Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерного проектирования

Кафедра электронных вычислительных средств

Дисциплина: Операционные системы и управление базами данных

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовой работе

на тему

**БАНКОВСКИЙ УЧЁТ**

БГУИР КП 1-58 01 01 006 ПЗ

|  |  |
| --- | --- |
| Студент: | Малиновская Д.В. |
| Руководитель: | Лихачёв Д. С. |

Минск 2014

**СОДЕРЖАНИЕ**

ВВЕДЕНИЕ………………………………. ………………………………………5

1 АНАЛИЗ ТЕХНИЧЕСКОГО ЗАДАНИЯ………………………………….…..6

2 ПРОЕКТИРОВАНИЕ СТРУКТУРЫ ИНФОРМАЦИОННОЙ СИСТЕМЫ………………………………………………………………..……….7

# 3 РАЗРАБОТКА БД…………………………………………………………….....9

4 РАЗРАБОТКА КЛИЕНТСКОЙ ЧАСТИ…………………………………..…11

# 5 ПРОВЕРКА РАБОТОСПОСОБНОСТИ РАЗРАБОТАННОЙ ИНФОРМАЦИОННОЙ СИСТЕМЫ…………………………………………..15

ЗАКЛЮЧЕНИЕ………...………………………………………………………..17

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ……………………………18

ПРИЛОЖЕНИЕ А - диаграмма базы данных «Банковский учёт»…..………19

ПРИЛОЖЕНИЕ Б – листинг программы………………………………………20

**ВВЕДЕНИЕ**

В данной курсовой работе рассматривается создание базы данных банковского учёта, рассчитанной на облегчение работы администратора по работе с клиентами, их счетами, займами и депозитами, а также различными операциями над ними, такими как добавление, удаление, транзакция и подсчёт процентов.

База данных – это единое, вместительное хранилище разнообразных данных и описаний их структур, которое после своего определения, осуществляемого отдельно и независимо от приложений, используется одновременно многими приложениями.

Кроме данных база данных может содержать средства, позволяющие каждому из пользователей оперировать только теми данными, которые входят в его компетенцию. В результате взаимодействия данных, содержащихся в базе, с методами, доступными конкретным пользователям, образуется информация, которую они потребляют и на основании которой в пределах собственной компетенции производят ввод и редактирование данных.

С ростом численности банков и банковских услуг стремительно усиливается и конкуренция, что неизбежно приводит к необходимости эффективно и рационально использовать имеющиеся ресурсы. В этих условиях для успешного ведения бизнеса необходимо инвестировать в средства и инструменты его поддержания и развития. Один из основных инструментов развития – это современная база данных ресторанов.

База данных банка или Современная система автоматизации банка – это профессиональная система учёта клиентов и их действий, многофункциональная и легко модернизируемая. Целью автоматизации является повышение эффективности учёта клиента и его счетов, ускорение обслуживания и минимизация возможных злоупотреблений, особенно воровства. Значительная доля успеха складывается из отличного сервиса и оперативного обслуживания. Именно возможности автоматизации позволяют оптимально сочетать скорость и качество.

Таким образом, в результате автоматизации банк имеет возможность постоянно повышать конкурентоспособность, рентабельность своего бизнеса.

**1 АНАЛИЗ ТЕХНИЧЕСКОГО ЗАДАНИЯ**

Задача технического анализа предметной области заключается в определении целей создания системы, изучении предметной области и определении требований пользователей. Определяются границы предметной области, отбрасываются несущественные части. Накапливаются знания о предметной области. Описание обычно делается на естественном языке.

Основная цель данного клиентского приложения – предоставить администратору банка отслеживать банковские счета клиентов банка, производить различные операции над счетами. Программный продукт должен позволять осуществлять добавление и удаление клиента. Создание для пользователя счёта в разной валюте. Пополнение или снятие со счёта любой денежной суммы. Добавление займа и депозита для счёта. Должны производиться транзакции с одного счёта на другой. Подсчёт процентов на займ и депозит за любой промежуток времени. Печать информации о займах и депозитах. В программе должны быть реализованы следующие компоненты:

1. Хранение всей информации в базе данных.

2. Возможность добавления и удаления пользователя.

3. Возможность поиска счёта по фамилии и № паспорта.

4. Возможность добавления нового счёта (в разных валютах), займа и депозита для клиента.

5. Возможность добавить или снять деньги со счёта.

6. Возможность совершить транзакцию с одного счёта на другой.

7. Возможность подсчёта процентов для займа и депозита за любой промежуток времени.

8. Возможность печати информации о займах и депозитах в формате .txt.

Формы базы данных «Банковский учёт» демонстрируют удобные способы работы с таблицами и запросами. Они созданы для иллюстрации процессов ввода, изменения и просмотра данных, работы диалоговых окон с приглашением на ввод данных с последующей обработкой введенной информации, а также панелей управления, позволяющих открывать другие формы и отчеты базы данных пользователя.

**2 ПРОЕКТИРОВАНИЕ СТРУКТУРЫ ИНФОРМАЦИОННОЙ СИСТЕМЫ**

Под информационной системой понимают совокупность средств сбора, передачи, обработки и хранения информации, а также персонал, выполняющий подобные действия. Информационный фонд системы, как правило, представляет базу или совокупность БД, создаваемых в виде табличных, иерархических и сетевых структур.

Для хранения информации система будет использовать базу данных под управлением СУБД MS SQL Server 2012. В базе данных будет храниться вся необходимая информация для работы с приложением. База данных под управлением СУБД будет являться подсистемой данной системы. Второй подсистемой системы будет являться клиентская часть. Клиентская часть нужна для удобства работы с базой и получения нужной информации в контексте предметной области. Реализовываться она будет в качестве приложения Windows Forms.

