|  |  |
| --- | --- |
| **Gerb-BMSTU_01** | **Министерство науки и высшего образования Российской Федерации**  Калужский филиал  федерального государственного бюджетного  образовательного учреждения высшего образования  ***«Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)»***  ***(КФ МГТУ им. Н.Э. Баумана)*** |

|  |  |
| --- | --- |
| **ФАКУЛЬТЕТ** | ***ИУК «Информатика и управление»*** |
| **КАФЕДРА** | ***ИУК4 «Программное обеспечение ЭВМ,*** |
|  | ***информационные технологии»*** |

**РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**к курсовой работе на тему:**

|  |
| --- |
| ***Разработка банковского программного обеспечения*** |
|  |

|  |  |
| --- | --- |
| по дисциплине | *Базы данных* |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Студент гр. | *ИУК4-62Б* | |  |  | ( | *Калашников А.С.* | ) |
|  | |  |  | (подпись) |  | (Ф.И.О.) |  |
| Руководитель | |  |  |  | ( | *Глебов С.А.* | ) |
|  | |  |  | (подпись) |  | (Ф.И.О.) |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Оценка руководителя |  | баллов |  |  |
|  | 30-50 |  | (дата) |  |
| Оценка защиты |  | баллов |  |  |
|  | 30-50 |  | (дата) |  |
| Оценка проекта |  | баллов |  | |
|  |  |  | (оценка по пятибалльной шкале) | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Комиссия: |  |  | ( | *Глебов С.А.* | ) |
|  |  | (подпись) |  | (Ф.И.О.) |  |
|  |  |  | ( | *Гришунов С.С.* | ) |
|  |  | (подпись) |  | (Ф.И.О.) |  |
|  |  |  | ( | *Гагарин Ю.Е.* | ) |
|  |  | (подпись) |  | (Ф.И.О.) |  |

Калуга, 2023

Калужский филиал   
федерального государственного бюджетного образовательного учреждения высшего образования

***«Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)»   
(КФ МГТУ им. Н.Э. Баумана)***

УТВЕРЖДАЮ

Заведующий кафедрой *ИУК4*

( *Гагарин Ю.Е.)*

« *09* » *февраля* 20*23* г.

**ЗАДАНИЕ**

**на выполнение курсовой работе**

|  |  |
| --- | --- |
| по дисциплине | ***Базы данных*** |

|  |  |  |
| --- | --- | --- |
| Студент | *Калашников А.С. ИУК4-62Б* | |
|  | | (фамилия, инициалы, индекс группы) |
| Руководитель | | *Глебов С.А.* |
|  | | (фамилия, инициалы) |

График выполнения работы: 25% к *4* нед., 50% к *7* нед., 75% к *10* нед., 100% к *14*  нед.

|  |
| --- |
| ***1. Тема курсовой работы*** |
| Разработка банковского программного обеспечения |
|  |
|  |
| ***2. Техническое задание*** |
| Разработать набор приложений обеспечивающий комфортное взаимодействие |
| пользователей и сотрудников банка |
|  |

|  |
| --- |
| ***3. Оформление курсовой работы*** |
| 3.1. Расчетно-пояснительная записка на  *48*  листах формата А4. |
| 3.2. Перечень графического материала КР (плакаты, схемы, чертежи и т.п.) |
| *1. Структура БД* |
| *2. Демонстрационный чертеж* |
|  |
|  |

Дата выдачи задания « *09* » *февраля* *2023* г.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Руководитель курсовой работы | | |  | | / | *Глебов С.А.* |
|  | | (подпись) | | |  | (Ф.И.О.) |
| Задание получил |  | | / | *Калашников А.С.* | / | « *09* » *февраля* 20*23* г. |
|  | (подпись) | |  | (Ф.И.О.) |  |  |

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 4](#_Toc134722967)

[1. МЕТОДЫ И ИНСТРУМЕНТЫ ПРОГРАММНОЙ ИНЖЕНЕРИИ 6](#_Toc134722968)

[1.1. Техническое задание 6](#_Toc134722969)

[1.2. Постановка задачи 9](#_Toc134722970)

[1.3. Анализ существующих аналогов 9](#_Toc134722971)

[1.4. Перечень задач, подлежащих решению в процессе разработки. 9](#_Toc134722972)

[1.5. Обоснование выбора СУБД 10](#_Toc134722973)

[1.6. Обоснование выбора языков программирования 17](#_Toc134722974)

[1.7. Обоснование выбора сред разработки графического интерфейса 23](#_Toc134722975)

[1.8. Обоснование выбора сред разработки 25](#_Toc134722976)

[1.9. Выбор средства для взаимодействия с базой данных 28](#_Toc134722977)

[1.10 Выводы 29](#_Toc134722978)

[2. ПРОЕКТИРОВАНИЕ КОМПОНЕНТОВ ПРОГРАММНОГО ПРОДУКТА 31](#_Toc134722979)

[2.1. Общие сведения 31](#_Toc134722980)

[2.2. Разработка структуры системы 31](#_Toc134722981)

[2.3. Разработка базы данных 31](#_Toc134722982)

[2.4. Расчет денежных средств на счете 37](#_Toc134722983)

[2.4.1 Расчет денежных средств при переводе 37](#_Toc134722984)

[2.4.2 Расчет денежных средств на счетах с пополнением 37](#_Toc134722985)

[2.4.3 Расчет денежных средств на счетах с снятием 37](#_Toc134722986)

[2.4.4 Расчет денежных средств на счету с «Оставить средства на счете» 37](#_Toc134722987)

[3. ТЕСТИРОВАНИЕ И ИНТЕГРАЦИЯ КОМПОНЕНТОВ ПРОГРАММНОГО ПРОДУКТА 38](#_Toc134722988)

[3.1. Назначение системы 38](#_Toc134722989)

[3.2. Требования к аппаратной платформе 38](#_Toc134722990)

[3.3. Руководство пользователя 39](#_Toc134722991)

[3.4. Руководство администратора 45](#_Toc134722992)

[ЗАКЛЮЧЕНИЕ 46](#_Toc134722993)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 47](#_Toc134722994)

ВВЕДЕНИЕ

Курсовой работы по созданию банковского программного обеспечения является актуальной темой в современном мире, так как банки активно развиваются и используют новые технологии для улучшения своей деятельности. Кроме того, традиционные банковские услуги становятся все более цифровыми и современными, поэтому банковское ПО становится необходимым инструмент.

Объектом курсовой работы по созданию банковского программного обеспечения является система, состоящая из отдельных программ, которые обеспечивают эффективную работу банка в целом.

Предметом исследования курсовой работы по созданию банковского программного обеспечения являются различные аспекты разработки ПО для банковской сферы, включая определение требований к создаваемому ПО, проектирование архитектуры системы, выбор используемых технологий и основных модулей, создание базы данных, написание программного кода, тестирование и внедрение программного продукта в банковскую инфраструктуру.

Целью курсовой работы по созданию банковского программного обеспечения является разработка функционально полного программного продукта, который будет обеспечивать эффективное управление финансовыми процессами в банке и удобство использования услуг банка для клиентов.

Для достижения поставленной цели решаются следующие задачи:

1. Выполнить анализ предметной области
2. Провести сравнительный анализ существующих аналогов
3. Определить оптимальную структуру системы
4. Осуществить выбор средств реализации программного продукта, соответствующего выбранной структуре.
5. Реализовать схему базы данных
6. Реализовать базу данных и программные компоненты системы
7. Осуществить тестирование компонентов
8. Разработать сопроводительную документацию

1. МЕТОДЫ И ИНСТРУМЕНТЫ ПРОГРАММНОЙ ИНЖЕНЕРИИ

1.1. Техническое задание

Наименование системы

Настоящее Техническое задание определяет требования и порядок создания банковского программного обеспечения «Просто банк».

Основания для разработки

Основанием для разработки банковского программного обеспечения является необходимость обновления и улучшения банковских услуг, а также растущие потребности клиентов в удобном и быстром взаимодействии с банком. Современная жизнь требует быстрого и качественного обслуживания, а развитие технологий позволяет создать продукты и сервисы, которые максимально соответствуют потребностям клиентов.

Кроме того, банки действуют в условиях жесткой конкуренции, и чтобы сохранять свои позиции и привлекать новых клиентов, им необходимо обеспечить удобные и современные сервисы, использующие все новые технологии.

Разработка банковского программного обеспечения позволяет автоматизировать процессы, связанные с обслуживанием клиентов, услугами платежей, кредитования и другими банковскими операциями, а также обеспечить более точный и удобный контроль за всеми финансовыми потоками. Все это помогает банкам повысить свою эффективность и конкурентоспособность на рынке.

