Оглавление

[ГЛОССАРИЙ 2](#_Toc105638335)

[Введение 4](#_Toc105638336)

[ГЛАВА1. ОБОСНОВАНИЕ ОБЪЕКТА АВТОМАТИЗАЦИИ. 5](#_Toc105638337)

[1.1. Информационной системе 5](#_Toc105638338)

[1.2. Общие требования к информационной системе 5](#_Toc105638339)

[1.3. Разработка технического задания 6](#_Toc105638340)

[1.4 Обоснование необходимости внедрения и использования информационной системы 9](#_Toc105638341)

[ГЛАВА 2. РЕАЛИЗАЦИЯ БАЗЫ ДАННЫХ ИНФОРМАЦИОННОЙ СИСТЕМЫ. 10](#_Toc105638342)

[2.1 Физическая структура базы данных. 12](#_Toc105638343)

[2.2 Запросы 14](#_Toc105638344)

[ГЛАВА 3. РАЗРАБОТКА КОДА ИС 16](#_Toc105638345)

[3.1. Что такое разработка 16](#_Toc105638346)

[ГЛАВА 4. ТЕСТИРОВАНИЕ 25](#_Toc105638347)

[4.1 Тест- план 26](#_Toc105638348)

[1.5 Написание текст-кейсов 26](#_Toc105638349)

[1.6 Юнит- тестирование 28](#_Toc105638350)

[Заключение 30](#_Toc105638351)

[Список источников 31](#_Toc105638352)

[Приложение №1 32](#_Toc105638353)

# ГЛОССАРИЙ

**Банк –** кредитная организация, имеющая исключительное право осуществлять в совокупности банковские операции: по привлечению во вклады денежных средств физических и юридических лиц; размещение указанных средств от своего имения за свой счет на условиях возвратности, платности, срочности; открытие и ведение банковских счетов физических и юридических лиц.

**Заемщик -** сторона кредитных отношений, получающая кредит, и обязанная возвратить полученную ссуду и уплатить ссудный процент

**Коммерческий банк** – вид банка, кредитная организация, специализирующаяся на предоставлении банковских услуг. Как правило, коммерческие банки универсальные, обслуживают как юридических, так и физических лиц.

**Кредит –** ссуда в денежной или товарной форме на условиях срочности, возвратности, платности.

**Кредитование** – это финансовые взаимоотношения, в которых одна сторона – кредитодатель, предоставляет во временное пользование ссуду в денежной или натуральной форме, а вторая сторона–заемщик, пользуется предоставленной ссудой на условиях возвратности и возмездности. Кредитование, как явление возникло вместе с развитием товарно-денежных отношений не только в России, но и во всех других странах. Разница состоит только в наименованиях сторон кредитования (ростовщик, кооператив, ломбард, банк) и в формальностях, которые сопровождают процедуру кредитования.

**Платежеспособность клиента**—это его возможность и способность своевременно погасить все виды обязательств и задолженности.

**Платежеспособность** – это способность клиентов расплачиваться по собственным обязательствам в полном объёме в установленные сроки.

**Срочный кредит-** кредит на срок не более 90 дней, обычно оформляемый в виде простого векселя, подписанного заемщиком.

**Техническое задание** — документ или несколько документов, определяющих цель, структуру, свойства и методы какого-либо проекта, и исключающие двусмысленное толкование различными исполнителями. **Техническое задание** используется в машиностроении, производстве и бизнесе для того, чтобы поставщики, покупатели и пользователи материалов, продуктов или услуг понимали и согласовывали все требования.

# Введение

Понятие ИС связано с компьютерными технологиями и поэтому для большинства людей ассоциируется с чем-то непостижимым. Но на деле все не так сложно, как кажется.

Информационная система представляет собой хранилище информации, снабженное процедурами ввода, поиска, размещения и выдачи информации. Наличие таких процедур – главная особенность информационных систем, отличающих их от простых скоплений информационных материалов.

Современное понимание информационной системы предполагает использование в качестве основного технического средства переработки информации персонального компьютера. Кроме того, техническое воплощение информационной системы само по себе ничего не будет значить, если не учтена роль человека, для которого предназначена производимая информация и без которого невозможно ее получение и представление.

Существует несколько видов классификаций информационных систем.

По масштабности они подразделяются на следующие категории:

* Индивидуальные. Такие системы создаются только для одного человека. Они содержат вводимые самим пользователем персональные данные. В качестве примера приведем блокнот или записную книгу.
* Коллективная. Эта система предназначена для небольшой группы людей. Вся информация, которую она содержит, может быть использована только ими.
* Корпоративная. В этом случае имеем дело с ИС, рассчитанной как на мелкие, так и на крупные предприятия.

# ГЛАВА1. ОБОСНОВАНИЕ ОБЪЕКТА АВТОМАТИЗАЦИИ.

В данной главе описаны требования, цели и задачи ИС, и разработка технического задания для ИС, так же представлена БД.

# Информационная система

Рассмотрим информационную систему – банк, а точнее кредитование клиента, с процессом исследования его платежеспособности и одобрение выплаты ему кредита.

|  |  |
| --- | --- |
| **Название ИС** | **SouthBank** |
| Цель ИС (с точки зрения заказчика) | Ускорить работу отдела кредитования |
| Задачи ИС (с точки зрения заказчика) | Заполнение анкет для кредитования |

Данная информационная система нужна для автоматизации оформления анкет для кредитования.

# Общие требования к информационной системе

Данная информационная область будет иметь следующий функционал:

1. Пользователь заполняет анкету на одобрение кредита (Клиент должен заполнить свое ФИО, номер телефона, указать: постоянное место работы, размер постоянной заработной платы, цель кредита, адрес места жительства, размер кредита)
2. Банк обрабатывает анкету клиента и либо одобряет, либо отказывает в кредите клиенту
3. Если банк все одобрил, то вместе с клиентом подбирают удобный срок кредита и размер ежемесячной выплаты.
4. Если же банк не одобрил клиенту кредит, то он его оповестит об этом и тот подбирает себе другой банк, где ему одобрят кредит на нужную сумму.

Так же ИС будет иметь следующее визуальное представление:



Рис. 1 Визуальное представление ИС

Нефункциональные требования к информационной системе, описанные в ТЗ, на практике должны отвечать следующим критериям:

• единичность. Одна задача касается одного элемента системы;

• завершенность. После реализации ТЗ система не должна требовать доработок;

• последовательность. Поставленные задачи должны быть реализуемыми и не противоречащими друг другу;

• актуальность. Система не должна требовать обновления сразу после внедрения;

• выполнимость. Нерешаемые задачи не должны попасть в ТЗ;

• обязательность. Если существуют требования к ИС, обусловленные бизнес-процессами или действующим законодательством, они должны быть учтены;

• проверяемость. Решение задач должно быть проверяемо в процессе аудита.

## Разработка технического задания

ТЗ – основополагающий документ, которым руководствуются разработчики и проектировщики, приступая к разработке нового изделия. Оно определяет основные направления разработки: конструкции и принципа работы будущего изделия. ТЗ заявляет, с одной стороны, о потребностях общества в новых изделиях, с другой – о технических и технико-экономических характеристиках изделия.

