}

finally

{

m\_dbConn.Close();

GbKredit.Enabled = false;

GpOper.Enabled = true;

this.TCcontrol.SelectedIndex = 2;

CbNameoper.SelectedIndex = 0;

}

}

else

{

MessageBox.Show("Договор с таким номером уже заключен.");

m\_dbConn.Close();

return;

}}}

Рисунок 109 - Листинг модуля «Addorer.cs»

Продолжение приложение 12

}

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

{

m\_dbConn = new SQLiteConnection();

m\_sqlCmd = new SQLiteCommand();

DataTable dTable = new DataTable();

String sqlQuery;

dbFileName = "bank.sqlite";

m\_dbConn = new SQLiteConnection("Data Source=" + dbFileName + ";Version=3;");

m\_dbConn.Open();

m\_sqlCmd.Connection = m\_dbConn;

if (Txtseria.Text == "" | Txtnomer.Text == "" | Txtkemvyd.Text == "" | TxtLastname.Text == "" | TxtFirstname.Text == "" | TxtOtchestvo.Text == "" | TxtAdres.Text == "" | CBPol.Text == "" | TxtTelephone.Text == "")

{

MessageBox.Show("Заполните все поля.");

return;

}

else

{

try

{

sqlQuery = "SELECT \* FROM Паспорт where Серия='" + Txtseria.Text + "' and Номер='" + Txtnomer.Text + "'";

SQLiteDataAdapter adapter = new SQLiteDataAdapter(sqlQuery, m\_dbConn);

adapter.Fill(dTable);

if (dTable.Rows.Count == 0)

{

m\_sqlCmd.CommandText = $"INSERT INTO Паспорт values ('{Txtseria.Text}',{Txtnomer.Text},'{Txtkemvyd.Text}','{DtpDatevyd.Value.ToString("dd.MM.yyyy")}')";

m\_sqlCmd.ExecuteNonQuery();

m\_sqlCmd.CommandText = $"INSERT INTO Клиент values (null,'{TxtLastname.Text}','{TxtFirstname.Text}','{TxtOtchestvo.Text}','{DtpDatebirth.Value.ToString("dd.MM.yyyy")}','{CBPol.Text}','{Txtseria.Text}','{TxtAdres.Text}','{TxtTelephone.Text}')";

Рисунок 110 - Листинг модуля «Addorer.cs»

Продолжение приложение 12

m\_sqlCmd.ExecuteNonQuery();

Pklient.Enabled = false;

Gbklient.Enabled = false;

Pselectoper.Enabled = true;

this.TCcontrol.SelectedIndex = 1;

s = 1;

MessageBox.Show("Клиент добавлен.");

}

else

{

MessageBox.Show("Клиент с такими паспортными данными уже есть в базе.");

return;

}

}

catch (SQLiteException ex)

{

MessageBox.Show("Error: " + ex.Message);

}

finally

{

m\_dbConn.Close();

}

}

}

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

{

m\_dbConn = new SQLiteConnection();

m\_sqlCmd = new SQLiteCommand();

string sqlQuery;

DataTable dTable = new DataTable();

DataTable dTable1 = new DataTable();

dbFileName = "bank.sqlite";

m\_dbConn = new SQLiteConnection("Data Source=" + dbFileName + ";Version=3;");

m\_dbConn.Open();

m\_sqlCmd.Connection = m\_dbConn;

Рисунок 111 - Листинг модуля «Addorer.cs»

Продолжение приложение 12

//Добавление вклада

if (Cmbselectoper.SelectedIndex == 1)

{

if (Txtnomvklada.Text == "" | Cbvydvklada.Text == "" | Txtsumvklada.Text == "")

{

MessageBox.Show("Заполните все поля");

m\_dbConn.Close();

return;

}

else

{

sqlQuery = "SELECT \* FROM Вклады where Номер\_договора='" + Txtnomvklada.Text + "'";