Таблица 1. Описание функционала

|  |  |
| --- | --- |
| Действие | Результат |
| Запуск приложения | Главная форма приложения с выводом информации об общем количестве клиентов, счетов, депозитов, с кнопками создания клиента, счёта, займа и депозита, также с кнопкой поиска клиентов |
| Кнопка “Add customer” | Форма для добавления нового клиента |
| Кнопка “Печать меню” | Данные о займе/депозите выгружаются в .txt файл и автоматически открываются для просмотра |
| Кнопка “Add account” | Форма для добавления нового счёта с указанием паспортных данных ранее созданного клиента или с указанием новых паспортных данных |
| Кнопка “Add deposit” | Форма для добавления нового депозита с указанием паспортных данных ранее созданного клиента или с указанием новых паспортных данных |
| Кнопка “Add loan” | Форма для добавления нового займа с указанием паспортных данных ранее созданного клиента или с указанием новых паспортных данных |
| Кнопка “Find” | Поиск всех клиентов или поиск конкретного клиента с указанием фамилии и паспортных данных |

Продолжение Таблицы 1. Структура информационной системы

|  |  |
| --- | --- |
| Действие | Результат |
| Два раза кликаем по созданному клиенту | Переход на форму «Customer» |
| Форма «Customer» | Вся информация о клиенте |
| Кнопка “Accounts” | Форма для добавления нового счёта для данного клиента |
| Кнопка “Loans” | Форма для добавления нового займа для данного клиента |
| Кнопка “Deposits” | Форма для добавления нового депозита для данного клиента |
| Кнопка “Delete customer” | Удаление данного клиента из БД |
| Два раза кликаем по созданному займу | Форма для подсчёта процентов на займ за любой промежуток времени |
| Два раза кликаем по созданному депозиту | Форма для подсчёта процентов на займ за любой промежуток времени |
| Два раза кликаем по созданному ксчёту | Переход на форму “Editing account” |
| Форма “Editing account” | Имеет два поля: «Summary» и «Cash to transact» |
| Поле «Summary» | Операции с денежной суммой на счёте |
| Кнопка “Add cash” | Начисление какой-либо денежной суммы |
| Low cash | Снятие какой-либо денежной суммы |
| Annul account | Обнуление счёта |
| Поле «Cash to transact» | Перечисление денежной суммы с данного счёта на другой |
| Find | Поиск другого счёта |
| Transact | Транзакция с данного счёта на другой |

**3 РАЗРАБОТКА БД**

База данных (БД) – это совокупность специальным образом организованных данных хранимых в памяти вычислительной системы отображающих состояние объектов и их взаимосвязи в рассматриваемой предметной области. [1]

СУБД – комплекс языковых и программных средств, предназначенных для создания, ведения и совместного использования БД несколькими пользователями. СУБД позволяет: создавать БД; вставлять, обновлять, удалять и извлекать информацию из БД; предоставляет контролируемый доступ к базе данных. [1]

В качестве СУБД для курсового проекта была выбрана MS SQL Server 2012.

Microsoft SQL Server обеспечивает значительные преимущества для бизнеса и обладает широкими возможностями. В SQL Server имеются все основные инструменты для работы с базами данных. Имеются дополнительные функции или продукты, такие как средства обеспечения высокой доступности, удаленного аварийного восстановления, секционирования, сжатия данных и прозрачного шифрования данных, инструменты для работы с пространственными данными, управления основными данными и обработки сложных событий, технологии ETL и OLAP, службы интеллектуального анализа данных и подготовки отчетности, инструменты бизнес-аналитики (BI) с поддержкой самообслуживания.

В результате была создана база данных «Банковский учёт» со следующими таблицами:

- Customer

- Histories

- Accounts

- Loans

- OptionLoans

- Deposits

- OptionDeposits

- Currencies

Связи могут быть между двумя (бинарные), тремя (тернарные) и более сущностями. Чаще всего используются бинарные. Они классифицируются следующим образом: Связь «один – к – одному» . Когда каждому экземпляру сущности А соответствует один и только один экземпляр сущности Б, и наоборот. Связь двунаправленная. Связь «один – ко – многим». Это такой тип связи, когда каждому экземпляру сущности А может соответствовать ни одного, один или несколько экземпляров сущности Б, однако каждому экземпляру сущности Б соответствует один и только один экземпляр сущности А. Связь «многие – к – одному». Это отображение обратно предыдущему. Связь «многие – ко – многим». Это такой тип связи, при котором каждому экземпляру сущности А может соответствовать ни одного, один или несколько экземпляров сущности Б, и наоборот. [2]

В данной базе данных используются связи:

«один – к – одному», например одному блюду соответствует один рецепт.

«один – ко – многим» и «многие – к – одному», например такие как Должности <=>> Сотрудники, где на одной должности могут находится разные люди одновременно.

«многие – ко – многим», одно блюдо можно готовить каждый день, и в определенный день готовят разные блюда.

Более подробная информация о базе данных и о связях между таблицами представлена на диаграмме базы данных (см. прил. А).

