**Содержание**

Введение 4

1 Теоретическая часть 5

1.1 Основные понятия ООП 5

1.2 Постановка задачи и методы её решения 9

1.3 Сервисы, работающие благодаря API 10

2 Практическая часть 15

2.1 Алгоритм решения задачи 15

Введение 4

Введение 4

Введение 4

Введение 4

Введение 4

## Введение

API, или интерфейс прикладного программирования, играет важную роль в современной разработке приложений. Он предоставляет программистам возможность взаимодействовать с внешними сервисами, использовать их функциональность и данные без необходимости знания тонкостей внутренней реализации. Использование API функций в приложении визуальной среды открывает новые возможности для разработчиков, упрощая процесс создания приложений и расширяя их функциональность. Благодаря API, приложения могут интегрировать различные сервисы, получать доступ к расширенным возможностям и повышать удобство использования для конечного пользователя. Таким образом, понимание и эффективное использование API функций визуальной среды становится важным аспектом для современных разработчиков приложений.

Развитие современных технологий и постоянное расширение функциональности визуальных сред разработки приводят к увеличению спроса на инструменты, позволяющие максимально эффективно использовать API функции в создаваемых приложениях. Важность этого процесса заключается в возможности интеграции различных сервисов и возможностей в приложениях, что способствует расширению их функциональности и удобства использования. Таким образом, использование API функций в визуальной среде представляет собой важный аспект современной разработки приложений, обеспечивая программистам возможность создавать более мощные, гибкие и интегрированные приложения для различных целей и платформ.

**1 Теоретическая часть**

## 1.1 Основные понятия ООП

История становления программирования богата какими-либо нововведениями, которые коренным образом меняют всю предшествующую систему. Одним из первых и наиболее ярких таких открытий можно считать выход в свет книги «Программирование без GOTO». Следующим наиболее важным шагом по праву считают переход к процедурно-функциональному программированию, когда программы разбивались на ряд независимых блоков, и в итоге просто грамотно соединялись в единое целое. Ну и третьим, наверное, самым большим открытием можно считать появление объектно-ориентированного программирования. ООП объединило в себе одновременно процедурные методы, для чего создавались самостоятельные модули, структурное проектирование и стимулировало творческий подход к созданию программ.

Сегодня ни одна программа не представляется без наличия в ней объектов. Объектно-ориентированное программирование вышло на новую ступень своего развития, когда внедрить объект в программу не составляет большого труда, а порой, это за вас сделает визуальная среда. Таким образом, такой мощный инструмент, как объект стал доступным даже для начинающего программиста. Теперь любой программист может использовать всю мощь и эффективность объектно-ориентированного программирования.

Неформально можно сказать, что объектно-ориентированное программирование стоит на трёх китах, таких как инкапсуляция, полиморфизм и наследование. Что же это такое?

Инкапсуляция – это процесс отделения друг от друга элементов объекта, определяющих его устройство и поведение; инкапсуляция служит для того, чтобы изолировать обязательства абстракции от их реализации.

Наследование означает такое отношение между классами (отношение родитель - потомок), когда один класс заимствует структурную или функциональную часть одного или нескольких других классов (соответственно, одиночное и множественное наследование).

Полиморфизм: одно и то же имя может означать объекты разных типов, но, имея общего предка, все они имеют и общее подмножество операций, которые можно над ними выполнять.

Главным звеном программы в ООП является объект (например класс).Главным в понятии объекта является объединение идей абстракции данных и алгоритмов. Объекты обладают целостностью, которая не может быть нарушена. Свойства, которые характеризуют объект и его поведение, остаются неизменными.

Объектно-ориентированное проектирование – это методология программирования, основанная на представлении программы в виде совокупности объектов, каждый из которых является экземпляром определенного класса, а классы образуют иерархию наследования.

Для ООП очень важное значение имеет понятие абстрагирования.

Абстракция выделяет существенные характеристики некоторого объекта, отличающие его от всех других видов объектов и, таким образом, четко определяет его концептуальные границы с точки зрения наблюдателя.

Абстрагирование концентрирует внимание на внешних особенностях объекта и позволяет отделить самые существенные особенности поведения от несущественных.

Класс делится на две части: интерфейс и реализацию.

Интерфейс отражает внешнее поведение объекта, описывая абстракцию поведения всех объектов данного класса.

Внутренняя реализация описывает представление этой абстракции и механизмы достижения желаемого поведения объекта.

Для удобного представления программы используются модульность и иерархия. Модульность – это разделение программы на фрагменты, которые компилируются по отдельности, но могут устанавливать связи с другими модулями. Иерархия – это упорядочение абстракций, разложение их по уровням.