Техническое задание является начальным этапом работ и составляется на все разработки и виды работ, необходимые для создания нового изделия. Оно может предшествовать научно-исследовательским и опытно-конструкторским работам (НИОКР) по разработке средств механизации и автоматизации, отдельных узлов и систем, технологии, измерительных средств, средств контроля и других изделий (выполнение работы, оказание услуги, промышленный комплекс, прибор, машина, аппарат, система управления, информационная система, нормативная документация (например, стандарт) и т. д.).

Требования, включаемые в ТЗ, должны основываться на современных достижениях науки и техники, на итогах выполненных научно-исследовательских и экспериментальных работ. ТЗ должно устанавливать следующие показатели разрабатываемого изделия:

- основное назначение, технические и тактико-технические характеристики, уровень стандартизации и унификации;

- технико-экономические показатели;

- патентно-правовые показатели;

- специальные требования к изделию и др.

В технических заданиях оговариваются этапы разработки и сроки выполнения каждого этапа, сроки разработки в целом. Качество ТЗ обеспечивается объемом и полнотой сбора материалов, необходимых для разработки. При разработке используются следующие материалы:

- научно-техническая информация;

- патентная информация;

- характеристика рынка сбыта;

- характеристика производства, на котором изделие будет изготавливаться (технологическая оснащенность, квалификация кадров, технологическая дисциплина, уровень организации труда и др.).

При разработке ТЗ разработчик учитывает информацию об аналогичной продукции, содержащуюся в базах данных (общероссийской и региональных), созданных в Госстандарте России на основе каталожных листов продукции.

Техническое задание разрабатывается, как правило, организацией-разработчиком изделия. Сформулировать задачу максимально полно и грамотно, обосновать необходимость её решения – главная цель ТЗ. Исполнитель выполняет его в контакте с заказчиком. Обязанность заказчика – предъявить разработчику исходные данные для разработки изделия.

ТЗ разрабатывают и утверждают в порядке, установленном заказчиком и разработчиком. К разработке ТЗ могут привлекаться другие заинтересованные организации (предприятия): изготовитель, торговая (посредническая) организация, страховая организация, организация-проектировщик, монтажная организация и др.

Для подтверждения отдельных требований к продукции, в том числе требований безопасности, охраны здоровья и окружающей среды, а также оценки технического уровня продукции, ТЗ может быть направлено разработчиком или заказчиком на экспертизу (заключение) в сторонние организации. Решение по полученным заключениям принимают разработчик и заказчик до утверждения ТЗ.

К техническому заданию прилагаются схемы и эскизы по конструкции будущего изделия, а для технологических разработок – технологические и технико-экономические показатели существующего производства. Техническое задание должно содержать максимум информации, облегчающей работу над изделием и сокращающей сроки разработки.

Несмотря на всю свою важность, содержание ТЗ мало регламентировано нормативными документами. Требования к содержанию и порядку построения документа зависят от направления разработки и содержатся в следующих документах:

**ГОСТ Р 15.201-2000.** Система разработки и постановки продукции на производство (СРПП). Продукция производственно-технического назначения. Порядок разработки и постановки продукции на производство (приведены общие требования и краткие рекомендации по разработке).

**ГОСТ 19.201-78.** Единая система программной документации. Техническое задание. Требования к содержанию и оформлению (кратко изложено содержание ТЗ);

**ГОСТ 34.602-89.** Информационная технология. Комплекс стандартов на автоматизированные системы. Техническое задание на создание автоматизированной системы (достаточно подробно изложены состав и содержание ТЗ);

**ГОСТ 25123-82.** Машины вычислительные и системы обработки данных. Техническое задание. Порядок построения, изложения и оформления (приведен порядок построения ТЗ). Обобщая требования этих стандартов, порядок построения, изложения и оформления ТЗ можно свести к последовательности.

На основании анализа, было разработано техническое задание для представленной ИС см. Приложение №1»

## Обоснование необходимости внедрения и использования информационной системы

Данное направление автоматизации было выбрано для того, чтобы люди могли в удобное для себя время оформить кредит на нужную им сумму и срок. Поэтому для этого было выбрано настольное приложение.

# ГЛАВА 2. РЕАЛИЗАЦИЯ БАЗЫ ДАННЫХ ИНФОРМАЦИОННОЙ СИСТЕМЫ.

Система управления базами данных (сокращенно **СУБД**) – это программное обеспечение для создания и работы с базами данных.

**Общие этапы проектирования БД:**

**Этап 1. Уточнение задач**

На первом этапе составляется список всех основных задач, которые в принципе должны решаться этим приложением, – включая и те, которые не нужны сегодня, но могут появиться в будущем. Под «основными» задачами понимаются функции, которые должны быть представлены в формах или отчетах приложения.

**Этап 2. Последовательность выполнения задач**

Для того, чтобы приложение работало логично и удобно, лучше всего объединить основные задачи в тематические группы и затем упорядочить задачи каждой группы так, чтобы они располагались в порядке их выполнения. Может получиться так, что некоторые задачи будут связаны с разными группами или, что выполнение некоторой задачи должно предшествовать выполнению другой, принадлежащей к иной группе.

**Этап 3. Анализ данных**

После формирования списка задач, наиболее важным этапом является составление подробного перечня всех данных, необходимых для решения каждой задачи. Некоторые данные понадобятся в качестве исходных и меняться не будут. Другие данные будут проверяться и изменяться в ходе выполнения задачи. Некоторые элементы данных могут быть удалены или добавлены. И наконец, некоторые данные будут получены с помощью вычислений: их вывод будет частью задачи, но в базу данных вноситься они не будут.

**Этап 4. Определение структуры данных**

После предварительного анализа всех необходимых элементов, данных нужно упорядочить их по объектам и соотнести объекты с таблицами и запросами базы данных. Для реляционных баз данных типа Access используется процесс, называемый нормализацией, в результате которого вырабатывается наиболее эффективный и гибкий способ хранения данных.

**Этап 5. Разработка макета приложения и пользовательского интерфейса**

После задания структуры таблиц приложения, в Microsoft Access легко создать его макет с помощью форм и связать их между собой, используя несложные макросы или процедуры обработки событий. Предварительный рабочий макет легко продемонстрировать заказчику и получить его одобрение еще до детальной реализации задач приложения.

**Этап 6. Создание приложения**

В случае очень простых задач созданный макет является практически законченным приложением. Однако довольно часто приходится писать процедуры, позволяющие полностью автоматизировать решение всех намеченных в проекте задач. Поэтому, понадобится создать специальные связующие формы, которые обеспечивают переход от одной задачи к другой.

**Этап 7. Тестирование и усовершенствование**

После завершения работ по отдельным компонентам приложения необходимо проверить функционирование приложения в каждом из возможных режимов. Необходимо проверить работу макросов, для этого использовав пошаговый режим отладки, при котором будет выполняться одна конкретная макрокоманда. При использовании Visual Basic для приложений в вашем распоряжении имеются разнообразные средства отладки, позволяющие проверить работу приложения, выявить и исправить ошибки.