**4 РАЗРАБОТКА КЛИЕНТСКОЙ ЧАСТИ**

Microsoft Visual Studio — линейка продуктов компании [Microsoft](https://ru.wikipedia.org/wiki/Microsoft), включающих [интегрированную среду разработки](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D0%B3%D1%80%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%B0%D1%8F_%D1%81%D1%80%D0%B5%D0%B4%D0%B0_%D1%80%D0%B0%D0%B7%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D0%B8) программного обеспечения и ряд других инструментальных средств. Данные продукты позволяют разрабатывать как [консольные](https://ru.wikipedia.org/wiki/%D0%A2%D0%B5%D0%BA%D1%81%D1%82%D0%BE%D0%B2%D1%8B%D0%B9_%D0%B8%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81_%D0%BF%D0%BE%D0%BB%D1%8C%D0%B7%D0%BE%D0%B2%D0%B0%D1%82%D0%B5%D0%BB%D1%8F) [приложения](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%B8%D0%BA%D0%BB%D0%B0%D0%B4%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5), так и приложения с [графическим интерфейсом](https://ru.wikipedia.org/wiki/%D0%93%D1%80%D0%B0%D1%84%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B9_%D0%B8%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81_%D0%BF%D0%BE%D0%BB%D1%8C%D0%B7%D0%BE%D0%B2%D0%B0%D1%82%D0%B5%D0%BB%D1%8F), в том числе с поддержкой технологии [Windows Forms](https://ru.wikipedia.org/wiki/Windows_Forms), а также [веб-сайты](https://ru.wikipedia.org/wiki/%D0%A1%D0%B0%D0%B9%D1%82), [веб-приложения](https://ru.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D0%BF%D1%80%D0%B8%D0%BB%D0%BE%D0%B6%D0%B5%D0%BD%D0%B8%D0%B5), [веб-службы](https://ru.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D1%81%D0%BB%D1%83%D0%B6%D0%B1%D0%B0) как в [родном](https://ru.wikipedia.org/wiki/%D0%9C%D0%B0%D1%88%D0%B8%D0%BD%D0%BD%D1%8B%D0%B9_%D0%BA%D0%BE%D0%B4), так и в [управляемом](https://ru.wikipedia.org/wiki/%D0%A3%D0%BF%D1%80%D0%B0%D0%B2%D0%BB%D1%8F%D0%B5%D0%BC%D1%8B%D0%B9_%D0%BA%D0%BE%D0%B4) кодах для всех платформ, поддерживаемых [Windows](https://ru.wikipedia.org/wiki/Windows), [Windows Mobile](https://ru.wikipedia.org/wiki/Windows_Mobile), [Windows CE](https://ru.wikipedia.org/wiki/Windows_CE), [.NET Framework](https://ru.wikipedia.org/wiki/.NET_Framework), [Xbox](https://ru.wikipedia.org/wiki/Xbox), [Windows Phone](https://ru.wikipedia.org/wiki/Windows_Phone) [.NET Compact Framework](https://ru.wikipedia.org/wiki/.NET_Compact_Framework) и [Silverlight](https://ru.wikipedia.org/wiki/Silverlight).

Visual Studio включает в себя [редактор исходного кода](https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D0%B4%D0%B0%D0%BA%D1%82%D0%BE%D1%80_%D0%B8%D1%81%D1%85%D0%BE%D0%B4%D0%BD%D0%BE%D0%B3%D0%BE_%D0%BA%D0%BE%D0%B4%D0%B0) с поддержкой технологии [IntelliSense](https://ru.wikipedia.org/wiki/IntelliSense) и возможностью простейшего [рефакторинга кода](https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D1%84%D0%B0%D0%BA%D1%82%D0%BE%D1%80%D0%B8%D0%BD%D0%B3). Встроенный [отладчик](https://ru.wikipedia.org/wiki/Microsoft_Visual_Studio_Debugger) может работать как отладчик уровня исходного кода, так и как отладчик машинного уровня. Остальные встраиваемые инструменты включают в себя редактор форм для упрощения создания графического интерфейса приложения, веб-редактор, дизайнер [классов](https://ru.wikipedia.org/wiki/%D0%9A%D0%BB%D0%B0%D1%81%D1%81_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)) и дизайнер [схемы базы данных](https://ru.wikipedia.org/wiki/%D0%A1%D1%85%D0%B5%D0%BC%D0%B0_%D0%B1%D0%B0%D0%B7%D1%8B_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85). Visual Studio позволяет создавать и подключать сторонние дополнения ([плагины](https://ru.wikipedia.org/wiki/%D0%9F%D0%BB%D0%B0%D0%B3%D0%B8%D0%BD)) для расширения функциональности практически на каждом уровне, включая добавление поддержки систем [контроля версий исходного кода](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0_%D1%83%D0%BF%D1%80%D0%B0%D0%B2%D0%BB%D0%B5%D0%BD%D0%B8%D1%8F_%D0%B2%D0%B5%D1%80%D1%81%D0%B8%D1%8F%D0%BC%D0%B8) (как например, [Subversion](https://ru.wikipedia.org/wiki/Subversion) и [Visual Source Safe](https://ru.wikipedia.org/wiki/Microsoft_Visual_SourceSafe)), добавление новых наборов инструментов (например, для редактирования и визуального проектирования кода на [предметно-ориентированных языках программирования](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%B5%D0%B4%D0%BC%D0%B5%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D1%8B%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) или инструментов для прочих аспектов [процесса разработки программного обеспечения](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D1%86%D0%B5%D1%81%D1%81_%D1%80%D0%B0%D0%B7%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D0%B8_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B3%D0%BE_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D1%8F) (например, клиент Team Explorer для работы с [Team Foundation Server](https://ru.wikipedia.org/wiki/Team_Foundation_Server)).