SQLiteDataAdapter adapter1 = new SQLiteDataAdapter(sqlQuery, m\_dbConn);

adapter1.Fill(dTable);

if (dTable.Rows.Count == 0)

{

if (s == 1)

{

try

{

sqlQuery = "SELECT Код\_клиента FROM Клиент where Серия\_паспорта='" + Txtseria.Text + "'";

SQLiteDataAdapter adapter = new SQLiteDataAdapter(sqlQuery, m\_dbConn);

adapter.Fill(dTable1);

if (dTable1.Rows.Count > 0)

{

m\_sqlCmd.CommandText = $"INSERT INTO Счета values (null,'{dTable1.Rows[0][0]}','{DateTime.Now.ToString("ddMMMMyyyy")}',null,{Txtsumvklada.Text})";

m\_sqlCmd.ExecuteNonQuery();

}

}

catch (SQLiteException ex)

{

MessageBox.Show("Error: " + ex.Message);

Рисунок 112 - Листинг модуля «Addorer.cs»

Продолжение приложение 12

}

}

if (s == 0)

{

try

{

m\_sqlCmd.CommandText = $"INSERT INTO Счета values (null,'{CBFamalyklienta.SelectedValue}','{DateTime.Now.ToString("dd MMMM yyyy")}',null,'{Txtsumvklada.Text}')";

m\_sqlCmd.ExecuteNonQuery();

}

catch (SQLiteException ex)

{

MessageBox.Show("Error: " + ex.Message);

}

}

try

{

m\_sqlCmd.CommandText = $"INSERT INTO Вклады values (null,'{Cbvydvklada.SelectedValue}',{Txtnomvklada.Text},'{DtDatevklada.Value.ToString("dd.MM.yyyy")}','{Txtsumvklada.Text}')";

m\_sqlCmd.ExecuteNonQuery();

p = 1;

MessageBox.Show("Вклад добавлен.");

}

catch (SQLiteException ex)

{

MessageBox.Show("Error: " + ex.Message);

}

finally

{

m\_dbConn.Close();

GbKredit.Enabled = false;

GpOper.Enabled = true;

this.TCcontrol.SelectedIndex = 2;

CbNameoper.SelectedIndex = 0;

}

}

Рисунок 113 - Листинг модуля «Addorer.cs»

Продолжение приложение 12

else

{

MessageBox.Show("Договор с таким номером уже заключен.");

return;

}

}

}

}

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

{

m\_dbConn = new SQLiteConnection();

DataTable dTable = new DataTable();

DataTable dTable1 = new DataTable();

DataTable dTable2 = new DataTable();

DataTable dTable3 = new DataTable();

String sqlQuery;

dbFileName = "bank.sqlite";

m\_dbConn = new SQLiteConnection("Data Source=" + dbFileName + ";Version=3;");

m\_dbConn.Open();

sqlQuery = "Select \* from Вид\_вклада";

SQLiteDataAdapter adapter = new SQLiteDataAdapter(sqlQuery, m\_dbConn);

adapter.Fill(dTable);

Cbvydvklada.DataSource = dTable;

Cbvydvklada.ValueMember = "Код\_вида\_вклада";

Cbvydvklada.DisplayMember = "Наименование";

if (dTable.Rows.Count > 0)

{

for (int i = 0; i < dTable1.Rows.Count; i++)

{

Cbvydvklada.SelectedIndex = i;

}

}

else

{

MessageBox.Show("Видов вклада нету.");

return;

Рисунок 114 - Листинг модуля «Addorer.cs»

Продолжение приложение 12

}

sqlQuery = "Select \* from Клиент";

SQLiteDataAdapter adapter1 = new SQLiteDataAdapter(sqlQuery, m\_dbConn);

adapter1.Fill(dTable1);

CBFamalyklienta.DataSource = dTable1;

CBFamalyklienta.ValueMember = "Код\_клиента";

CBFamalyklienta.DisplayMember = "Фамилия";

if (dTable1.Rows.Count > 0)

{

for (int i = 0; i < dTable1.Rows.Count; i++)