По мере разработки автономных разделов приложения желательно передать их заказчику для проверки их функционирования и получения мнения о необходимости внесения тех или иных изменений. После того как заказчик ознакомится с работой приложения, у него практически всегда возникают дополнительные предложения по усовершенствованию, какой бы тщательной не была предварительная проработка проекта. Пользователи часто обнаруживают, что некоторые моменты, о которых в процессе постановки задач, они говорили, как об очень важных и необходимых, на самом деле не играют существенной роли при практическом использовании приложения. Выявление необходимых изменений на ранних стадиях разработки приложения позволяет существенно сократить время на последующие переделки.

## 2.1 Физическая структура базы данных.

**Основные этапы проектирования баз данных**

* Концептуальное (инфологическое) проектирование
* Логическое (дата логическое) проектирование
* Физическое проектирование

ER-модель (Entity-relationship model или Entity-relationship diagram) – это семантическая модель данных, которая предназначена для упрощения процесса проектирования базы данных. Из ER-модели могут быть порождены все виды баз данных: реляционные, иерархические, сетевые, объектные. В основе ER-модели лежат понятия «сущность», «связь» и «атрибут».

Для больших баз данных построение ER-модели позволяет избежать ошибок проектирования, которые чрезвычайно сложно исправлять, в особенности, если база данных уже эксплуатируется или на стадии тестирования. Ошибки в разработке структуры базы данных могут привести к переделке кода программного обеспечения управляющего этой базой данных. В результате время, средства и человеческие ресурсы будут использованы неэффективно.

ER-модель – это представление базы данных в виде наглядных графических диаграмм. ER-модель визуализирует процесс, который определяет некоторую предметную область. Диаграмма «сущность» - «связь» – это диаграмма, которая представляет в графическом виде сущности, атрибуты и связи.

ER-модель – это только концептуальный уровень моделирования. ER-модель не содержит деталей реализации. Для той же самой ER-модели детали ее реализации могут отличаться.

Сущность в базе данных – это любой объект в базе данных, который можно выделить исходя из сути предметной области для которой разрабатывается эта база данных. Разработчик базы данных должен уметь правильно определять сущности.

Каждый тип сущности имеет определенный набор атрибутов. Атрибуты предназначены для описания конкретной сущности.

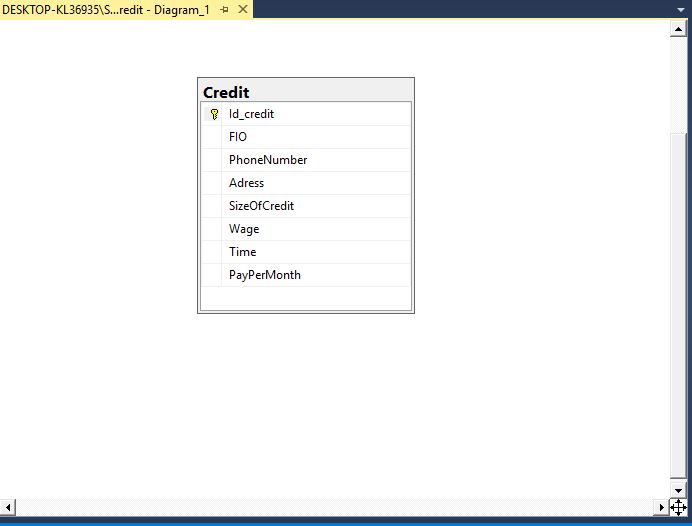


Рис. 2.1 Это логическая структура БД ИС

Описание сущностей в модели:

* Bank- включает в себя наименование банка и его идентификатор;
* Bank employees – сотрудник банка, включает в себя идентификатор сотрудника, его ФИО и должность;
* Bank client – клиент банка, со своими личными данными и идентификатором;
* Credit – кредит, с датой оформления, суммой кредита, идентификатором и типом.

## 2.2 Запросы

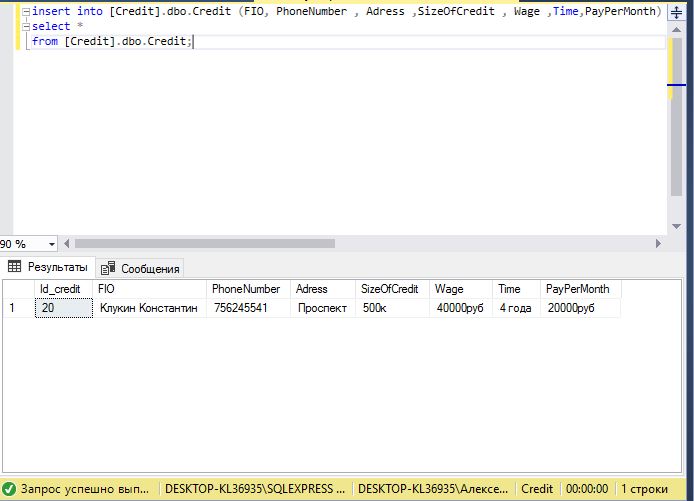


Рис 2.2 Запрос добавления

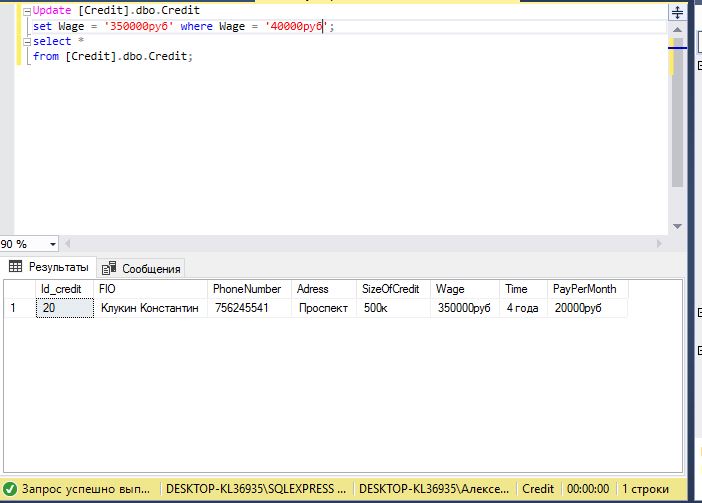


Рис 2.3 Запрос обновления

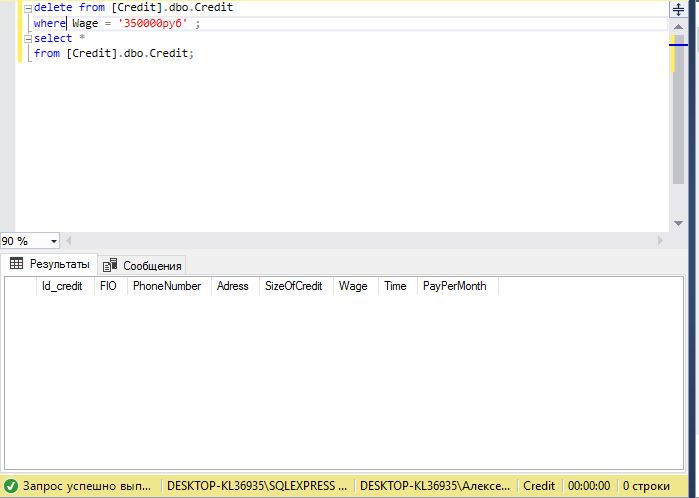


Рис 2.4 Запрос удаления

Таким образом, в данной главе были разобраны этапы разработки БД, была спроектирована БД для ИС. Кроме этого были прописаны запросы к БД.