ADO.NET — это набор классов, предоставляющих службы доступа к данным программисту, работающему на платформе .NET Framework. ADO.NET имеет богатый набор компонентов для создания распределенных приложений, совместно использующих данные. Это неотъемлемая часть платформы .NET Framework, которая предоставляет доступ к реляционным данным, XML-данным и данным приложений. ADO.NET удовлетворяет различные потребности разработчиков, включая создание клиентских приложений баз данных, а также бизнес-объектов среднего уровня, используемых приложениями, средствами, языками и браузерам. [3]

В результате выбора технологий и способов создания приложения, представленных выше, были созданы следующие формы, где содержатся описание и выполнение всех функций программы:

Главная форма для работы с данными. На данной форме администратор может создать пользователя, счёт, займ или депозит. Посмотреть общее количество созданных клиентов, счетов, займов и депозитов (рисунок 1). Также может осуществить поиск клиента по фамилии или № паспорта (рисунок 2). [Отчеты демонстрируют способ эффективного представления данных в печатной форме. Они созданы для предоставления выдаваемых базой данных сведений в удобном для восприятия виде (рисунок 3).]

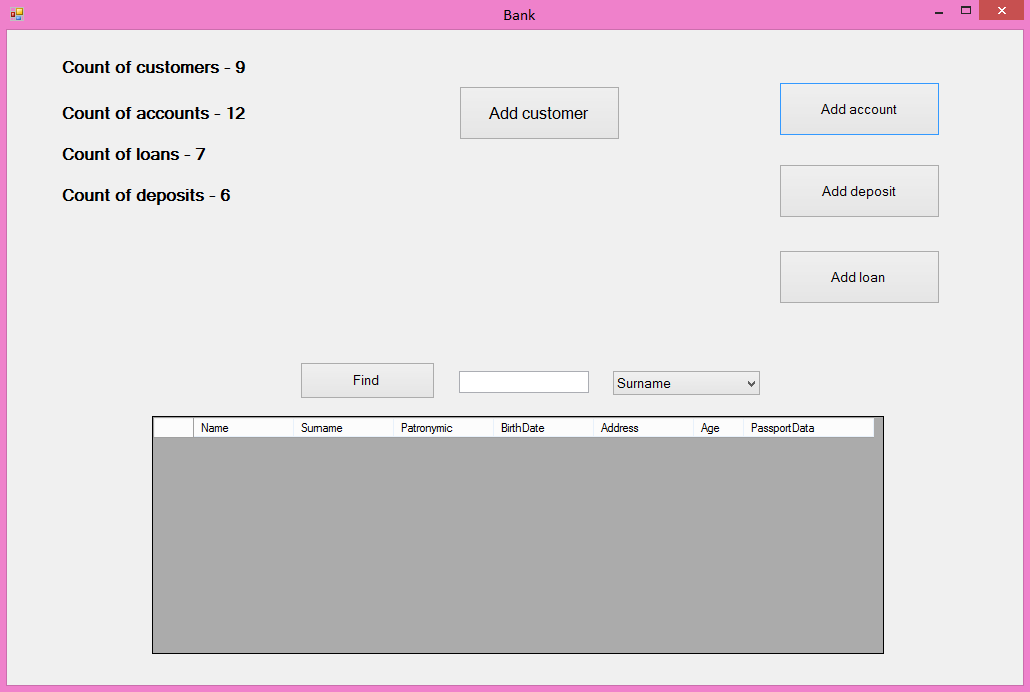


Рисунок 1 – Главная форма

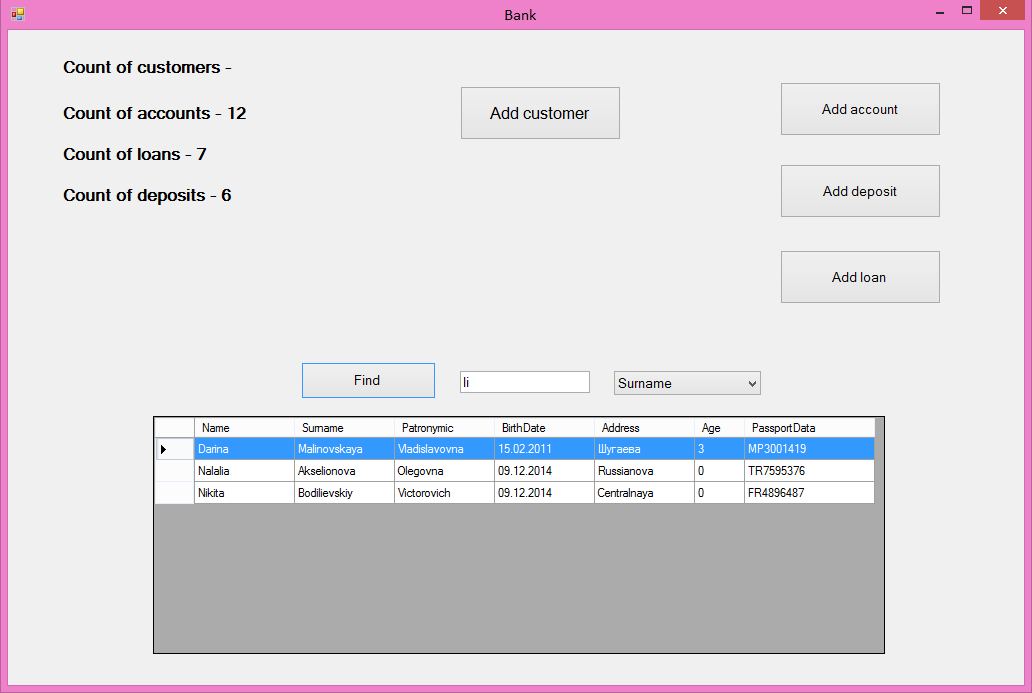


Рисунок 2 – Поиск клиента

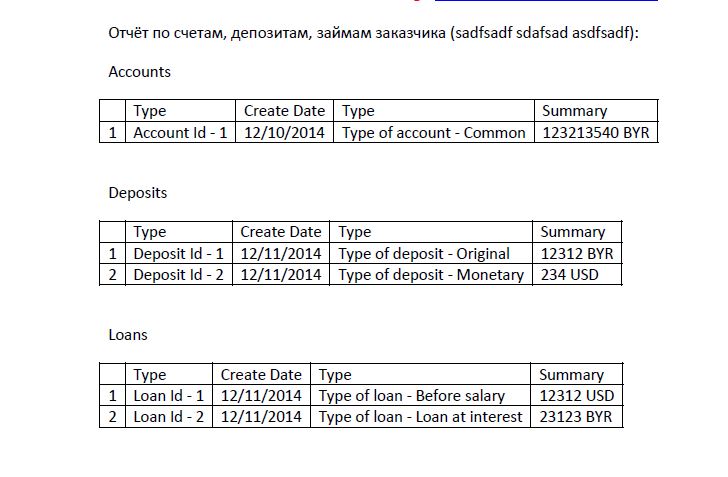
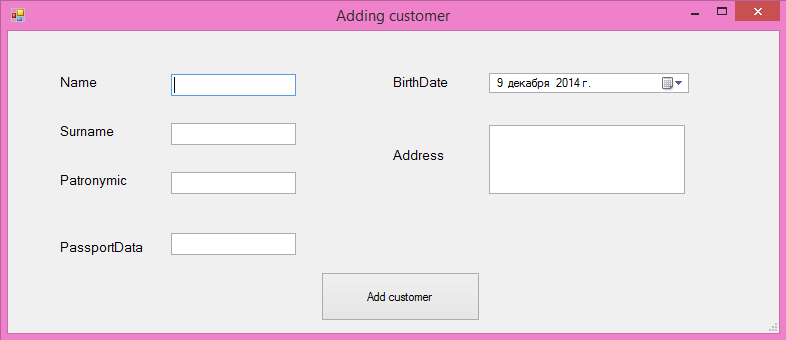
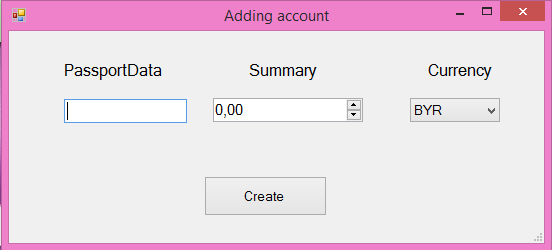


Рисунок 3 – Отчет в печатной форме

Следующие формы предназначены для создания информации БД. Администратор имеет возможность добавлять данные в БД. (рисунок 4, 5, 6, 7).



Рисунок 4 – Форма для добавления клиента

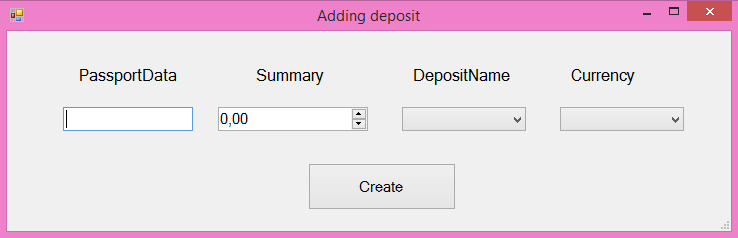
Рисунок 5 – Форма для добавления счёта

Рисунок 6 – Форма для добавления депозита

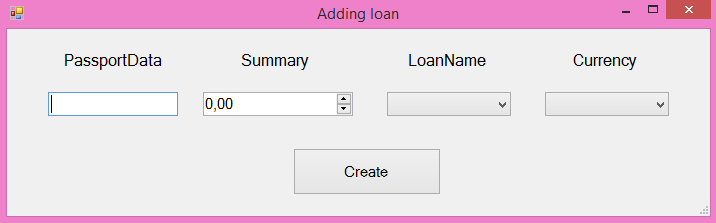


Рисунок 7 – Форма для добавления займа

Форма «Customer» вызывается 2 кликом по клиенту в поиске (рисунок 8). Имеет возможность удалить клиента. Также администратор может добавить счёт (рисунок 9), займ (рисунок 10) и депозит (рисунок 11).