Исполнитель

Исполнителем проекта является студент Калужского филиала МГТУ им. Н. Э. Баумана, факультета ИУК, группы ИУК4-62Б, Калашников Артем Сергеевич.

Краткая характеристика области применения

Разрабатываемая система предназначена для автоматизации процессов управления регистрационными записями клиентов, обработки платежей, управления оборотным капиталом и денежными потоками, а также для контроля за финансовыми операциями и проведения аналитики деятельности банка.

Целевая аудитория

Инженерно-технический персонал и клиенты.

Назначение приложений

Основное назначение банковской системы заключается в обеспечении как следующих основных задач:

1. предоставление услуг клиентам: банковская система позволяет клиентам получать доступ к банковским услугам, таким как открытие счетов, выдача кредитов, проведение платежных операций и т.д.;
2. управление финансовыми ресурсами банка: банковская система позволяет банку эффективно управлять своими ресурсами, включая мониторинг и управление денежными потоками, управление кредитным портфелем, управление рисками и т.д.;
3. обработка финансовых операций: банковская система обеспечивает обработку и учет финансов

Цели разработки

Целью создания банковской системы являются оптимизация работы банка и улучшение качества обслуживания клиентов.

Плановые сроки начала и окончания работы по созданию системы

Планируемые сроки начала и окончания работы над проектом: 09.02.2023 – 28.04.2023.

Требования к приложениям

Разрабатываемая система должна являться набором приложений для взаимодействия клиента и банка и должна иметь следующий функционал:

1. Регистрация и авторизация пользователя;
2. Автоматическое добавление каждому клиенту базового счета
3. Просмотр информации о пользователе;
4. Добавление нового счета;
5. Перевод денежных средств
6. Просмотр недавних взаимодействий по счету
7. История по счету
8. Вкладывания денежных средств по определенным условиям
9. Добавление администратором новые вклады с различными условиями

Требования к временным характеристикам

Время восстановления после отказа, вызванного сбоем электропитания технических средств (иными внешними факторами), не фатальным сбоем (не крахом) операционной системы, не должно превышать времени восстановления операционной системы и восстановления работы сети.

Время восстановления после отказа, вызванного неисправностью технических средств, фатальным сбоем (крахом) операционной системы, не должно превышать времени, требуемого на устранение неисправностей технических средств и переустановки программных средств.

Этапы разработки

На стадии разработки технического задания должен быть выполнен этап разработки, согласования и утверждения настоящего технического задания.

На стадии проектирования программы должен быть выполнен этап выборки программного обеспечения, библиотек для создания, этап проектирования системы в целом, разработка технической документации.

На стадии реализации производится разработка и тестирование спроектированного приложения.

1.2. Постановка задачи

Задачей данной курсовой работы является разработка банковской системы, которая обеспечивает эффективное управление финансовыми процессами в банке и удобство использования услуг банка для клиентов.

1.3. Анализ существующих аналогов

В настоящее время существуют у каждого банка существует свой набор банковских приложений для взаимодействия клиентов и сотрудников. Примерами таких систем могут послужить банки «Сбербанк», «ВТБ» и «Альфа банк» и т.д. Функционал, который реализуют данные банковские системы, может быть различен в зависимости от специфики работы каждой организации, но всё же в основе всегда лежат базовые пользовательские функции, такие как: перевод денежных счетов, открытие и закрытие счетов, создание новых банковских вкладов с различными условиями и т.д.

1.4. Перечень задач, подлежащих решению в процессе разработки.

Для выполнения поставленной задачи необходимо определить оптимальную архитектуру для построения системы и выбрать подходящие средства разработки для данной информационной системы.

В приложениях должны быть реализованы функции, с помощью которых пользователь сможет взаимодействовать с системой, а именно: регистрацию и авторизацию пользователей, выполнять переводы с одного счета на другой, возможность просматривать информацию о данном счете, создавать или открывать новые счета. Также пользователь должен иметь возможность просматривать историю по всем счетам или интересующему счету.

Также должна быть корректно разработана схема базы данных, которая будет отвечать за хранение и систематизацию данных приложения.

Средства реализации включают в себя: язык программирования для написания приложения, среду разработки, СУБД, графический интерфейс.

1.5. Обоснование выбора СУБД

Общие сведения

В разрабатываемой системе данные играют важную роль, поэтому для их хранения и обработки используется база данных. Системы управления базами данных (СУБД) используются для управления базами данных и обеспечения доступа к различным типам данных. Это позволяет эффективно работать с информацией в системе и добиться более высокой производительности и надежности в работе с данными.

Хотя все системы управления базами данных выполняют общую функцию - позволяют пользователям создавать, редактировать и получать доступ к информации, хранящейся в базах данных, процессы выполнения этой функции имеют различия в широком диапазоне. Кроме того, функциональные возможности каждой СУБД могут со значительными различиями.

Существует шесть основных типов баз данных: иерархическая, сетевая, реляционная, объектно-ориентированная, объектно-реляционная, функциональная.

При разработке приложений, которые используют иерархическую или сетевую модель СУБД, структура базы данных «вшита» в само приложение. Это означает, что приложение зависит от конкретной реализации базы данных, что делает его неспособным к работе с другими СУБД. Если мы добавляем новый атрибут в базу данных, в которой уже имеются данные, то приложение может потребовать значительных изменений в своей структуре, что может остановить работу приложения.

Функциональные базы данных используются для решения аналитических задач, таких как финансовое моделирование и управление производительностью. Включает в себя многомерное иерархическое объединение.

Реляционная модель данных позволяет представлять информацию о предметной области с помощью взаимосвязанных таблиц. В реляционных базах данных вся информация сведена в таблицы, строки и столбцы, которые называются записями и полями соответственно. Записи в таблицах не повторяются, их уникальность обеспечивается первичным ключом, содержащим набор полей, однозначно определяющих запись.

Объектно-реляционные базы становятся популярны и являются реляционными базами данных с объектно-ориентированными подходами, что является удобным инструментом. Но такие базы переносят часть логики с сервера в базу данных, что противоречит принципу независимости данных от приложения, что не даёт разработчикам отказаться от реляционных моделей.

Для системы было решено использовать либо реляционную, либо объектно-реляционную базу данных. Это обосновано рядом преимуществ, среди которых: простота и наглядность по сравнению с другими моделями, удобство физической реализации, применение строгих правил при проектировании баз данных, независимость данных, их логическая связь, широкая техническая поддержка, наличие стандартов и документации.

Для разработки базы данных было необходимо выбрать наиболее подходящую СУБД для поставленной задачи. В силу распространенности, для рассмотрения были выбраны: PostgreSQL, SQLite, Firebird, DB2, MySQL и Oracle.

PostgreSQL

PostgreSQL – cвободно распространяемая СУБД, относится к объектно-реляционному типу. Существует в реализациях для множества UNIX-подобных платформ, включая AIX, различные BSD-системы, HP-UX, IRIX, Linux, macOS, Solaris/OpenSolaris, Tru64, QNX, а также для Microsoft Windows.

Postgresql, можно смело говорить, что СУБД отличается стабильными свойствами, в ней сложно разрушить таблицы, о чем упоминалось выше при рассмотрении MySQL. Для многих это свойство при выборе системы для хранения данных может быть решающим.

Отдают предпочтение PostgreSQL те, кто нуждаются в отлично структурированном хранилище, но с гибкими возможностями схемы JSON/BJSON. Расширять кластеры в приложении можно, положившись на помощь сторонних библиотек. Особых сложностей возникнуть не должно, все достаточно грамотно продумано создателями PostgreSQL. Также можно осуществлять шардинг табличной структуры.

Отмечу, что для работы с Postgresql нужно иметь подобный опыт, иначе могут быть непонятные моменты в ее настройке. Для новичков она вряд ли подойдет, либо нужно иметь поддержку опытного учителя.

Система авторизации, установленная по умолчанию в PostgreSQL, на практике может вызвать непонимание. Даже некоторые разработчики с опытом работы с данной системой не понимают, как на самом деле она функционирует. Это же можно отнести к настройкам авторизации.

Преимущества:

1. ПО открытого типа, которое в полной мере отвечает действующему стандарту SQL;
2. Бесплатное приложение;
3. Высокая мощность;
4. Открытый исходный код;
5. Поддержка сообщества пользователей, всегда можно найти ответы на имеющиеся вопросы;
6. Ряд дополнений, которые регулярно пополняются;
7. Многофункциональность;
8. Можно расширить опции за счет сохранения произведенных процедур;
9. Объектно-ориентированная СУБД, а не только реляционная;
10. Есть поддержка наследования.