# ГЛАВА 3. РАЗРАБОТКА КОДА ИС

Разработка программного обеспечения — деятельность по созданию нового программного обеспечения. Разработка программного обеспечения как инженерная дисциплина является составной частью программной инженерии, наряду с дисциплинами, отвечающими за функционирование и сопровождение программных продуктов.

Процесс разработки состоит из множества подпроцессов, или дисциплин, некоторые из которых перечислены ниже. Процесс — совокупность взаимосвязанных или взаимодействующих видов деятельности, преобразующих входы в выходы.

## 3.1. Что такое разработка

Для разработки информационной системы используется C#. Рассмотрим общие этапы разработки программы:

* Определение проблемы.
* Выработка требований.
* Создание плана разработки.
* Разработка архитектуры.
* Детальное проектирование.
* Кодирование и отладка.
* Тестирование компонентов.
* Интеграция компонентов.
* Тестирование всей системы.
* Сопровождение, внесение поправок и оптимизация.

Основные окна приложения:

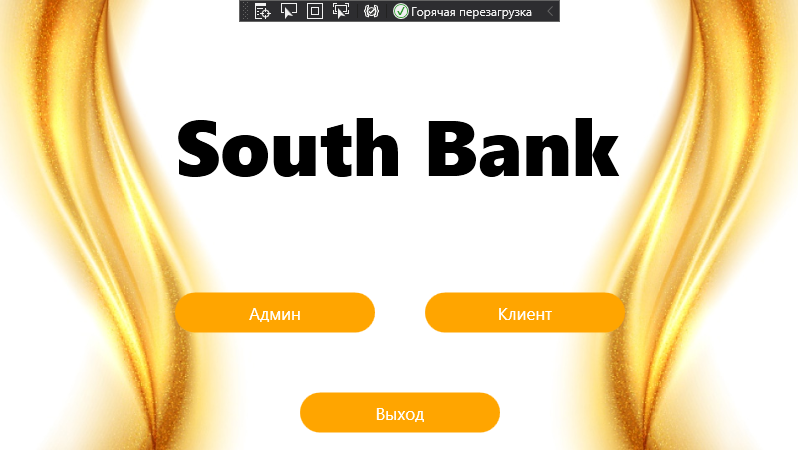


Рис 3. Главное окно

**Код xaml разметки.**

<Grid >

<Grid.RowDefinitions>

<RowDefinition></RowDefinition>

<RowDefinition></RowDefinition>

</Grid.RowDefinitions>

<Grid.Background>

<ImageBrush ImageSource="Back.jpg"></ImageBrush>

</Grid.Background>

<TextBlock Text="South Bank" FontSize="79" FontWeight="Black" HorizontalAlignment="Center" VerticalAlignment="Bottom" Margin="171,0,171,29" Width="450"></TextBlock>

<Button x:Name="btnClient"

Width="200"

Height="40"

VerticalAlignment="Center"

HorizontalAlignment="Center"

Background="#ff1d222c"

Foreground="White"

Content="Клиент"

FontSize="17"

Grid.Row="1"

Margin="250,0,0,50"

Click="btnClient\_click">

<Button.Style>

<Style TargetType="Button" >

<Setter Property="Foreground" Value="#a0a0a0"/>

<Setter Property="Template">

<Setter.Value>

<ControlTemplate TargetType="{x:Type Button}">

<Border x:Name="ButtonBackground" BorderBrush="Gray">

<ContentPresenter VerticalAlignment="Center" HorizontalAlignment="Center"/>

<Border.BorderThickness>0</Border.BorderThickness>

<Border.CornerRadius>20</Border.CornerRadius>

<Border.Background>

<LinearGradientBrush EndPoint="0.5,1" StartPoint="0.5,0">

<GradientStop Color="Orange" Offset="0" />

<GradientStop Color="Orange" Offset="1" />

</LinearGradientBrush>

</Border.Background>

</Border>

<ControlTemplate.Triggers>

<Trigger Property="IsMouseOver" Value="True">

<Setter TargetName="ButtonBackground" Property="Background">

<Setter.Value>

<LinearGradientBrush StartPoint="0.5,0" EndPoint="0.5,1" >

<GradientStop Color="#ff232d3e" Offset="0" />

<GradientStop Color="#ff232d3e" Offset="0.75" />

<GradientStop Color="#ff232d3e" Offset="0.75" />

<GradientStop Color="#ff232d3e" Offset="1" />

</LinearGradientBrush>

</Setter.Value>

</Setter>

</Trigger>

</ControlTemplate.Triggers>

</ControlTemplate>

</Setter.Value>

</Setter>

</Style>

</Button.Style>

</Button>

<Button x:Name="btnAdmin"

Width="200"

Height="40"

VerticalAlignment="Center"

HorizontalAlignment="Center"

Background="#ff1d222c"

Foreground="White"

Content="Админ"

FontSize="17"

Grid.Row="1"

Margin="0,0,250,50"

Click="btnAdmin\_click">

<Button.Style>

<Style TargetType="Button" >

<Setter Property="Foreground" Value="#a0a0a0"/>

<Setter Property="Template">

<Setter.Value>

<ControlTemplate TargetType="{x:Type Button}">

<Border x:Name="ButtonBackground" BorderBrush="Gray">

<ContentPresenter VerticalAlignment="Center" HorizontalAlignment="Center"/>

<Border.BorderThickness>0</Border.BorderThickness>

<Border.CornerRadius>20</Border.CornerRadius>

<Border.Background>

<LinearGradientBrush EndPoint="0.5,1" StartPoint="0.5,0">

<GradientStop Color="Orange" Offset="0" />

<GradientStop Color="Orange" Offset="1" />

</LinearGradientBrush>

</Border.Background>

</Border>

<ControlTemplate.Triggers>

<Trigger Property="IsMouseOver" Value="True">

<Setter TargetName="ButtonBackground" Property="Background">

<Setter.Value>

<LinearGradientBrush StartPoint="0.5,0" EndPoint="0.5,1" >

<GradientStop Color="#ff232d3e" Offset="0" />

<GradientStop Color="#ff232d3e" Offset="0.75" />

<GradientStop Color="#ff232d3e" Offset="0.75" />

<GradientStop Color="#ff232d3e" Offset="1" />

</LinearGradientBrush>

</Setter.Value>

</Setter>

</Trigger>

</ControlTemplate.Triggers>

</ControlTemplate>

</Setter.Value>

</Setter>

</Style>

</Button.Style>

</Button>

<Button x:Name="btnClose"

Width="200"

Height="40"

VerticalAlignment="Center"

HorizontalAlignment="Center"

Background="#ff1d222c"

Foreground="White"

Content="Выход"

FontSize="17"

Grid.Row="1"