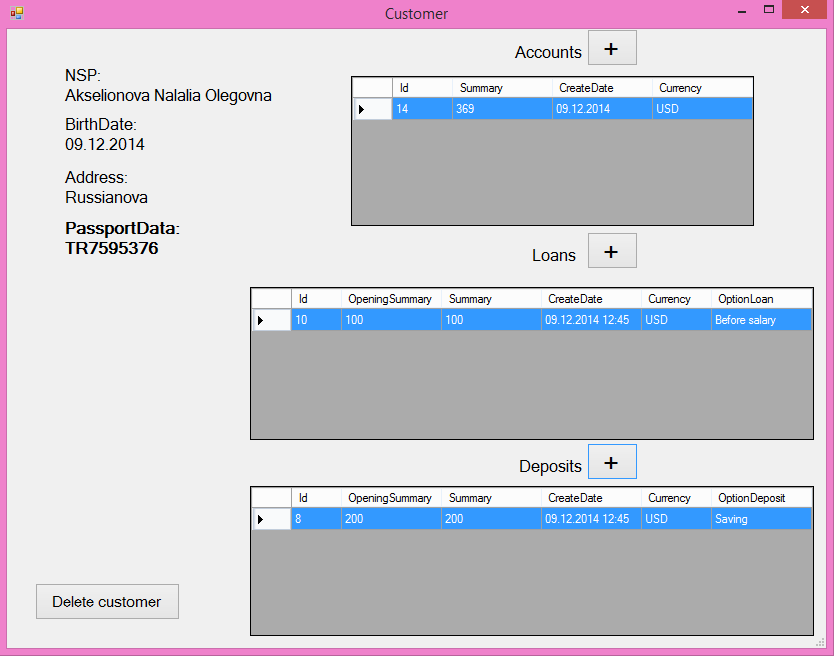


Рисунок 8 – Форма для учета клиента

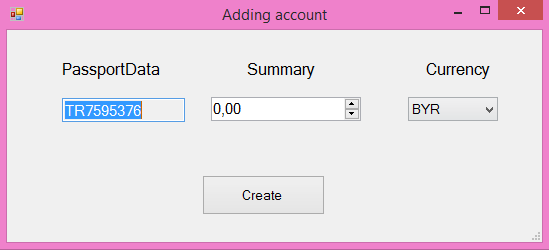


Рисунок 9 – Форма для создания счёта

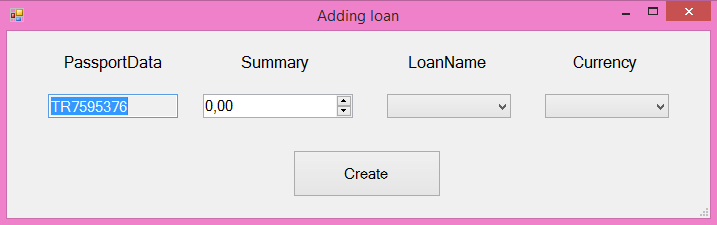


Рисунок 10 – Форма для создания займа

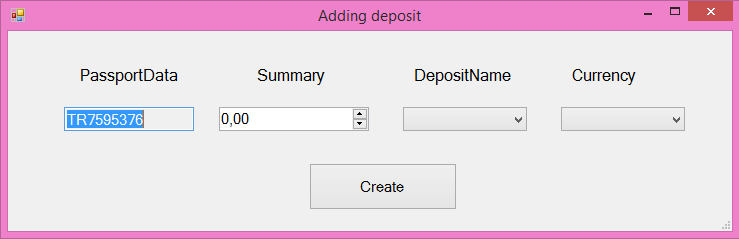


Рисунок 11 – Форма для создания депозита

Форма «Editing account» предназначена для начисления или снятия денежной суммы. Возможна функция обнуления счёта. Реализована функция транзакции денежной суммы с данного счёта на другой. Возможен поиск другого счёта (рисунок 12). Для вызова данной формы необхадимо 2 раза кликнуть на любой счёт из рисунка 8

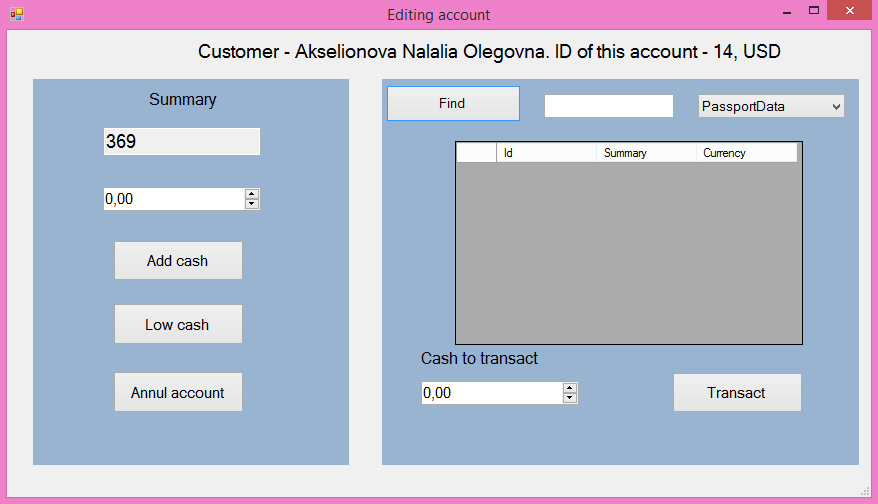


Рисунок 12 – Форма для создания депозита

Последующие формы реализуют задачу подсчёта продентов на займ и депозит за любой промежуток времени. Чтобы вызвать эти формы нужно 2 раза кликнуть на какой-либо займ или депозит из рисунка 8 (рисунок 13,14).