Недостатки:

1. Может замедлить сервер;
2. Слабая производительность даже при выполнении простых операций чтения данных;
3. Медленнее MySQL;
4. Невысокая популярность, несмотря на большое сообщество пользователей в Сети;
5. Сложно найти хостинг, где есть поддержка СУБД.

SQLite

SQLite - это встроенная файловая СУБД, которая не требует установки или настройки. В свою очередь это означает, что приложение не запускается под отдельным серверным процессом, который необходимо запустить, остановить или настроить. Эта безсерверная архитектура обеспечивает совместимость базы данных на разных платформах.

Полная база данных SQL содержится в одном файле на диске, и все операции чтения и записи выполняются непосредственно в этом файле на диске. Поскольку данные записываются непосредственно обратно в файл диска, SQLite придерживается атрибута ACID для защиты транзакций от сбоев выделения памяти и ошибок ввода-вывода диска, вызванных непредвиденными сбоями системы или сбоями питания.

Преимущества:

1. Полностью свободная лицензия;
2. Файловая структура. Вся база данных состоит из одного файла, поэтому её очень легко переносить на разные машины;
3. Кроссплатформенность;
4. Безопасность. БД хранится в одном файле, права доступа к которому можно контролировать стандартными средствами ОС.

Недостатки:

1. Встраиваемость. Подходит только для создания локальной базы данных.
2. Отсутствие системы пользователей. Более крупные СУБД включают в свой состав системы управления правами доступа пользователей. Обычно применения этой функции не так критично, так как эта СУБД используется в небольших приложениях.
3. Отсутствие возможности увеличения производительности. Исходя из проектирования, довольно сложно выжать что-то более производительное из этой СУБД.

**FireBird**

FireBird – свободно распространяемая СУБД. Поддерживает стандарты ANSI в синтаксисе языка SQL и позволяет работать на многих операционных системах. Firebird используется в различных промышленных системах (складские и хозяйственные, финансовый и государственный сектора).

Преимущества:

1. Многоверсионная архитектура (параллельная обработка оперативных и аналитических запросов: читающие пользователи не блокируют пишущих);
2. Соответствие требованиям ACID.

Недостатки:

1. Отсутствие кэша итогов запросов, полнотекстовых индексов;
2. Значительное падение производительности при росте внутренней фрагментации базы.

MySQL

MySQL – одна из самых популярных баз данных для веб-приложений. Она включает в себя бесплатный пакет программ, однако новые версии выходят постоянно, расширяя функционал и улучшая безопасность. Существуют специальные платные версии, предназначенные для коммерческого использования. В бесплатной версии наибольший упор делается на скорость и надежность, а не на полноту функционала, который может стать и достоинством, и недостатком - в зависимости от области внедрения

В силу того, что MySQL является серверной СУБД, приложения для доступа к данным, в отличии от SQLite работают со службами MySQL. Обычно MySQL используется в качестве сервера, к которому обращаются локальные или удалённые клиенты, однако в дистрибутив входит библиотека внутреннего сервера, позволяющая включать MySQL в автономные программы.

Преимущества:

1. Простота. MySQL легко устанавливается и проста в настройке и использовании. Существует много сторонних инструментов, облегчающих работу с базами данных;
2. Безопасность. Имеет много функций, обеспечивающих безопасность и поддерживаемых по умолчанию;
3. Масштабируемость. MySQL легко работает с большими объемами данных и просто масштабируется;
4. Скорость. Упрощение некоторых стандартов позволяет MySQL значительно увеличить производительность;

Недостатки:

1. Ограничения. По задумке в MySQL заложены некоторые ограничения функционала, которые иногда необходимы в особо требовательных приложениях;
2. Надёжность. Из-за некоторых способов обработки данных MySQL (связи, транзакции, аудиты) иногда уступает другим СУБД по надежности.

**Oracle**

Oracle Database или Oracle RDBMS – объектно-реляционная система управления базами данных компании Oracle. Современная СУБД Oracle – это мощный программный комплекс, позволяющий создавать приложения любой степени сложности. Ядром этого комплекса является база данных, хранящая информацию, количество которой за счет предоставляемых средств масштабирования практически безгранично. C высокой эффективностью работать с этой информацией одновременно может практически любое количество пользователей (при наличии достаточных аппаратных ресурсов), не проявляя тенденции к снижению производительности системы при резком увеличении их числа.

Преимущества:

1. Не требует больших объемов памяти для кеша. При этом память, не задействованная для кеширования файловой системы, может быть сконфигурирована для Oracle memory.;
2. Масштабируемость;
3. Высокая скорость выполнения запросов.

Недостатки:

1. Коммерческая направленность продукта, высокая стоимость;
2. Необходимость дорогостоящего аппаратного обеспечения;
3. Миграция из устаревших файловых систем в ASM может быть проблемой и часто требует отключения системы;

**DB2**

DB2 – это продукт базы данных от IBM. Это система управления реляционными базами данных (RDBMS). DB2 предназначена для эффективного хранения, анализа и извлечения данных.

Преимущества:

1. Бесплатная версия;
2. Нет ограничений на размер базы данных;
3. Ограничение на 256 таблиц нет;
4. Хорошая производительность;
5. Нет проблем с неуникальными индексами.

Недостатки:

1. Требуется тщательная настройка параметров субд;
2. Большой размер базы данных, больше чем в MS SQL Server, PostgreSQL.

1.6. Обоснование выбора языков программирования

**C++**

С++ — это язык программирования общего назначения. В настоящее время существует огромное множество реализаций этого языка, причем часть из них бесплатная, а за некоторые придется платить.

Благодаря тому, что С++ создавался на базе языка программирования С, он сумел перенять всю скорострельность своего родителя. При этом, он сегодня используется во всех сферах программирования.

В отличие от других языков, на C++ можно программировать всё что угодно. Это язык низкого уровня, то есть на нём можно сделать то, что требует каких-то специфических инструкций, например векторных, которые позволяют выполнять сразу несколько операций на одном ядре процессора.

C++ спроектирован так, чтобы код мог компилироваться для разных платформ. Например, браузер Chrome, написанный преимущественно на языке C++, работает на Windows, Linux и macOS. Для языков, поддерживающих низкий уровень, это редкость.

Другого языка с такими же функциями нет. Некоторые пытаются к этому стремиться, например язык программирования Rust. А многие даже и не пытаются — заведомо не получится.

Преимущеста:

1. С++ поддерживает разные технологии и стили программирования, включая объектно-ориентированное, обобщенное и метапрограммирование;
2. Более надежное освобождение ресурсов путем автоматического вызова деструкторов при уничтожении объектов;
3. Возможность создавать пользовательские функции-операторы, позволяющие записывать выражения над пользовательскими типами в алгебраической форме;
4. Важным достоинством является предсказуемое выполнение программ, что является важным для систем реального времени;
5. При помощи шаблонов можно создавать обобщенные контейнеры и алгоритмы для разных типов данных;
6. При помощи шаблонов и множественного наследия можно имитировать классы-примеси и комбинаторную параметризацию библиотек;
7. Поддержка понятия физической и логической константности, что делает программу надежнее, так как позволяет диагностировать ошибочные попытки изменения переменных;
8. C++ - кроссплатформенный язык, его стандарт накладывает минимальные требования на компьютер для запуска скомпилированной программы. На С++ разрабатывают программы для различных платформ и систем с помощью компиляторов для большого количества платформ;
9. В С++ существует возможность низкоуровневой работы с адресами памяти, однако, при неосторожном использовании это легко может превратиться в недостаток;
10. Язык С++ спроектирован так, чтобы у программиста был максимальный контроль над всеми аспектами структуры и порядка выполнения программы;
11. Существующий код на C может с минимальными изменениями использоваться в С++.

Недостатки:

1. Синтаксис, вызывающий ошибки - операцию сравнения и присваивания легко перепутать, тем более присваивание возвращает значение, поэтому конструкция с присваиванием вместо выражения не будет восприниматься компилятором как ошибка, в циклах и ветвлениях так же существует такая ошибка;
2. Макросы являются потенциально опасным средством, несмотря на то что их необходимость не велика, ввиду наличия шаблонов и встроенных функций;
3. Не интуитивные преобразования некоторых типов;
4. Примитивный, унаследованный от С препроцессор;
5. Плохая поддержка модульности. Подключение внешнего модуля через заголовочную вставку серьезно замедляет компиляцию при большом количестве модулей, что решено во многих компиляторах прекомпиляцией заголовочных файлов.
6. В С++ существует много возможностей, нарушающих принципы тип безопасности, что приводит к труднонаходимым ошибкам. Частой проблемой в С++ являются проблемы "переполнения буфера";
7. C++ является слишком сложным и большим языком, из за чего есть трудности с его изучением;
8. Сложность и ограниченность метапрограммирования на основе шаблонов С++;
9. При заявлении, что С++ мультипарадигменный язык, в нем отсутствует явная поддержка функционального программирования. Отчасти это решено различными библиотеками, использующими средства метапрограммирования для расширения языка;
10. Некоторые считают отсутствие встроенной системы сборки мусора недостатком языка.