Margin="0,150,0,0"

Click="btnClose\_click">

<Button.Style>

<Style TargetType="Button" >

<Setter Property="Foreground" Value="#a0a0a0"/>

<Setter Property="Template">

<Setter.Value>

<ControlTemplate TargetType="{x:Type Button}">

<Border x:Name="ButtonBackground" BorderBrush="Gray">

<ContentPresenter VerticalAlignment="Center" HorizontalAlignment="Center"/>

<Border.BorderThickness>0</Border.BorderThickness>

<Border.CornerRadius>20</Border.CornerRadius>

<Border.Background>

<LinearGradientBrush EndPoint="0.5,1" StartPoint="0.5,0">

<GradientStop Color="Orange" Offset="0" />

<GradientStop Color="Orange" Offset="1" />

</LinearGradientBrush>

</Border.Background>

</Border>

<ControlTemplate.Triggers>

<Trigger Property="IsMouseOver" Value="True">

<Setter TargetName="ButtonBackground" Property="Background">

<Setter.Value>

<LinearGradientBrush StartPoint="0.5,0" EndPoint="0.5,1" >

<GradientStop Color="#ff232d3e" Offset="0" />

<GradientStop Color="#ff232d3e" Offset="0.75" />

<GradientStop Color="#ff232d3e" Offset="0.75" />

<GradientStop Color="#ff232d3e" Offset="1" />

</LinearGradientBrush>

</Setter.Value>

</Setter>

</Trigger>

</ControlTemplate.Triggers>

</ControlTemplate>

</Setter.Value>

</Setter>

</Style>

</Button. Style>

</Button>

</Grid>

**Разметка cs главного окна**

public partial class MainWindow : Window

{

public MainWindow()

{

InitializeComponent();

}

private void btnClient\_click(object sender, RoutedEventArgs e)

{

CreditWindow credit = new CreditWindow();

credit.Show();

Close();

}

private void btnAdmin\_click(object sender, RoutedEventArgs e)

{

AdminWindow adminWindow = new AdminWindow();

adminWindow.Show();

Close();

}

private void btnClose\_click(object sender, RoutedEventArgs e)

{

Application.Current.Shutdown();

}

}

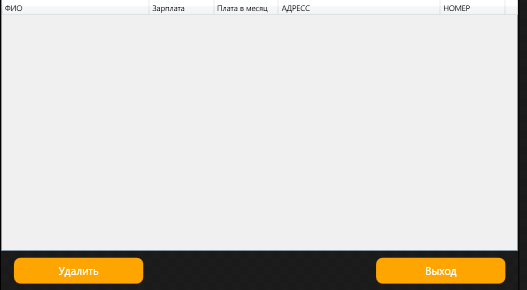


Рис 3.2. Окно администратора

Xaml разметка окна администратора

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="\*"></RowDefinition>

<RowDefinition Height="60"></RowDefinition>

</Grid.RowDefinitions>

<DataGrid Grid.Row="0" x:Name="DGridRecord" AutoGenerateColumns="False" IsReadOnly="True" >

<DataGrid.Columns>

<DataGridTextColumn Header="ФИО" Binding="{Binding FIO}" Width="\*"></DataGridTextColumn>

<DataGridTextColumn Header="Зарплата" Binding="{Binding Wage}" Width="100"></DataGridTextColumn>

<DataGridTextColumn Header="Плата в месяц" Binding="{Binding PayPerMonth}" Width="100"></DataGridTextColumn>

<DataGridTextColumn Header="АДРЕСС" Binding="{Binding Adress}" Width="250"></DataGridTextColumn>

<DataGridTextColumn Header="НОМЕР" Binding="{Binding PhoneNumber}" Width="100"></DataGridTextColumn>

<DataGridTemplateColumn Width="auto">

<DataGridTemplateColumn.CellTemplate>

<DataTemplate>

<Button Content="Редактировать" Margin="5" Width="175" Height="30" VerticalAlignment="Center" Background="Orange" Click="BtnEdit1\_click"></Button>

</DataTemplate>

</DataGridTemplateColumn.CellTemplate>

</DataGridTemplateColumn>

</DataGrid.Columns>

</DataGrid>

<Button x:Name="btndelete"

Width="200"

Height="40"

VerticalAlignment="Center"

HorizontalAlignment="Left"

Background="#ff1d222c"

Foreground="White"

Content="Удалить"

FontSize="17"

Grid.Row="1"

Margin="20,0,0,0"

Click="btnDel\_click">

<Button.Style>

<Style TargetType="Button" >

<Setter Property="Foreground" Value="#a0a0a0"/>

<Setter Property="Template">

<Setter.Value>

<ControlTemplate TargetType="{x:Type Button}">

<Border x:Name="ButtonBackground" BorderBrush="Gray">

<ContentPresenter VerticalAlignment="Center" HorizontalAlignment="Center"/>

<Border.BorderThickness>0</Border.BorderThickness>

<Border.CornerRadius>10</Border.CornerRadius>

<Border.Background>

<LinearGradientBrush EndPoint="0.5,1" StartPoint="0.5,0">

<GradientStop Color="Orange" Offset="0" />

<GradientStop Color="Orange" Offset="1" />

</LinearGradientBrush>

</Border.Background>

</Border>

<ControlTemplate.Triggers>

<Trigger Property="IsMouseOver" Value="True">

<Setter TargetName="ButtonBackground" Property="Background">

<Setter.Value>

<LinearGradientBrush StartPoint="0.5,0" EndPoint="0.5,1" >

<GradientStop Color="#ff232d3e" Offset="0" />

<GradientStop Color="#ff232d3e" Offset="0.75" />

<GradientStop Color="#ff232d3e" Offset="0.75" />

<GradientStop Color="#ff232d3e" Offset="1" />

</LinearGradientBrush>

</Setter.Value>

</Setter>

</Trigger>

</ControlTemplate.Triggers>

</ControlTemplate>

</Setter.Value>

</Setter>

</Style>

</Button.Style>

</Button>

<Button x:Name="btnExit"

Width="200"

Height="40"

VerticalAlignment="Center"

HorizontalAlignment="Right"

Background="#ff1d222c"

Foreground="White"

Content="Выход"

FontSize="17"

Grid.Row="1"

Margin="0,0,20,0"

Click="btnEXit\_click">

<Button.Style>

<Style TargetType="Button" >

<Setter Property="Foreground" Value="#a0a0a0"/>

<Setter Property="Template">

<Setter.Value>

<ControlTemplate TargetType="{x:Type Button}">

<Border x:Name="ButtonBackground" BorderBrush="Gray">

<ContentPresenter VerticalAlignment="Center" HorizontalAlignment="Center"/>

<Border.BorderThickness>0</Border.BorderThickness>

<Border.CornerRadius>10</Border.CornerRadius>

<Border.Background>

<LinearGradientBrush EndPoint="0.5,1" StartPoint="0.5,0">

<GradientStop Color="Orange" Offset="0" />

<GradientStop Color="Orange" Offset="1" />

</LinearGradientBrush>

</Border.Background>

</Border>

<ControlTemplate.Triggers>

<Trigger Property="IsMouseOver" Value="True">

<Setter TargetName="ButtonBackground" Property="Background">

<Setter.Value>

<LinearGradientBrush StartPoint="0.5,0" EndPoint="0.5,1" >

<GradientStop Color="#ff232d3e" Offset="0" />

<GradientStop Color="#ff232d3e" Offset="0.75" />

<GradientStop Color="#ff232d3e" Offset="0.75" />

<GradientStop Color="#ff232d3e" Offset="1" />

</LinearGradientBrush>

</Setter.Value>

</Setter>

</Trigger>

</ControlTemplate.Triggers>

</ControlTemplate>

</Setter.Value>

</Setter>

</Style>

</Button.Style>

</Button>

</Grid>

**Сs разметка окна администратора**

public partial class BasePage : Page

{

AdminWindow admin1;

public BasePage(AdminWindow admin)