Сохраняемость – способность объекта существовать во времени, переживая породивший его процесс, и (или) в пространстве, перемещаясь из своего первоначального адресного пространства. Говоря другими словами, сохраняемость – это способность классов использоваться не только в одной программе или части программы.

Программы в ООП должны быть написаны на соответствующем объектно-ориентированном языке программирования.

По каким же критериям определяется такой язык?

1. все объекты имеют свой специализированный тип - класс;
2. каждый класс должен иметь различные виды полей данных, в зависимости от степени доступа: закрытые, открытые, наследованные;
3. классы могут наследовать часть полей данных и методов у других классов;
4. методы представляются в виде именованных операций, что придает программе читабельность.

С++ — это попытка решения разработчиками языка С задач объектно-ориентированного программирования. С++ был разработан сотрудником исследовательской лаборатории компании AT&T Бьерном Страуструпом (Bjarn Stroustrup) в 1980 году. В своих исторических замечаниях Страуструп поясняет, почему в качестве базового языка был выбран С:

* многоцелевой, лаконичный и относительно низкого уровня;
* отвечает большинству задач системного программирования;
* "идет везде и на всем";
* пригоден в среде программирования UNIX.

Первоначальное название "С++ с классами" в 1983 году, по предложению Рика Масситти(Rick Mascitti), было изменено на С++. В этом же году С++ был впервые применен за пределами исследовательской группы. С 1980 года С++ претерпел два существенных изменения: в 1985 и 1990 годах. Первый рабочий проект языка С++ стандарта ANSI (American National Standarts Institute) был представлен в январе 1994 года.

## 1.2 Постановка задачи и методы ее решения

Цель данной курсовой работы заключается в разработке приложения с использованием API функций в приложении визуальной среды. Для успешного выполнения задачи необходимо решить следующие подзадачи:

Изучение API функций визуальной среды: требуется провести исследование и ознакомиться с доступными API функциями визуальной среды, которые можно использовать для создания приложений. Необходимо изучить документацию и примеры использования API функций.

Выбор платформы и инструментов разработки: на этом этапе необходимо выбрать платформу и инструменты разработки, которые лучше всего подходят для создания приложения с использованием API функций визуальной среды. Важно учесть требования проекта и особенности выбранной платформы.

Анализ требований и проектирование приложения: необходимо провести анализ требований к приложению и разработать его архитектуру. Это включает определение функциональности приложения, структуры данных, интерфейса пользователя и других аспектов.

Реализация приложения с использованием API функций: на этом этапе разработчик должен реализовать приложение с использованием выбранного API функций визуальной среды. Это включает написание кода, тестирование и отладку приложения.

Тестирование и оптимизация: после реализации приложения необходимо провести тестирование для проверки его работоспособности и соответствия требованиям. При необходимости можно внести изменения и оптимизировать приложение.

Документирование и представление результатов: В конце работы необходимо составить отчет, в котором описать постановку задачи, методы решения, результаты, а также представить разработанное приложение и его функциональность.

В результате выполнения всех этих шагов будет разработано приложение с использованием API функций визуальной среды, которое будет соответствовать поставленным требованиям.

## 1.3 Сервисы, работающие благодаря API

Аббревиатура API расшифровывается как «Application Programming Interface» (интерфейс программирования приложений, программный интерфейс приложения).

Большинство крупных компаний на определённом этапе разрабатывают API для клиентов или для внутреннего использования. Facebook Messenger -официальный мессенджер от разработчиков крупнейшей в мире социальной сети Facebook. Существуют версии для персональных компьютеров и мобильных устройств. Приложение интегрировано с системой обмена сообщениями на основном сайте Facebook (Facebook Chat) и построено на базе открытого протокола MQTT.

Данный мессенджер позволяет обмениваться файлами и фотографиями. К текстовым сообщениям можно добавлять смайлики и стикеры. Мобильная версия мессенджера предоставляет голосовую и видеосвязь, а также возможность передавать геометки. С июля 2016 года Facebook на своём новостном портале сообщил о запуске тестирования сквозного (End-to-End) шифрования с секретными чатами.

Теперь появилась возможность создавать секретный диалог, содержимое которого доступно исключительно для создателя диалога и его собеседника -пользователя, приглашённого в секретный чат. Также доступна возможность задавать таймер -после истечения заданного времени сообщения удаляются из беседы. Сквозное шифрование в секретных чатах основано на базе протокола Signal, разработанного некоммерческой организацией разработчиков открытого программного обеспечения «Open Whisper Systems».