{

CBFamalyklienta.SelectedIndex = i;

}

}

else { Btnselectklient.Enabled = false; }

sqlQuery = $"SELECT \* FROM Пользователи where Код\_пользователя='{FRMautorization.kod}'";

SQLiteDataAdapter adapter2 = new SQLiteDataAdapter(sqlQuery, m\_dbConn);

adapter2.Fill(dTable2);

Cbsotrud.DataSource = dTable2;

Cbsotrud.ValueMember = "Код\_пользователя";

Cbsotrud.DisplayMember = "ФИО";

if (dTable2.Rows.Count > 0)

{

for (int i = 0; i < dTable2.Rows.Count; i++)

{

Cbsotrud.SelectedIndex = i;

}

}

else

{

return;

}

m\_dbConn.Close();

}

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

{

m\_dbConn = new SQLiteConnection();

Рисунок 115 - Листинг модуля «Addorer.cs»

Продолжение приложение 12

m\_sqlCmd = new SQLiteCommand();

string sqlQuery1;

DataTable dTable = new DataTable();

string sqlQuery2;

string sqlQuery3;

DataTable dTable1 = new DataTable();

DataTable dTable2 = new DataTable();

dbFileName = "bank.sqlite";

m\_dbConn = new SQLiteConnection("Data Source=" + dbFileName + ";Version=3;");

m\_dbConn.Open();

m\_sqlCmd.Connection = m\_dbConn;

if (CbNameoper.Text == "" | Cbsotrud.Text == "")

{

MessageBox.Show("Заполните все поля.");

}

else

{

if (p == 0)

{

try

{

if (s == 0)

{

sqlQuery1 = "SELECT Счета.Номер\_счета FROM Счета inner join Клиент on Клиент.Код\_клиента=Счета.Код\_клиента where Клиент.Код\_клиента='" + CBFamalyklienta.SelectedValue + "'";

SQLiteDataAdapter adapter = new SQLiteDataAdapter(sqlQuery1, m\_dbConn);

adapter.Fill(dTable);

sqlQuery2 = "SELECT \* FROM Кредит where Номер\_договора='" + Txtnomdogkred.Text + "'";

SQLiteDataAdapter adapter1 = new SQLiteDataAdapter(sqlQuery2, m\_dbConn);

adapter1.Fill(dTable1);

m\_sqlCmd.CommandText = $"INSERT INTO Операции values (null,'{CbNameoper.Text}',null,{dTable1.Rows[0][0].ToString()},null,{dTable.Rows[0][0].ToString()},{Cbsotrud.SelectedValue},'{Dtdateoper.Value.ToString("dd.MM.yyyy")}')";

Рисунок 116 - Листинг модуля «Addorer.cs»

Продолжение приложение 12

m\_sqlCmd.ExecuteNonQuery();

MessageBox.Show("Кредит добавлен.");

}

if (s == 1)

{

sqlQuery3 = "SELECT Счета.Номер\_счета FROM Счета inner join Клиент on Клиент.Код\_клиента=Счета.Код\_клиента where Серия\_паспорта='" + Txtseria.Text + "'";

SQLiteDataAdapter adapter = new SQLiteDataAdapter(sqlQuery3, m\_dbConn);

adapter.Fill(dTable2);

sqlQuery2 = "SELECT \* FROM Кредит where Номер\_договора='" + Txtnomdogkred.Text + "'";

SQLiteDataAdapter adapter1 = new SQLiteDataAdapter(sqlQuery2, m\_dbConn);

adapter1.Fill(dTable1);

m\_sqlCmd.CommandText = $"INSERT INTO Операции values (null,'{CbNameoper.Text}',null,{dTable1.Rows[0][0].ToString()},null,{dTable2.Rows[0][0].ToString()},{Cbsotrud.SelectedValue},'{Dtdateoper.Value.ToString("dd.MM.yyyy")}')";

m\_sqlCmd.ExecuteNonQuery();

MessageBox.Show("Кредит добавлен.");