{

InitializeComponent();

DGridRecord.ItemsSource = CreditEntities.GetContext().Credits.ToList();

admin1 = admin;

}

private void BtnEdit1\_click(object sender, RoutedEventArgs e)

{

EditWindow pageA = new EditWindow((sender as Button).DataContext as Credit);

pageA.Show();

}

private void btnDel\_click(object sender, RoutedEventArgs e)

{

var rec = DGridRecord.SelectedItems.Cast<Credit>().ToList();

if (MessageBox.Show($"Вы точно хотите удалить следующие элементы {rec.Count()}?", "Alarm", MessageBoxButton.YesNo, MessageBoxImage.Question) == MessageBoxResult.Yes)

{

CreditEntities.GetContext().Credits.RemoveRange(rec);

CreditEntities.GetContext().SaveChanges();

MessageBox.Show("Вы удалили");

DGridRecord.ItemsSource = CreditEntities.GetContext().Credits.ToList();

}

}

private void btnEXit\_click(object sender, RoutedEventArgs e)

{

MainWindow main = new MainWindow();

main.Show();

if (admin1 != null)

{

admin1.Close();

}

}

}

В данной главе был спроектирован код ИС. Кроме этого были разобраны этапы разработки ИС и понятие, что такое разработка.

# ГЛАВА 4. ТЕСТИРОВАНИЕ

Тестирование программного обеспечения — процесс исследования, испытания программного продукта, имеющий своей целью проверку соответствия между реальным поведением программы и её ожидаемым поведением на конечном наборе тестов, выбранных определённым образом. Тестирование является неотъемлемой частью жизненного цикла программного обеспечения. Само по себе тестирование – длительный процесс проверок на соответствие ожидаемого результата. Нельзя выделить какой-то один этап как важный, каждый из них имеет одинаковый вес. При создании продукта тестировщик не просто играет важную роль, а участвует на каждом этапе разработки от концепции до выхода продукта в свет.

Как уже было отмечено, тестирование – неотъемлемая часть жизненного цикла продукта, однако необходимо понимать, с чего стоит начинать тестирование, и когда оно заканчивается. Например, неопытные специалисты ошибочно думают, что после релиза они могут переключаться на другой проект.

Всего принято выделять 7 этапов тестирования:

* Работа с требованиями. Знакомство с требованиями заказчика, что должен из себя представлять итоговый продукт, обсуждение.
* Разработка стратегии тестирования. Оценка сроков тестирования, выявление среды тестирования, объединение всей информации, полученной при работе с требованиями.
* Создание тестовой документации. Написание сценариев, которые позволят проверить функционал.
* Тестирование прототипа. Тестирование основного функционала продукта, корректировка целей, добавление фичей.
* Основное тестирование. Выполнение общей проверки продукта.
* Стабилизация. На данном этапе происходит работа над устранением багов.
* Эксплуатация. Проводится регресс-тестирование, устранение ошибок, которые нашел конечный пользователь.

## 4.1 Тест- план

Тест-план – это большой документ, который чаще всего описывает весь объем работ по тестированию проекта либо части проекта. План тестирования описывает, что будет тестироваться, в какие сроки, какими инструментами, какая команда, обязанности и ответственности каждого члена команды.

Тестирование может быть начато, если выполнены следующие условия:

1. Готова и утверждена необходимая документация;
2. Тестируемый функционал окончен и готов для передачи в тестирование.

Тестирование окончено, если выполнены следующие условия:

1. Все найденные дефекты задокументированы.

Тест план работы следующий:

1. Позитивное тестирование приложения (корректные шаги, корректные данные).
2. Негативное тестирование (подразумевает введение некорректных данных).
3. Функциональное тестирование;
4. Юзабилити тестирование;
5. Тестирование пользовательского интерфейса.

## Написание текст-кейсов

Тест кейс — это проверка работоспособности программы или проекта.  
Написать тест кейс — значит создать текстовое описание процесса тестирования какой-то части или функции проекта.

Тест кейсы нужны, чтобы члены команды могли проверить программу и познакомиться с ней, не читая весь код, а изучив только тест кейс.

У стандартного тест кейса есть 5 частей, то есть 5 атрибутов тест кейса:

1. Порядковый номер тест кейса
2. Название тест кейса. Из него должно быть понятно, в чем суть тест кейса
3. Предусловия тест кейса. Это условия, которые необходимы для проведения тест кейса. Они должны быть выполнены еще до запуска тест кейса.
4. Порядок действий в тест кейсе и описания действий в тест кейсе
5. Ожидаемый результат тест кейса.

Тест-кейс №1 Заполнение анкеты на кредит.

|  |  |  |
| --- | --- | --- |
| **Шаги** | **Ожидаемый результат** | **Результат** |
| Запустить приложение | Приложение должно запуститься | Приложение запустилось |
| Зайти в во вкладку клиента | Должно открыться окно с анкетой | Окно с анкетой открылось |
| Заполнение анкеты | После заполнения должна подсветиться кнопка «далее» | После ввода корректных данных появилась кнопка. |

Тест-кейс №2 Запуск панели администратора

|  |  |  |
| --- | --- | --- |
| **Шаги** | **Ожидаемый результат** | **Результат** |
| Вернуться на главный экран | Должна открыться главная страница | Главная страница открылась |
| Нажимаем на кнопку «Админ» | Открывается окно администратора | Окно администратора открылось |

Тест-кейс №3 Удаление данных при помощи панели администратора

|  |  |  |
| --- | --- | --- |
| **Шаги** | **Ожидаемый результат** | **Результат** |
| Выбираем нужную строку для удаления | Открывается диалоговое окно для удаления | Диалоговое окно открылось |
| Нажимаем кнопку «удалить» | Должно появиться подтверждение удаления строки | Подтверждение появилось , строка успешно удалилась |

## Юнит- тестирование

Модульное тестирование, или юнит-тестирование — процесс в программировании, позволяющий проверить на корректность отдельные модули исходного кода программы.

Юнит-тесты могут служить в качестве документации к коду. Грамотный набор тестов, который покрывает возможные способы использования, ограничения и потенциальные ошибки, ничуть не хуже специально написанных примеров, и, кроме того, его можно скомпилировать и убедиться в корректности реализации.