**Java**

Java - строго типизированный объектно-ориентированный язык программирования общего назначения, разработанный компанией Sun Microsystems (в последующем приобретённой компанией Oracle). Разработка ведётся сообществом, организованным через Java Community Process; язык и основные реализующие его технологии распространяются по лицензии GPL. Права на торговую марку принадлежат корпорации Oracle.

Приложения Java обычно транслируются в специальный байт-код, поэтому они могут работать на любой компьютерной архитектуре, для которой существует реализация виртуальной Java-машины.

Скорость выполнения приложений на Java в несколько раз быстрее чем на Python. Так например, простое двоичное дерево выполняется в Java в 10 раз быстрее.

**C#**

С# («Си Шарп») – один из наиболее быстро растущих, востребованных и при этом «удобных»  языков программирования. C# является объектно-ориентированным и в этом плане много перенял у Java и С++. Это модификация фундаментального языка С от компании Microsoft, призванная создать наиболее универсальное средство для разработки программного обеспечения для большого количества устройств и операционных систем.

Язык C# практически универсален. Можно использовать его для создания любого ПО: продвинутых бизнес-приложений, видеоигр, функциональных веб-приложений, приложений для Windows, macOS, мобильных программ для iOS и Android.

Практически вся операционная система Microsoft существует благодаря С#. Привычные вам утилиты и приложения созданы с использованием этого языка и фреймворков, разработанных для него.

В эту категорию попадает мессенджер Skype, браузер Internet Explorer, среда для разработки Visual Studio 2012, Microsoft Office (все его составляющие, включая Word, PowerPoint, Excel, Outlook и так далее).

Сюда же можно отнести продукты компании Adobe (Photoshop, Lightroom), браузер Mozilla Firefox и Winamp.

Преимущества:

1. C# - простой, надежный и масштабируемый язык программирования
2. Динамически типизированный характер C# облегчает разработчикам поиск ошибок в коде. C# устраняет проблему утечки памяти
3. Он имеет знакомый синтаксис, идентичный языкам C и C ++

Недостатки:

1. Отсутствие классов и наследования, что затрудняет написание кода в объектно ориентированной парадигме
2. Очень строгий компилятор кода, иногда чересчур сильно контролирующий обращения к памяти

**Python**

Python — это высокоуровневый язык программирования, отличающийся эффективностью, простотой и универсальностью использования. Он широко применяется в разработке веб-приложений и прикладного программного обеспечения, а также в машинном обучении и обработке больших данных. За счет простого и интуитивно понятного синтаксиса является одним из распространенных языков для обучения программированию.

В Python операторы кода исполняются последовательно с помощью программы-интерпретатора. Если по ходу исполнения программы встречается ошибка, оно сразу же прекращается. Это позволяет Python-разработчику быстро обнаружить и устранить недочеты, но в то же время снижает производительность.

Язык является полностью объектно-ориентированным в том плане, что всё является объектами. Необычной особенностью языка является выделение блоков кода пробельными отступами. Синтаксис ядра языка минималистичен, за счёт чего на практике редко возникает необходимость обращаться к документации. Сам же язык известен как интерпретируемый и используется в том числе для написания скриптов.

Преимущества:

1. При разработке на Питоне в большинстве случаев требуется писать куда меньше кода, чем при работе с его конкурентами. Сам синтаксис визуально чище и предельно логичен.
2. Язык работает на всех платформах, поэтому с помощью него можно разрабатывать программы на чем угодно и под что угодно.
3. У языка колоссальная поддержка и действительно внушительная коллекция дополнений от сторонних разработчиков.
4. У него динамическая типизация (можно передавать в функции любой тип данных, предварительно его не указывая).
5. Он интерпретируемый, что позволяет раньше находить ошибки в ПО (до его полной сборки в полноценное приложение).
6. Также Python славится своей системой оповещения об ошибках. Язык очень ясно дает понять, где и почему возникла ошибка. В отличие от того же C++.

Недостатки

1. Python медлительный. Он способен на многое, но это многое он делает медленнее конкурентов.
2. Python не особо адекватно распоряжается памятью и часто «съедает» больше ресурсов, чем для решения тех же задач использовали бы JavaScript или C.
3. Python строго привязан к системным библиотеками. Отсюда возникают сложности при попытке использовать язык на новых программных платформах.

1.7. Обоснование выбора сред разработки графического интерфейса

**PyQt**

PyQt- это графический пользовательский интерфейс (GUI) для Python. Он очень популярен среди разработчиков, и графический интерфейс может быть создан с помощью кодирования или QT-дизайнера. Фреймворк разработки QT-это визуальный фреймворк, который позволяет перетаскивать виджеты для создания пользовательских интерфейсов.

PyQt-это зрелый набор привязок Python к Qt для кросс-платформенной разработки настольных приложений. Он предлагает богатый выбор встроенных виджетов и инструментов для создания пользовательских виджетов для формирования сложных графических интерфейсов, а также надежную поддержку баз данных SQL для подключения к базам данных и взаимодействия с ними.

**Tkinter**

Tkinter – одна из самых популярных библиотек графического интерфейса Python для разработки настольных приложений. Это комбинация стандартного графического интерфейса TK и python.

Tkinter предоставляет различные виджеты , такие как метки , кнопки, текстовые поля, флажки, которые используются в приложении с графическим пользовательским интерфейсом.

Виджеты управления кнопками используются для отображения и разработки приложений, в то время как виджет canvas используется для рисования фигур, таких как линии, полигоны, прямоугольники и т. Д. В приложении. Кроме того, Tkinter-это встроенная библиотека для Python, поэтому вам не нужно устанавливать ее, как другие графические фреймворки.

**Kivy**

**Kivy** - это библиотека Python с открытым исходным кодом для быстрой разработки приложений, использующих инновационные пользовательские интерфейсы, такие как мультитач-приложения.

Kivy работает на Linux, Windows, OS X, Android, iOS и Raspberry Pi. Вы можете запустить один и тот же код на всех поддерживаемых платформах. Он может изначально использовать большинство входов, протоколов и устройств, включая WM\_Touch, WM\_Penn, трекпад Mac OS X и Magic Mouse, Netdev, ядро Linux HID.

Kivy бесплатен в использовании по лицензии MIT.

Инструментарий профессионально разработан, поддерживается и используется. Вы можете использовать его в коммерческом продукте. Фреймворк стабилен и имеет хорошо документированный API, а также руководство по программированию, которое поможет вам начать работу.

Графический движок Kivy построен на OpenGL ES 2, используя современный и быстрый графический конвейер.

Инструментарий поставляется с более чем 20 виджетами, все они очень расширяемы. Многие части написаны на языке Си с использованием Cython и протестированы с помощью регрессионных тестов

**wxPython**

wxPython - это кроссплатформенный графический инструментарий для языка программирования Python.

Это позволяет программистам на Python создавать программы с надежным, высоко функциональным графическим пользовательским интерфейсом, просто и легко. Он реализован в виде набора модулей расширения Python, которые обертывают компоненты графического интерфейса популярной кроссплатформенной библиотеки wxWidgets, написанной на C++.

wxPython-это кроссплатформенный инструментарий. Это означает, что одна и та же программа будет работать на нескольких платформах без изменений. В настоящее время поддерживаются следующие платформы: Microsoft Windows, Mac OS X и macOS, а также Linux.

1.8. Обоснование выбора сред разработки

Данный выбор оказывает значительное влияние на удобство разработки, имеющийся инструментарий, возможности отладки и развертывания приложений на различных платформах, эффективность и качество реализуемых приложений.

В вопросе разработки приложения необходимо учесть, что разработка предполагается на языке Python. Самыми популярными являются следующие IDE: Visual Studio Code, PyCharm и Visual Studio.

**Visual Studio Code**

Это среда разработки, которая была разработана компанией Microsoft для Windows, Linux и macOS. Позиционируется как легкий редактор кода для кроссплатформенной разработки веб- и облачных приложений. Включает в себя отладчик, инструменты для работы с Git, подсветку синтаксиса с помощью технологии IntelliSense и средства рефакторинга. Имеет широкие возможности для кастомизации: пользовательские темы, сочетания клавиш, файлы конфигурации.

Многие возможности Visual Studio Code недоступны через графический интерфейс, зачастую они используются через палитру команд или JSON-файлы. Палитра команд представляет собой подобие командной строки, которая вызывается сочетанием клавиш.