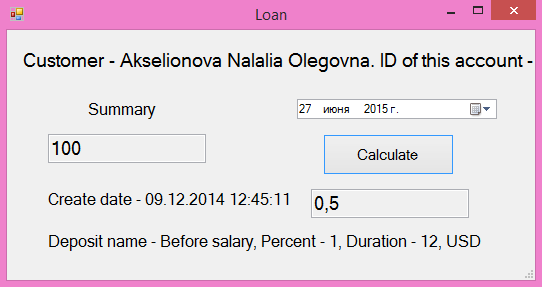


Рисунок 13 – Форма для подсчёта процентов для займа

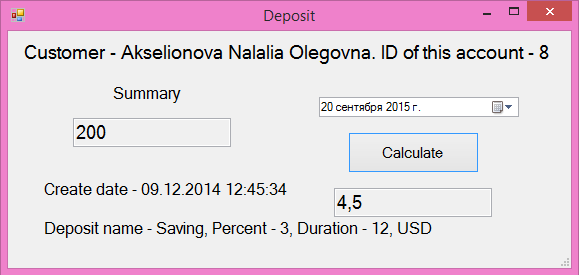


Рисунок 14 – Форма для подсчёта процентов для депозита

5 ПРОВЕРКА РАБОТОСПОСОБНОСТИ РАЗРАБОТАННОЙ ИНФОРМАЦИОННОЙ СИСТЕМЫ

Для проверки работоспособности разработанной системы необходимо выполнить проверку запуска программы, основного функционала, соответствие прототипам, а также реакцию системы на ввод некорректных данных в различные поля. Для проверки функциональных возможностей будет использовано описание требований к основному функционалу системы (раздел 2 «Проектирование структуры информационной системы»). Контрольный лист представлен в таблице 2 - Контрольный лист проверки системы.

Таблица 2 – Контрольный лист проверки системы

|  |  |  |
| --- | --- | --- |
| Полномочие | Описание | Результат тестирования |
| Запуск приложения | Главная форма приложения с выводом информации об общем количестве клиентов, счетов, депозитов, с кнопками создания клиента, счёта, займа и депозита, также с кнопкой поиска клиентов | Выполнено |
| Поиск | Вывод всех клиентов и поиск клиента по фамилии или паспортным данным | Выполнено |
| Двойной клик по клиенту | Переход на форму “Customer” | Выполнено |
| Клик на кнопки | Формs для создания соответсвующей операции | Выполнено |
| Печать | Данные о займе/депозите выгружаются в .txt файл и автоматически открываются для просмотра | Выполнено |
| Форма «Customer» | Содержит всю информация о клиенте, кнопку удаления клиента, создание счёта, займа и депозита клиента | Выполнено |
| Двойной клик по созданному счёту | Открытие формы “Editing account” | Выполнено |
| Клик на кнопки (+) | Создание соответсвующей операции для данного клиента | Выполнено |
| Удалить | Клиент удаляется из БД | Выполнено |
| Двойной клик по созданному займу | Открытие формы для подсчёта процентов на займ за любой промежуток времени | Выполнено |

Продолжение Таблицы 2

|  |  |  |
| --- | --- | --- |
| Полномочие | Описание | Результат тестирования |
| Двойной клик по созданному депозиту | Открытие формы для подсчёта процентов на депозит за любой промежуток времени | Выполнено |
| Форма “Editing account” | Имеет два поля: «Summary» и «Cash to transact» | Выполнено |
| Поле «Summary» | Начисление/снятие денежной суммы со счёта, её обнуление | Выполнено |
| Поле «Cash to transact» | Транзакция с данного счёта на другой. Поиск другого счёта | Выполнено |

Headline

UIBunk -> Customer -> Accounts: возможно начислить сумму = 0 на счёт клиента

Description

1. Зайти в UIBank
2. Создать клиента, нажав кнопку «Add customer»
3. Нажать кнопку «Find»
4. 2 раза кликнуть на клиента, созданного в п.2
5. Нажать на «Accounts»
6. Нажать кнопку «Create»

Actual Result

Счёт добавлен

Expected Result

Сообщение об ошибке «Сумма счёта не должна быть ниже 50»

Добавление счёта невозможно

ЗАКЛЮЧЕНИЕ

В результате курсовой работы создана программа «Банковский учёт». Программа позволяет осуществлять просмотр счетов, займов и депозитов, которые взял у банка пользователь. Так же имеется возможность добавлять, изменять, удалять счета, аккаунты, депозиты и личные данные пользователя, просматривать отчёты по счетам пользователя. Вести учет всех банковских ячеек, принадлежащих банку. Данные загружаются из базы данных MS SQL Server 2012 с использованием технологии Entity Framework. Программный продукт выполнен в среде Microsoft Visual Studio 2012.

На примере базы данных «Банковский учёт» мы познакомились с инструментом разработки баз данных MS SQL Server. С его помощью можно быстро создавать приложения для различных сфер деятельности человека.

Основной целью информационных систем с базой данных является поиск или учет информации по определенным параметрам, пополнение таблиц новыми данными, просмтотр результатов полученной информации в текстовом или табличном видах, анализ доступной информации и прогнозирование. Информационная система «Банковский учёт» соответсвует всем перечисленным параметрам.

**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

[1] Википедия [Электронный ресурс]. – Режим доступа : <https://ru.wikipedia.org/wiki>.

[2] Работа с БД на языке C#. Технология ADO .NET: учебное пособие/ сост. О.Н. Евсеева, А. Б. Шамшев. – Ульяновск, 2009. – 170 с.

[3] Основы ADO.NET [Электронный ресурс]. – Режим доступа : <http://citforum.ru/programming/csharp/ado_dot_net/>.