Чтобы модульное тестирование было максимально эффективным, тесты должны:

* соответствовать конкретному модулю — нельзя применять один и тот же тест для тестирования разных по назначению и реализации программных компонентов;
* быть автоматизированными — тест лучше вписать в сам код, тогда он будет запускаться автоматически и сильно упростит жизнь разработчику;
* быть своевременными — если тест нельзя написать до разработки самого кода, его лучше создавать параллельно, что сэкономит много времени в дальнейшем;
* отвечать основным задачам — при написании теста не нужно стараться учесть все возможные сценарии, лучше сосредоточиться сначала на основных, а остальные дополнять по мере необходимости;
* иметь хорошее название — описывающее, что именно тестируется, в каких условиях и с каким желаемым результатом.

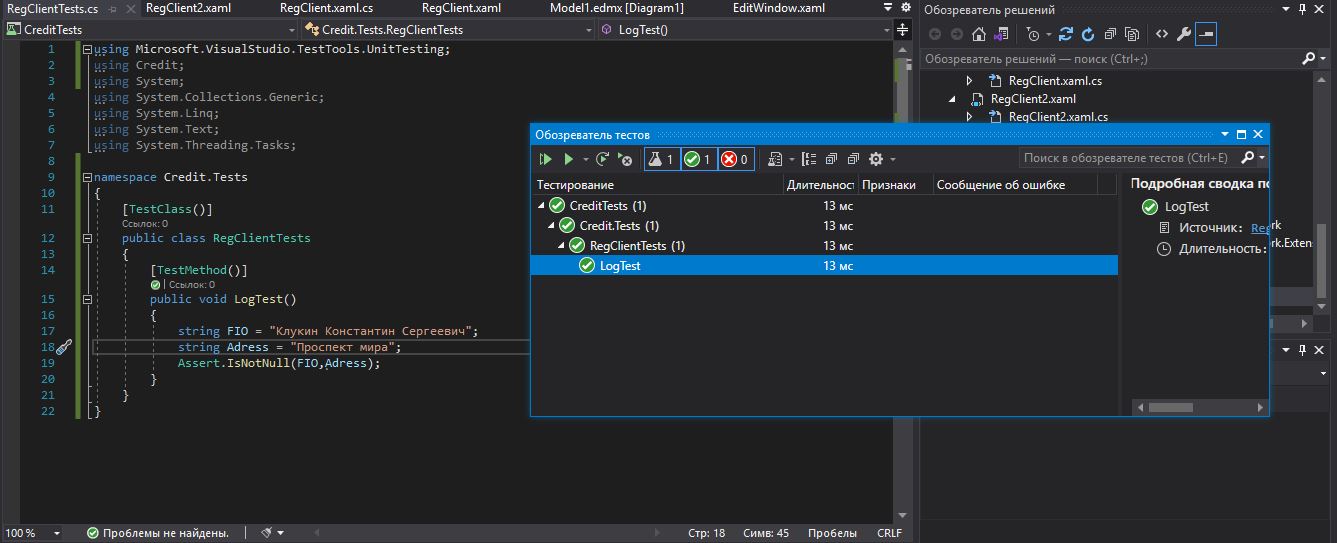


Рис 4 Юнит-тест

На данном скриншоте представлен юнит-тест, который проверяет существование поступленных данных.

В данной главе был разработан те-т план, составлены тест-кейсы и успешно проведено юнит тестирование программы.

# Заключение

Таким образом, в данной работе была рассмотрена определенная предметная область по кредитованию клиентов, со своими бизнес процессами. Кроме этого было сделано настольное приложение, в котором реализованы функции для кредитования. В процессе работы мы закрепили знания по программированию и созданию настольного приложения с графическим интерфейсом, кроме этого подкрепили работу с базами данных.

# Список источников

1. Правильные тест-кейсы. <https://victorz.ru/202001101079>
2. Юнит-тестирование для чайников. <https://habr.com/ru/post/169381/>
3. Модульное тестирование: что это? Типы, инструменты <https://logrocon.ru/news/unit_testing>
4. Разработка ПО. [https://ru.wikipedia.org/wiki/Разработка\_программного\_обеспечения#:~:text=Разрабо́тка](https://ru.wikipedia.org/wiki/%D0%A0%D0%B0%D0%B7%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D0%B0_%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#:~:text=%D0%A0%D0%B0%D0%B7%D1%80%D0%B0%D0%B1%D0%BE%CC%81%D1%82%D0%BA%D0%B0%20%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%CC%81%D0%BC%D0%BC%D0%BD%D0%BE%D0%B3%D0%BE%20%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%CC%81%D1%87%D0%B5%D0%BD%D0%B8%D1%8F%20(%D0%B0%D0%BD%D0%B3%D0)
5. РД 50-34.698-90 Руководство пользователя (пример оформления). <https://www.prj-exp.ru/patterns/pattern_user_guide.php>
6. Основные этапы разработки БД - <https://studfile.net/preview/4601161/page:10/>
7. Коммерческий банк - <https://www.banki.ru/wikibank/kommercheskiy_bank/>
8. Банковское дело - <https://studme.org/1800010219742/bankovskoe_delo/glossariy_banki_i_bankovskoe_delo>
9. Тест-план - <https://software-testing.ru/library/testing/test-analysis/2949-how-to-write-a-software-test-plan>
10. Фундаментальная теория тестирования - <https://habr.com/ru/post/549054/>
11. Тест-кейсы - [https://solvery.io/blog/ru/interesting/chto-takoe-test-case-primer-i-check-list-dlya-nachinayushhih-testirovshhikov](https://solvery.io/blog/ru/interesting/chto-takoe-test-case-primer-i-check-list-dlya-nachinayushhih-testirovshhikov/#%D0%A7%D1%82%D0%BE-%D1%82%D0%B0%D0%BA%D0%BE%D0%B5-%D1%82%D0%B5%D1%81%D1%82-%D0%BA%D0%B5%D0%B9%D1%81)

# Приложение №1

**«Техническое задание»**

**1. Общие сведения о проекте**

1.1. Полное наименование системы и ее условное обозначение

*Информационная система «$outh bank банковская система».*

1.2. Наименование предприятий разработчика и заказчика (пользователя) системы и их реквизиты:

*Заказчик:* ОГБОУ «Костромской энергетический техникум имени Ф.В. Чижова»

*Разработчик: Клукин К.С.*

1.3. Перечень документов, на основании которых создается система, кем и когда утверждены эти документы:

*ISO/IES 12207:1995-08-01 «Информационная технология. Процессы ЖЦ программного обеспечения»*

*ГОСТ 34.601-90 «Стадии создания АС»*

*ГОСТ 34.602-89 «Техническое задание на создание АС»*

*ГОСТ 34.603-92 «Виды испытаний АС»*

*РД 50-34.698-90 «Требование к содержанию документов»*

1.4. Плановые сроки начала и окончания работы по созданию системы:

*В соответствии с календарным графиком на курсовую работу 06.04.2022-14.06.2022.*

1.5. Порядок оформления и предъявления заказчику результатов работ по созданию системы (ее частей), по изготовлению и наладке отдельных средств (технических, программных, информационных) и программно-технических (программно-методических) комплексов системы:

* *при выполнении каждого этапа раз в неделю разработчик предъявляет заказчику результаты своей работы в виде графиков, диаграмм, отчетов, таблиц, рисунков.*
* *после окончания каждого этапа разработчик предоставляет заказчику документацию, соответствующую этому этапу.*

**2. Назначения и цели создания системы**

2.1. Назначение системы

*Информационная система «$outh bank банковская система» для автоматизации кредитования клиентов.*

2.2. Цели создания системы:

* *формирование ведомости по оплате;*
* *регистрация договоров на кредитование;*
* *учет* *клиентов;*
* **3. Характеристики объекта автоматизации**

3.1. Краткие сведения об объекте автоматизации

*Объектом автоматизации является деятельность банковской системы я, которую можно разделить на два бизнес-процесса: учет оплаты, учет клиентов.*

3.2. Сведения об условиях эксплуатации объекта автоматизации и характеристиках окружающей среды.