Для сохранения прогресса работы было необходимо использование системы Git. В качестве средства ведения такого репозитория был выбран GitHub за свои особенности:

* Система представляет собой своеобразную социальную сеть для разработчиков, в которой можно хранить код, общаться с коллегами программистами и анализировать их деятельность, отслеживать ошибки.
* Добавлен инструмент управления проектами, сделанный на основе техники Канбан-досок.
* В сервисе есть возможность добавлять и утверждать новый запрос непосредственно в коде.
* На самой доске можно перетаскивать карточки задач, багов или заметок.
* GitHub позволяет вносить изменения в строки.
* На любой стадии процесса можно откатить приложение и запустить его локально.
* Доступна опция истории в листе коммитов, и нет необходимости пролистывать каждый из них, когда нужно выяснить момент добавления кода.
* GitHub предоставляет информацию о том, кто из участников процесса изменил линию файла.
* Сервис абсолютно бесплатен для проектов с открытым исходным кодом и предоставляет им все возможности (включая SSL), а для частных проектов предлагаются различные платные тарифные планы.
* Кроме размещения кода, участники могут общаться, комментировать правки друг друга, а также следить за новостями знакомых.
* С помощью широких возможностей Git программисты могут объединять свои репозитории — GitHub предлагает удобный интерфейс для этого и может отображать вклад каждого участника в виде дерева.
* Для проектов есть личные страницы, небольшие Вики и система отслеживания ошибок.
* Прямо на сайте можно просмотреть файлы проектов с подсветкой синтаксиса для большинства языков программирования.
* Есть возможность прямого добавления новых файлов в свой репозиторий через веб-интерфейс сервиса.
* Код проектов можно не только скопировать через Git, но и скачать в виде обычных архивов с сайта. (Для этого достаточно добавить /zipball/master/ в конец адресной строки.)
* Кроме Git, сервис поддерживает получение и редактирование кода через SVN и Mercurial.
* На сайте есть pastebin-сервис gist.github.com для быстрой публикации фрагментов кода.

**Pycharm**

PyCharm — интегрированная среда разработки для языка программирования Python. Предоставляет средства для анализа кода, графический отладчик, инструмент для запуска юнит-тестов и поддерживает веб-разработку на Django.

PyCharm разработана компанией JetBrains на основе IntelliJ IDEA. PyCharm — это кроссплатформенная среда разработки, которая совместима с Windows, macOS, Linux.

Преимущества: это среда разработки для Python с поддержкой всего и вся и хорошим коммьюнити. В ней «из коробки» можно редактировать, запускать и отлаживать Python-код.

Недостатки: PyCharm может медленно загружаться, а настройки по умолчанию, возможно, придётся подкорректировать для существующих проектов.

**Visual Studio**

Visual Studio — полнофункциональная IDE от Microsoft, которая во многом сопоставима с Eclipse. Доступная на Windows и Mac OS, Visual Studio представлена как в бесплатном (Community), так и в платном (Professional и Enterprise) вариантах. Visual Studio позволяет разрабатывать приложения для разных платформ и предоставляет свой собственный набор расширений.

Python Tools for Visual Studio (PTVS) позволяет писать на Python в Visual Studio и включает в себя Intellisense для Python, отладку и другие инструменты.

Преимущества: как и в случае с Eclipse, если у вас уже установлена Visual Studio для других задач, установка PTVS пройдёт без проблем.

Недостатки: как и в случае с Eclipse, Visual Studio будет многовато, если вам нужен только Python. Кроме того, если вы пользуетесь Linux, то Visual Studio для этой платформы нет.

1.9. Выбор средства для взаимодействия с базой данных

**psycopg2**

Psycopg2 - это драйвер базы данных PostgreSQL, он используется для выполнения операций над PostgreSQL с использованием python, он предназначен для многопоточных приложений. SQL-запросы выполняются с помощью psycopg2 с помощью метода execute(). Он используется для выполнения запроса или команды операции с базой данных.

**Peewee**

Peewee – это небольшое ORM, которое в данный момент поддерживает postgresql, mysql и sqlite. Разумеется, это не единственное ORM для разработчиков Python.

На рис. 1.1 представлена архитектура взаимодействия с Peewee

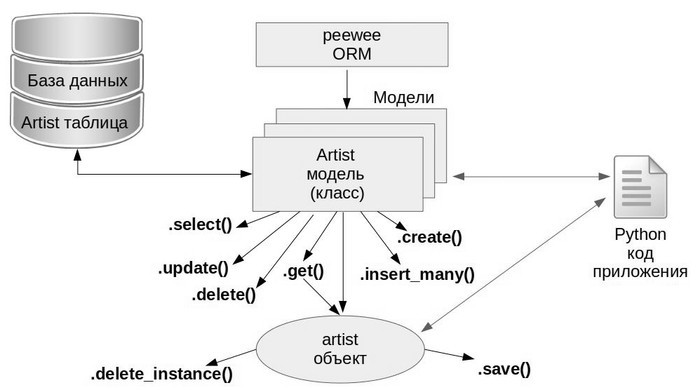


Рисунок 1.1. Архитектура peewee

1.10 Выводы

Таким образом, исходя из требований к реализуемой системе, рассмотрения возможностей наиболее подходящих инструментов, вариантов разработки и последующего их сравнения было решено использовать следующие решения:

В качестве инструментов для разработки было решено использовать: язык программирования – Python, в качестве среды графического интерфейса PyQt, а среда разработки Visual Studio Code

Исходя из требований к системе и характеристик каждой из рассмотренных СУБД в качестве оптимальной была выбрана PostgresSql по следующим причинам:

* Основными требованиями разработчика является надежность и целостность данных.
* Выполнение сложных процедур.
* Работы со сложными структурами данных.

В качестве средств для взаимодействия с БД был использован psycopg2

2. ПРОЕКТИРОВАНИЕ КОМПОНЕНТОВ ПРОГРАММНОГО ПРОДУКТА

2.1. Общие сведения

Данный курсовой проект представляет собой реализацию приложений банковской системы, которая помогает взаимодействовать клиентам и сотрудникам, а также разработку базы данных с использованием СУБД PostgreSQL.

2.2. Разработка структуры системы

Разрабатываемая система состоит из следующих компонентов:

База данных – хранит всю необходимую информацию об имеющимся на предприятии оборудовании.

Клиентская часть – приложение в котором клиент может получить всю необходимую информацию по своим вкладам, выполнить все необходимые действиями со своими счетами.

Часть сотрудников – приложение в котором сотрудник банка способен взаимодействовать с клиентом, создавать новые счета с различными условиями.

2.3. Разработка базы данных

База данных состоит из следующих таблиц:

* accounts
* users
* account\_conditions
* emloyes
* account\_history

Структура базы данных представлена на рисунке 2.1.



**Рисунок 2.1. Структура базы данных**

Ниже приведено описание каждой из таблиц.

| Наименование поля | Тип данных | Признак ключа | Примечание |
| --- | --- | --- | --- |
| id | int | PK | Уникальное значение |
| account\_number | string |  | Уникальное значение.  Хранит в себе номер счета клиента |
| user\_id | int | FK | Уникальное значение  Включает в себя id клиента |
| condition\_id | int | FK | Уникальное значение  Включает в себя id условий по счету |
| is\_money\_in\_account | bool |  | Служит в качестве значения о сохранении денег на счете или возврате на основной счет |
| create\_date | Data |  | Служит в качестве информации о дате создания счета |
| end\_date | Data |  | Служит в качестве информации о дате закрытия счета |
| pin | string |  | Значение pin-кода от счета в банкомате |
| balance | money |  | Служит в качестве информации о количестве денежных средств на счете |

Таблица 2.2.1 – «accounts» - хранит общую информацию о счетах клиента

| Наименование поля | Тип данных | Признак ключа | Примечание |
| --- | --- | --- | --- |
| id | int | PK | Уникальное значение |
| first\_name | string |  | Хранит в себе имя клиента |
| last\_name | string |  | Хранит в себе значение фамилии клиента |
| phone | string |  | Хранит в себе значение номера телефона клиента. Также это значение используется как логин при входе в приложение |
| email | string |  | Хранит в себе значение email телефона клиента |
| password | string |  | Служит в качестве пароля от аккаунта |

Таблица 2.2.2 – «users» - справочник для хранения информации о пользователях

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование поля | Тип данных | Признак ключа | Примечание |
| id | int | PK | Уникальное значение |
| name | string |  | Хранит в себе наименование сотрудника. |
| password | string |  | Хранит в себе пароль сотрудника для входа в приложения. |
| post | string |  | Хранит в себе должность сотрудника |
| phone | string |  | Хранит в себе телефон сотрудника. |
| email | string |  | Хранит в себе email сотрудника. Используется для входа в систему. |