ПРИЛОЖЕНИЕ А

(обязательное)

Диаграмма базы данных «Банковский учёт»

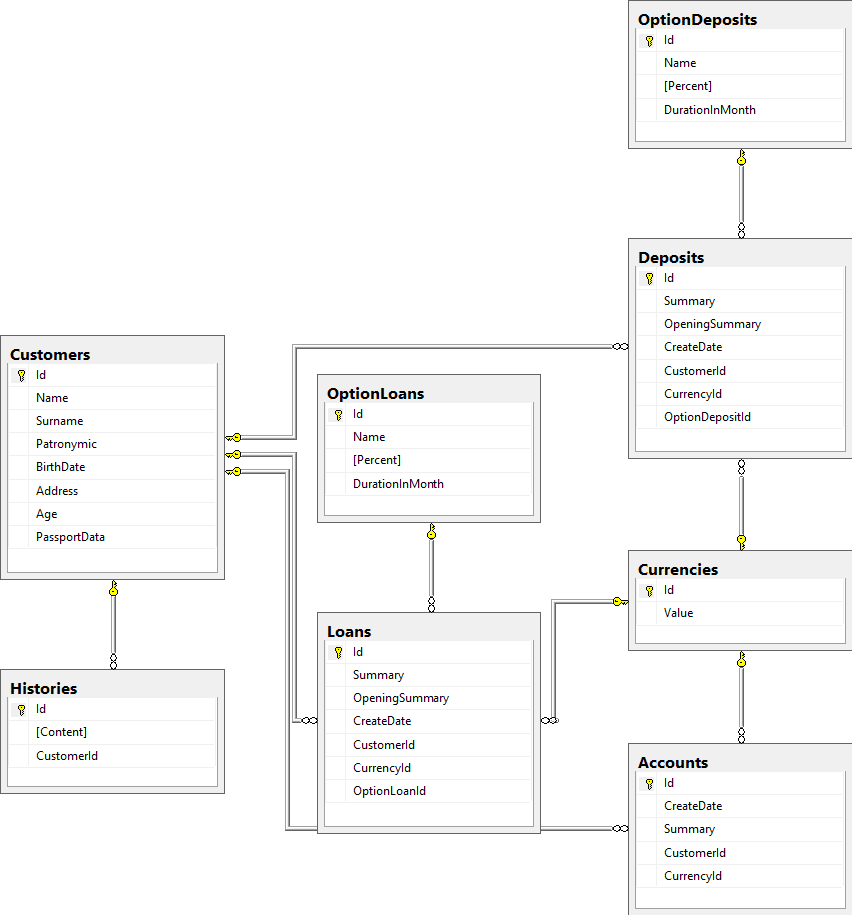


Рисунок А.1 – Диаграмма базы данных «Банковский учёт»

ПРИЛОЖЕНИЕ Б

(обязательное)

Листинг программы

// Листинг программы - сервис для работы со счетами

using System;

using System.Collections.Generic;

using System.Linq;

using CourseProject.Core;

using CourseProject.Core.Entities;

using CourseProject.Core.Exceptions;

using CourseProject.Infrastructure.Guard.Validation;

using CourseProject.Services.Exceptions;

namespace CourseProject.Services.Services

{

    public class AccountService : IService

    {

        private readonly IUnitOfWork \_unitOfWork;

        private readonly IRepositoryFactory \_factoryOfRepositories;

        public AccountService(IUnitOfWork unitOfWork, IRepositoryFactory factoryOfRepositories)

        {

            Guard.AgainstNullReference(unitOfWork, "unitOfWork");

            Guard.AgainstNullReference(factoryOfRepositories, "factoryOfRepositories");

            \_unitOfWork = unitOfWork;

            \_factoryOfRepositories = factoryOfRepositories;

        }

        public Account CreateAccount(double summary, int customerId, int currencyId)

        {

            Guard.AgainstNegatives((int) summary);

            var account = new Account

            {

                CreateDate = DateTime.Now,

                Summary = summary,

                CustomerId = customerId,

                CurrencyId = currencyId

            };

            var accountRepository = \_factoryOfRepositories.GetAccountRepository();

            accountRepository.Create(account);

            try

            {

                \_unitOfWork.PreSave();

            }

            catch (Exception exception)

            {

                throw new ServiceException(exception);

            }

            return account;

        }

        public void UpdateAccount(Account account)

        {

            var accountRepository = \_factoryOfRepositories.GetAccountRepository();

            try

            {

                accountRepository.Update(account);

            }

            catch (Exception e)

            {

                throw new AccountServiceException(e);

            }

        }

        public void RemoveAccount(Account account)

        {

            var accountRepository = \_factoryOfRepositories.GetAccountRepository();

            accountRepository.Remove(account);

        }

        public Account GetAccountById(int accountId)

        {

            var accountRepository = \_factoryOfRepositories.GetAccountRepository();

            try

            {

                return accountRepository.GetEntityById(accountId);

            }

            catch (RepositoryException e)

            {

                throw new AccountServiceException(e.Message);

            }

        }

        public List<Account> GetAccountsByPassportData(string passportData)

        {

            var accountRepository = \_factoryOfRepositories.GetAccountRepository();

            return accountRepository.Filter(e => e.Customer.PassportData == passportData).ToList();

        }

        public List<Account> GetAccountByCustomerSurname(string surname)

        {

            var accountRepository = \_factoryOfRepositories.GetAccountRepository();

            return accountRepository.Filter(e => e.Customer.Surname.Contains(surname)).ToList();

        }

        public List<Account> GetAccountsByCustomerId(int customerId)

        {

            var accountRepository = \_factoryOfRepositories.GetAccountRepository();

            return accountRepository.Filter(e => e.CustomerId == customerId).ToList();

        }

        public int GetCountOfAllAccounts()

        {

            var accountRepository = \_factoryOfRepositories.GetAccountRepository();

            return accountRepository.All().Count();

        }

        public IQueryable<Account> GetAllAccounts()

        {

            var accountRepository = \_factoryOfRepositories.GetAccountRepository();

            return accountRepository.All();

        }

    }

}