*При проектировании ИС банка должны быть автоматизировано рабочее место оператора.*

**4. Требования к системе**

4.1 Требования к структуре и функционированию системы:

*ИС представляет собой единую систему, разграничение прав доступа осуществляется на уровне операционной системы.*

*Пользователь «Администратор» имеет право доступа ко всей информационной системе.*

*Пользователь «Клиент» имеет право доступа к операциям, связанные с кредитованием.*

4.1.2. Требования к численности, квалификации персонала

*Данная область предполагает работу за компьютером оператора. Поэтому необходимо для него провести обучение для пользования ИС и основам работы на компьютере.*

4.1.3. Требования к надежности:

*Должна быть обеспечена надежность хранения ИС, которая заключается в возможности восстановления ИС после любого рода аппаратных и программных сбоев. Для этого необходимо обеспечить его устойчивую и бесперебойную работу, что достигается наличием источника бесперебойного питания. Периодически надо производить резервное копирование ИС.*

*ИС должна быть защищена от потери данных при неправильных действиях пользователя. Также должна иметься защита от несанкционированного доступа, что достигается путем идентификации пользователей (ведение системы паролей), то есть пользователь должен получить доступ только к тем данным и программам, которые требуются ему для его работы.*

4.1.4. Требования по эргономике и технической эстетике

*Пользовательские интерфейсы должны быть просты и понятны для неподготовленных пользователей и выполнены в спокойных тонах*

*Комфортность условий работы персонала должна обеспечиваться в соответствии со стандартами:*

*ГОСТ 12.2.032-78 «Рабочее место при выполнении работ сидя. Общие эргономические требования»;*

*ГОСТ Р 50923-96 «Дисплеи. Рабочее место оператора. Общие эргономические требования к производственной среде. Методы измерения».*

4.1.5. Требования по стандартизации и унификации:

*Необходимо применять стандартное общее и базовое программное обеспечение, включающее лицензионные ОС, СУБД, использовать утвержденные формы входных и выходных документов, применять унифицированные интерфейсы.*

4.2 Требования к функциям (задачам), выполняемым системой (функциональные блоки нижнего уровня иерархии дерева узлов):

* *анализ оплаты кредита;*
* *формирование списка клиентов;*
* *формирование договоров на кредитование;*
* *составление ведомости оплаты;*
* *составление отчета о оплатах;*

4.3 Требования к видам обеспечения

4.3.1. Требования к информационному и программному обеспечению

*Данные в системе должны быть реализованы в виде реляционной БД на основе MySQLi. БД располагается на локальном компьютере под управлением операционной системы Windows 8/8.1/10.*

*Требования к техническому обеспечению*

*Локальный компьютер должен быть оснащен процессором не ниже AMD Phenom(tm) II X6 1035T Processor 2.60 GHz, оперативной памятью от 1024MB. Клиентские места должны быть оснащены процессором AMD Phenom(tm) II X6 1035T Processor 2.60 GHz. Для распечатки документов и отчетов необходимо установить принтеры.*

**5. Состав и содержание работ по созданию (развитию) системы**

Работы по созданию системы выполняются последовательно и включают следующие этапы:

5.1. Формирование требований к ИС

*Характеристика объекта и результатов его функционирования:*

*Разработчик – изучает работу оператора банка, анализирует полученную информацию, делает вывод о необходимости ИС, выявляет лица, для которых будет предназначена данная система, выделяет проблемы, которые можно решить с помощью автоматизации.*

*Заказчик – предоставляет разработчику необходимую информацию в виде документов, отчетов, выписок, журналов и прочее. Результатом этапа является описание предметной области.*

5.2. Разработка концепций ИС

*Изучение объекта:*

*Разработчик – проводит детальное изучение работы отдела кредитования. Результатом на данном этапе является функциональной модели IDIF0. Заказчик – предоставляет разработчику все необходимую для этого информацию.*

5.4. Технический проект

*Разработка проектных решений по системе: разработчик - создает общие решения по системе и ее частям, по функционально-алгоритмической структуре системы, по функциям персонала и организационной структуре. Определяет структуру технических средств, алгоритмы решения и применяемые языки. Разрабатывает решения по организации и ведению информационной базы, по системе классификации и кодированию информации, по программному обеспечению.*

5.5. Рабочая документация

*Разработка рабочей документации на ИС: разработчик – создает руководство пользователя на основании стандартов РД 50-34.698-90 и ЕСПД, которое должно быть представлено в пояснительной записке.*

5.6. Ввод в действие

*Организационная подготовка объекта информатизации к вводу ИС: разработчик и заказчик - проводят организационную подготовку объекта к вводу ИС.*

*Подготовка персонала: разработчик - обучает персонал, проверяет их способность обеспечить функционирование ИС.*

*Строительно-монтажные работы: производится установка и наладка технических средств.*

*Испытания системы: проводятся предварительные испытания, опытная эксплуатация и приемочные испытания. Более подробно испытания системы представлены в пункте "Порядок контроля и приемки системы" настоящего документа.*

5.7. Сопровождение ИС

*Разработчик - анализирует функционирование системы, выявляет отклонения эксплуатационных характеристик от проектных значений, устраняет причины этих отклонений.*

**6. Порядок контроля и приемки системы**

*При проведении испытаний выявляются ошибки, ИС проверяется на соответствие функциональным и эксплуатационным требованиям, требованиям нормативных документов и ТЗ, проверяется надежность системы. При необходимости система дорабатывается и готовится к следующему этапу испытаний.*

**7. Требования к составу и содержанию работ по подготовке объекта автоматизации к вводу в действие**

7.1. Преобразование входной информации виду, пригодному для обработки с помощью ПК.

*Все виды документации приводятся к единой стандартной форме.*

7.2. Создание условий функционирования объекта автоматизации, при которых гарантируется соответствие создаваемой системы требованиям, содержащимся в ТЗ

*Заказчик обязуется к моменту запуска ИС подготовить необходимые помещения для серверной, оплатить разработку проекта локальной сети и ее реализацию.*

7.4. Сроки и порядок комплектования штатов и обучения персонала

*Перед началом работы с ИС работники отдела кредитования должны пройти начальный курс работы с ПК и курс обучения работы с данной ИС.*