Таблица 2.2.3 – «employs» - справочник для хранения информации о сотрудниках

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование поля | Тип данных | Признак ключа | Примечание |
| id | int | PK | Уникальное значение |
| employ\_id | int | FK | Уникальное значение. Хранит в себе id сотрудника, который создал данное условие |
| name | string |  | Хранит в себе наименование условия |
| min\_sum | money |  | Хранит в себе информацию о минимальной сумме, которая должна храниться на счете |
| count\_month | int |  | Хранит в себе количество месяцев, на которое будет действовать процент |
| percentage | int |  | Хранит в себе процентную ставку, которая будет использоваться в течение count\_month месяцев |
| is\_replenishment | bool |  | Хранит в себе значение отвечающее за возможность пополнение денежными средствами во время действия счета |
| is\_withdrawal | bool |  | Хранит в себе значение отвечающее за возможность снятие денежных средств во время действия счета |
| is\_valid | bool |  | Хранит в себе значение отвечающее за активность данных условий и можно ли его открыт на данный момент |

Таблица 2.2.4 – «account\_conditions» - служит для хранения информации о условиях счета

| Наименование поля | Тип данных | Признак ключа | Примечание |
| --- | --- | --- | --- |
| id | int | PK | Уникальное значение |
| in\_account\_id | int | FK | Уникальное значение. Указывает с каким вкладом было совершенно действие |
| out\_account\_id | int | FK | Уникальное значение. Указывает на то с какого вклада были списаны денежные средства |
| name | string |  | Указывает на действие, которое произошло с вкладом |
| operation | string |  | Наименование операции, которое выполняется по вкладу |
| edit\_time | timestamp |  | Указывает на дату произошедшего действия |
| new\_val | money |  | Указывает на новое значение на счете |
| old\_val | money |  | Указывает на старое значение на счете |

Таблица 2.2.5 – «account\_history» - служит для хранения информации о действиях по счетам

* 1. Расчет денежных средств на счете

Расчет денежных средств на счете производиться исключительно при помощи бизнес-правила. Бизнес-правила представляют собой механизмы управления БД и предназначены для поддержания БД в целостном состоянии. Используя данный механизм уменьшается шанс ошибки со стороны человека

* + 1. Расчет денежных средств при переводе

При выполнении перевода запускается процедура, которая выполнит пополнение денежными средствами один счет и вычтет такое же денежное кол-во с другого счета

2.4.2 Расчет денежных средств на счетах с пополнением

При пополнении счета денежными средствами до начисления процента, счет увеличиться на N% относительно той цифры, которая была заложена изначально, в начале месяца

2.4.3 Расчет денежных средств на счетах с снятием

При снятии со счета денежных средств в конце месяца происходит начисление на N% на минимальный остаток по счету.

2.4.4 Расчет денежных средств на счету с «Оставить средства на счете»

При пополнении счета денежными средствами в конце месяца процентами по предыдущей сумме. В следующем месяце процент уже будет начисляться на новую сумму с процентом

3. ТЕСТИРОВАНИЕ И ИНТЕГРАЦИЯ КОМПОНЕНТОВ ПРОГРАММНОГО ПРОДУКТА

3.1. Назначение системы

Назначение банковской системы заключается в обеспечении эффективного управления финансовыми процессами в банке и обеспечении удобства использования услуг банка для клиентов. Банковская система также предназначена для обеспечения повышения производительности в работе с данными.

Банковская система включает в себя комплекс программных приложений, которые обеспечивают широкий спектр услуг и функций, таких как управление счетами, просмотр истории и т.д.

Назначение банковской системы заключается в создании среды для управления банковскими процессами, обеспечивая стабильную и безопасную работу банка, удобство и доступность операций для клиентов и достижения конкурентных преимуществ на рынке.

3.2. Требования к аппаратной платформе

Требования к системе

Для работы приложений необходимо, чтобы компьютер обладал следующими минимальными характеристиками:

Минимальные требования для работы пользователя:

— процессор с тактовой частотой не менее 1 ГГц;

— ОЗУ 1 Гб;

— 2 Гбайт свободного пространства на диске;

— Microsoft Windows 7/8/8.1/10

— устройство с выходом в интернет.

Для взаимодействия с приложением необходимо наличие мыши или монитора.

Минимальные требования для работы сервера:

— Microsoft Windows 7 и выше, MaсOS, Linux или FreeBSD;

— процессор с тактовой частотой не менее 1 ГГц;

— ОЗУ 2 Гб;

— 512 МБ свободного дискового пространства для установки СУБД

— дополнительное дисковое пространство для данных, хранимых в базе данных.

3.3. Руководство пользователя

После открытия пользователем приложения перед пользователем отобразиться окно авторизации (рисунок 3.1).

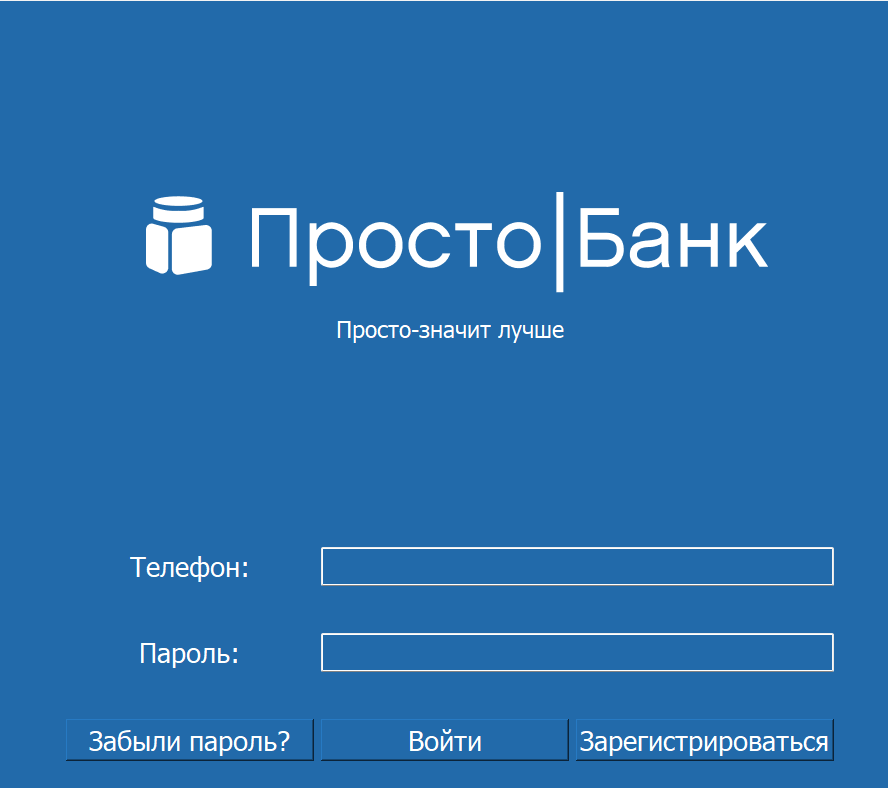


Рисунок 3.1. Демонстрация окна авторизации

Если пользователь заходит впервые, то он может нажать на кнопку «Зарегистрироваться» и тогда пользователь перейдет в окно регистрации, где сможет внести необходимые данные для регистрации (рисунок 3.2)