}

}

catch (SQLiteException ex)

{

MessageBox.Show("Error: " + ex.Message);

}

finally

{

m\_dbConn.Close();

this.Close();

}

}

// Операция со вкладом

if (p == 1)

Рисунок 117 - Листинг модуля «Addorer.cs»

Продолжение приложение 12

{

try

{

if (s == 0)

{

sqlQuery1 = "SELECT Счета.Номер\_счета FROM Счета inner join Клиент on Клиент.Код\_клиента=Счета.Код\_клиента where Клиент.Код\_клиента='" + CBFamalyklienta.SelectedValue + "'";

SQLiteDataAdapter adapter = new SQLiteDataAdapter(sqlQuery1, m\_dbConn);

adapter.Fill(dTable);

sqlQuery2 = "SELECT \* FROM Вклады where Номер\_договора='" + Txtnomvklada.Text + "'";

SQLiteDataAdapter adapter1 = new SQLiteDataAdapter(sqlQuery2, m\_dbConn);

adapter1.Fill(dTable1);

m\_sqlCmd.CommandText = $"INSERT INTO Операции values (null,'{CbNameoper.Text}',{dTable1.Rows[0][0].ToString()},null,null,{dTable.Rows[0][0].ToString()},{Cbsotrud.SelectedValue},'{Dtdateoper.Value.ToString("dd.MM.yyyy")}')";

m\_sqlCmd.ExecuteNonQuery();

MessageBox.Show("Вклад добавлен.");

}

if (s == 1)

{

sqlQuery3 = "SELECT Счета.Номер\_счета FROM Счета inner join Клиент on Клиент.Код\_клиента=Счета.Код\_клиента where Серия\_паспорта='" + Txtseria.Text + "'";

SQLiteDataAdapter adapter = new SQLiteDataAdapter(sqlQuery3, m\_dbConn);

adapter.Fill(dTable2);

sqlQuery2 = "SELECT \* FROM Вклады where Номер\_договора='" + Txtnomvklada.Text + "'";

SQLiteDataAdapter adapter1 = new SQLiteDataAdapter(sqlQuery2, m\_dbConn);

adapter1.Fill(dTable1);

Рисунок 118 - Листинг модуля «Addorer.cs»

Продолжение приложение 12

m\_sqlCmd.CommandText = $"INSERT INTO Операции values (null,'{CbNameoper.Text}',{dTable1.Rows[0][0].ToString()},null,null,{dTable2.Rows[0][0].ToString()},{Cbsotrud.SelectedValue},'{Dtdateoper.Value.ToString("dd.MM.yyyy")}')";

m\_sqlCmd.ExecuteNonQuery();

MessageBox.Show("Вклад добавлен.");

}

}

catch (SQLiteException ex)

{

MessageBox.Show("Error: " + ex.Message);

}

finally

{

m\_dbConn.Close();

this.Close();

}

}

}

}

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

{

Pklient.Enabled = false;

Gbklient.Enabled = false;

Pselectfam.Enabled = false;

Pselectoper.Enabled = true;

this.TCcontrol.SelectedIndex = 1;

s = 0;

}

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

{

Pselectfam.Visible = true;

Pklient.Enabled = false;

}

private void Txtseria\_KeyPress(object sender, KeyPressEventArgs e)

{

char text = e.KeyChar;

if (!Char.IsDigit(text) && text != 8)

Рисунок 119 - Листинг модуля «Addorer.cs»

Продолжение приложение 12

{

e.Handled = true;

}

}

private void TxtLastname\_KeyPress(object sender, KeyPressEventArgs e)

{

char text = e.KeyChar;

if (Char.IsDigit(text) && text != 8)

{

e.Handled = true;

}

}

private void Txtsumvklada\_KeyPress(object sender, KeyPressEventArgs e)

{

char text = e.KeyChar;

if (!Char.IsDigit(text) && text != 8)

{

e.Handled = true;

}

}

}

}

Рисунок 120 - Листинг модуля «Addorer.cs»