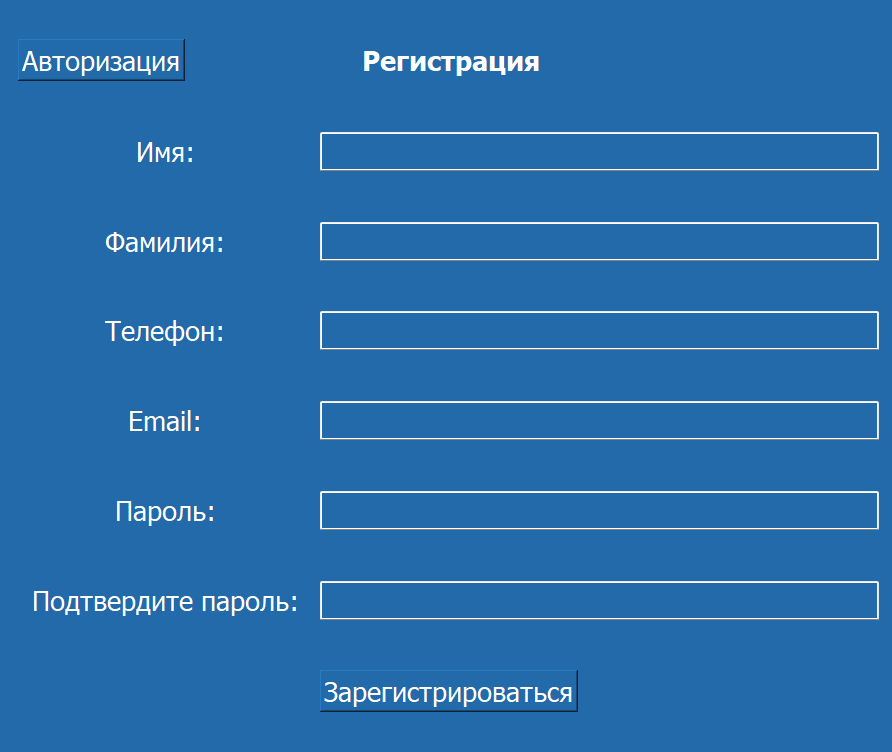
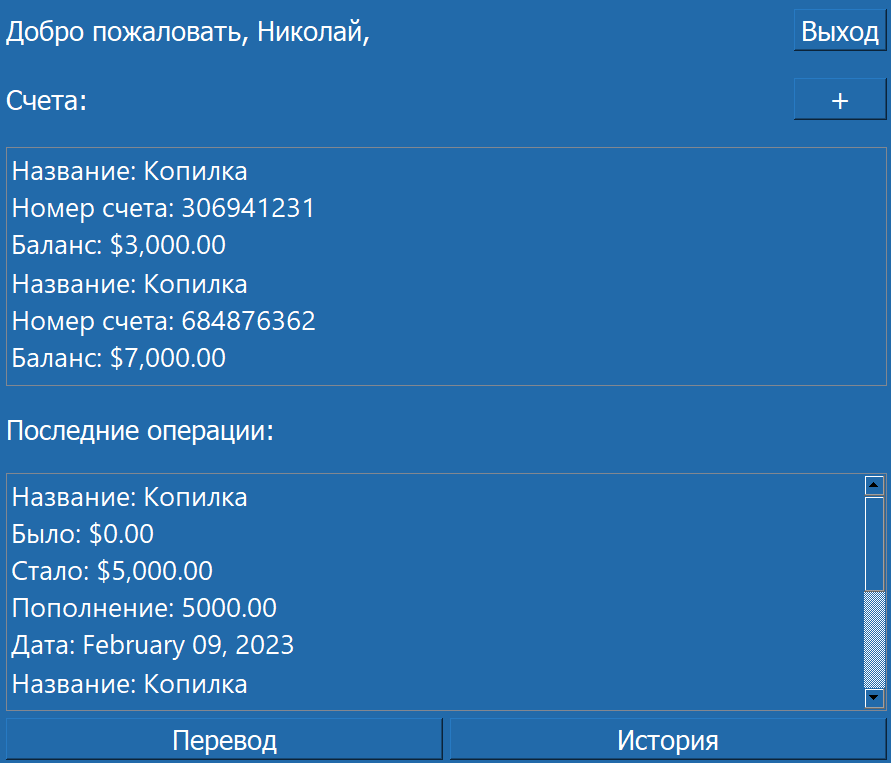


Рисунок 3.2. Демонстрация окна регистрации

После ввода телефона и пароля и нажатием кнопки «Войти» пользователь перейдет в главное окно, где будет предоставлена информация пользователя о его счетах, его история и недавние операции (рисунок 3.3).

 Рисунок 3.3. Демонстрация главного окна пользователя

Если же пользователь впервые заходит в приложения, то ему будет сразу создан счет «Копилка», с которым он сможет функционировать пополнять (рисунок 3.4).

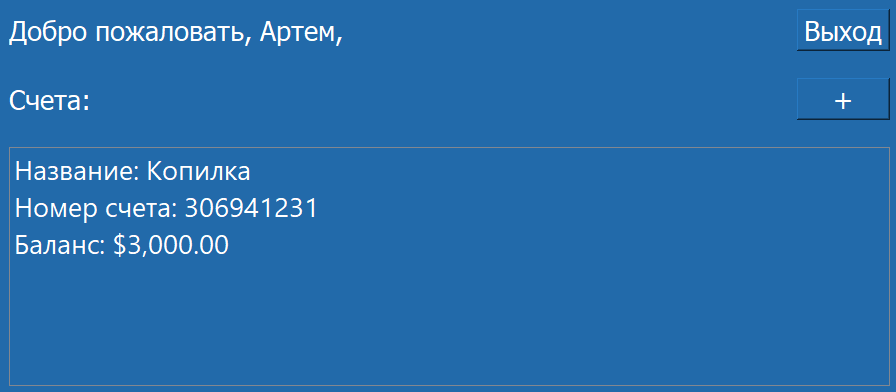


Рисунок 3.4. Демонстрация главного окна пользователя

При нажатии на главном окне пользователя на кнопку «+», пользователь может создать новый счет с выгодным для него предложением по определенным условиям. Доход, который получит клиент по истечению срока счета также отобразиться на окне пользователя (рисунки 3.5 - 3.7)

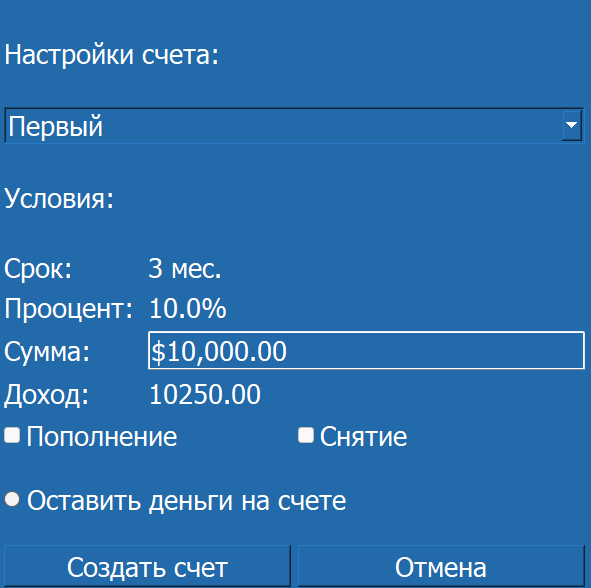


Рисунок 3.5. Демонстрация создания счета «Первый»

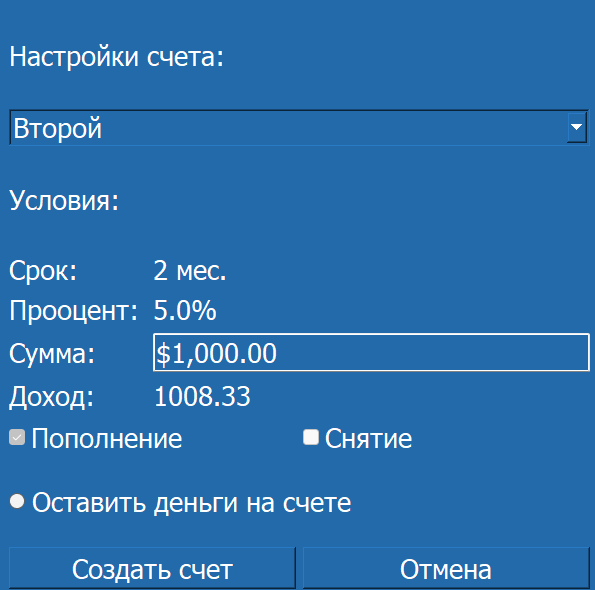


Рисунок 3.6. Демонстрация создания счета «Второй»

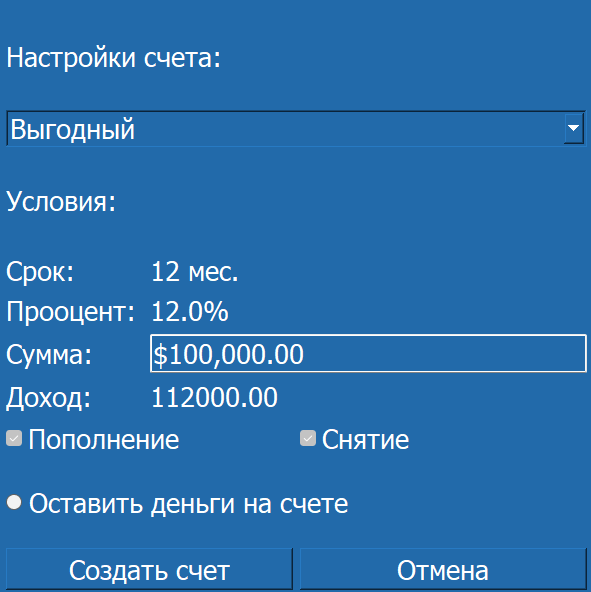


Рисунок 3.7. Демонстрация создания счета «Выгодный»

В главном окне пользователь, нажав на кнопку «История» имеет возможность увидеть историю по данному счету (рисунок 3.8.).



Рисунок 3.8. Демонстрация окна истории

В главном окне пользователь, нажав на кнопку «Перевод» имеет возможность переводить другим пользователям или между своими счетами денежные средства (рисунок 3.9.).

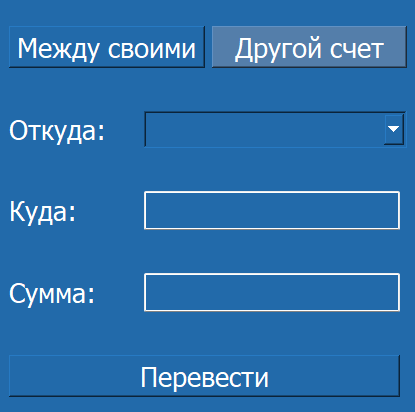


Рисунок 3.9. Демонстрация окна истории

3.4. Руководство администратора

Пройдя окно авторизации, администратор попадет в главное окно, где он имеет возможность перейти во вкладку «Счета» (рисунок 3.10.).

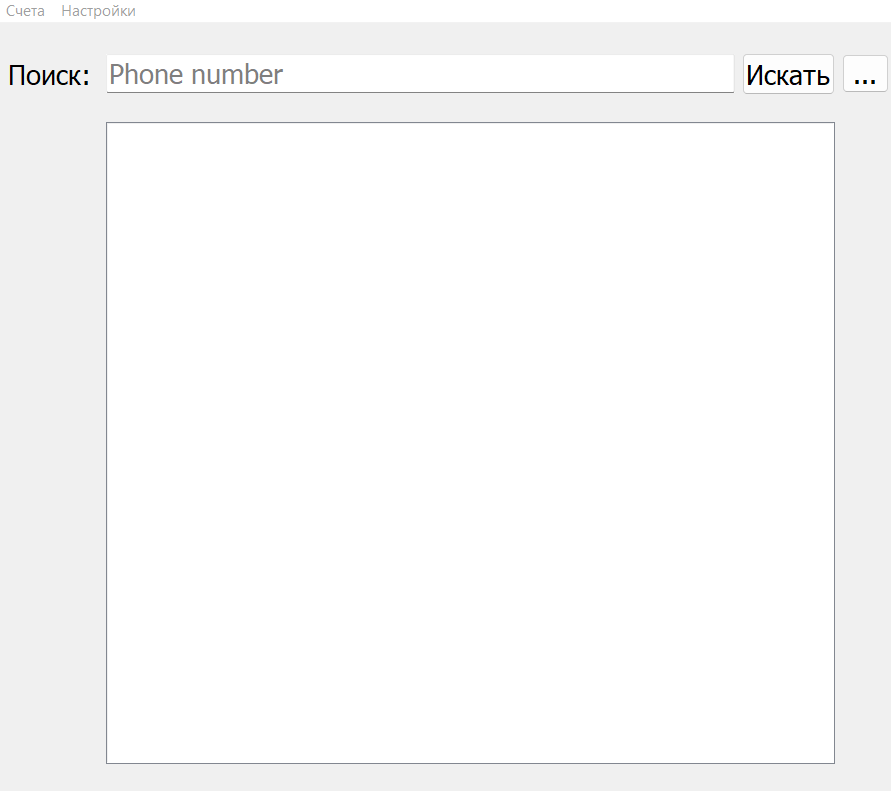


Рисунок 3.10. Демонстрация вкладки счета на главном окне администратора

Перейдя во вкладку счета, администратор попадает в новое окно, где он может создать новые условия для счета (рисунок 3.11.).

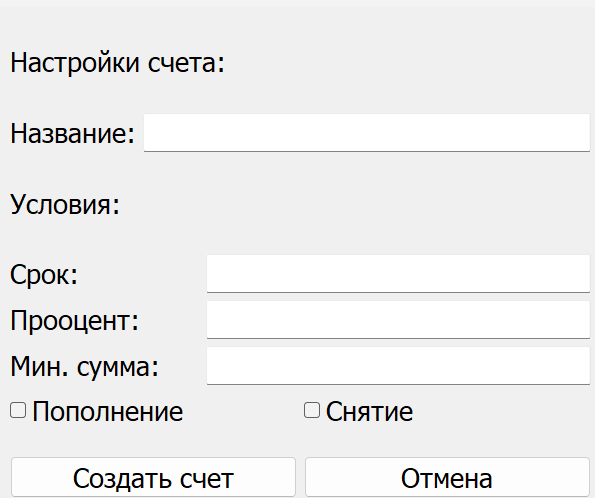


Рисунок 3.10. Демонстрация окна создания счета

ЗАКЛЮЧЕНИЕ

В результате выполнения выпускной квалификационной работы была создана банковская система, взаимодействующий с базой данных и осуществляющий базовые пользовательские функции, такие как: регистрация, авторизация, перевод и создание новых счетов со стороны пользователя, а со стороны администратора возможность создавать новые условия для счетов. Результат соответствует всем заранее определенным требованиям технического задания. Разработанная система обладает гибкостью, многофункциональностью и удобством в эксплуатации.

Поскольку поставленная задача была выполнена в полном объеме. Планируется дальнейшее развитие этого проекта, а именно возможность оплаты с помощью QR-кода. Также планируется расширить возможности, которые будет иметь администратор.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

**Основная литература**

1. Романенко, В.В. Объектно-ориентированное программирование : учебное пособие / В.В. Романенко ; Министерство образования и науки Российской Федерации, Томский Государственный Университет Систем Управления и Радиоэлектроники (ТУСУР). - Томск : Томский государственный университет систем управления и радиоэлектроники, 2014. - 475 с. : ил. - Библиогр.: с. 442. ; То же [Электронный ресурс]. - URL: <http://biblioclub.ru/index.php?page=book&id=480517>.
2. Николаев, Е.И. Объектно-ориентированное программирование : учебное пособие / Е.И. Николаев ; Министерство образования и науки Российской Федерации, Федеральное государственное автономное образовательное учреждение высшего профессионального образования «Северо-Кавказский федеральный университет». - Ставрополь : СКФУ, 2015. - 225 с. : ил. - Библиогр. в кн. ; То же [Электронный ресурс]. - URL: <http://biblioclub.ru/index.php?page=book&id=458133>.

**Дополнительная литература**

1. Гагарин Ю.Е. Применение языка SQL в MS Access: учебно-методическое пособие. – М.: МГТУ им. Н.Э. Баумана, 2012.
2. Генник Д. SQL. Карманный справочник. Москва: Рид Групп,   
   2012, с. 256.
3. Голицина О.Л., Максимов Н.В., Попов И.И. Базы данных: учеб. пособие. – М.: Форум: Инфра-М, 2007.
4. Давыдова, Е.М. Базы данных [Электронный ресурс]: учеб. пособие / Е.М. Давыдова, Н.А. Новгородова. — Электрон. дан. — Москва: ТУСУР, 2007. — 166 с. — Режим доступа: <https://e.lanbook.com/book/11636>. — Загл. с экрана
5. Карпова, Т.С. Базы данных: модели, разработка, реализация: учебное пособие / Т.С. Карпова. - 2-е изд., исправ. - Москва: Национальный Открытый Университет «ИНТУИТ», 2016. - 241 с. : ил. ; То же [Электронный ресурс]. - URL: <https://biblioclub.ru/index.php?%20page=book&id=429003>
6. Кручинин В.В. Разработка сетевых приложений. Томск: Томский государственный университет систем управления и радиоэлектроники, 2013, с. 120 c. URL: <http://www.iprbookshop.ru/72174.html>
7. Мейер, Б. Объектно-ориентированное программирование и программная инженерия [Электронный ресурс] / Б. Мейер. — 3-е изд. — Электрон. текстовые данные. — М. : Интернет-Университет Информационных Технологий (ИНТУИТ), Ай Пи Эр Медиа, 2019. — 285 c. — 978-5-4486-0513-0. — Режим доступа: <http://www.iprbookshop.ru/79706.html>
8. Стасышин В.М. Проектирование информационных систем и баз данных. Учебное пособие. Новосибирск: Новосибирский государственный технический университет, 2012, с. 100. URL: <http://www.iprbookshop.ru/45001.html>
9. Харрингтон, Д. Проектирование объектно ориентированных баз данных [Электронный ресурс] — Электрон. дан. — Москва: ДМК Пресс, 2007. — 272 с. — Режим доступа: <https://e.lanbook.com/book/1231>. — Загл. с экрана