WhatsApp - бесплатный частный коммерческий кроссплатформенный мессенджер с поддержкой голосовой связи. Позволяет пересылать текстовые сообщения, изображения, видео и аудио. Также существуют версии для

5 персональных компьютеров под управлением ОС Windows и мобильных устройств под управлением ОС Android, iOS, Windows Phone, BlackBerry, Nokia Symbian.

WhatsApp использует модифицированный протокол Extensible Messaging and Presence Protocol (XMPP, ранее известный как Jabber).С апреля 2016 года с выходом обновления версии 2.16.12 в WhatsApp появилось сквозное шифрование (End-to-End) на базе протокола Signal. Шифрование распространяется на все типы сообщений: текст, фото, видео и голосовые сообщения. Шифрование также доступно в групповых чатах.

В реализации используются алгоритмы ECDH на Curve25519, AES-256, AES-GCM, HMAC-SHA256, HDKF.Viber - один из популярнейших мессенджеров, поддерживающих голосовую и видеосвязь. Работает на платформах Android, BlackBerry, iOS, Symbian, Windows Phone, Bada и компьютерах под управлением Windows, OS X и Ubuntu. Позволяет отправлять текстовые сообщения со стикерами, а также нарисованные прямо в приложении картинки. Также в мессенджере есть групповые и публичные тематические чаты.

Начиная с версии 6.0, разработчики Viber внедрили в свой сервис сквозное шифрование. Таким образом, все сообщения, фотографии, видео, телефонные и видео вызовы будут зашифрованы с момента отправки с одного устройства до момента их получения адресатом.

Как заявляют разработчики, ключ, необходимый для расшифровки данных, доступен только на устройстве, используемом для отправки сообщения, и на устройстве, используемом для приёма сообщения. Таким образом, никто другой не может расшифровать передаваемые данные. Групповые чаты также шифруются при условии, что все участники чата имеют версию мессенджера 6.0 или старше.

Telegram – бесплатный кроссплатформенный мессенджер, позволяющий обмениваться текстовыми сообщениями со смайликами стикерами, голосовыми сообщениями, а также различным медиаконтентом.

Поддерживает групповые чаты и публичные каналы. Сервисориентированна приватность – мессенджер гарантирует, что данные надёжно зашифрованы. В настройках безопасности доступна возможность «самоликвидации» аккаунта в случае длительного отсутствия пользователя в сети.

Для мессенджера был создан протокол MTProto, предполагающий использование нескольких протоколов шифрования. При авторизации и аутентификации используются алгоритмы RSA-2048, DH-2048 для шифрования, при передаче сообщений протокола в сеть они шифруются AES с ключом, известным клиенту и серверу.

Также применяются криптографические хеш-алгоритмы SHA-1и MD5.С 8 октября 2013 года добавлена функциональность секретных чатов, в которых реализуется сквозное шифрование с применением алгоритма AES-256 в режиме IGE для пересылаемых сообщений.

В отличие от Viber, в Telegram секретные чаты доступны для двух пользователей. Как и в FacebookMessenger, в секретном чате Telegram есть возможность задания таймера. Спустя указанное время прочитанное сообщение будет удалено на устройстве пользователя.

qTox – свободный кроссплатформенный мессенджер, реализованный на базе протокола Tox. Позволяет осуществлять текстовую, голосовую, видеосвязь и передачу файлов.

Доступен режим конференции(группы). Данный мессенджер отличается от ранее рассмотренных тем, что регистрация пользователей на основном сервере не требуется. После установки qTox автоматически создаётся пара ключей-публичный и секретный. Публичный ключ служит как уникальный идентификатор для поиска собеседника. Секретный ключ хранится только у владельца и подтверждает его подлинность, не раскрывая персональные данные.

Поиск собеседников происходит через DHT. Изначально вся переписка в qTox защищена с использованием сквозного шифрования.

KateMobile-неофициальный мобильный клиент социальной сети «ВКонтакте». Предоставляет все возможности веб-версии ВКонтакте для ведения переписки-обмен текстовой информацией со смайликами стикерами, голосовыми сообщениями другим медиаконтентом, включая файлы допустимых форматов.

KateMobile является тонким клиентом, использующим открытое API, которое предоставляется разработчиками ВКонтакте.

В социальной сети «ВКонтакте» отсутствует шифрование переписки, поэтому она хранится на серверах сети в открытом виде и доступна для прочтения третьим лицам, имеющим доступ к серверам.

**2 Практическая часть**

## 2.1 Алгоритм решения задачи

## Алгоритм решения задачи:

## Изучение API функций визуальной среды

## Прочитать документацию по API функциям визуальной среды.

## Изучить примеры использования API функций.

## Выбор платформы и инструментов разработки:

## Выбрать платформу разработки, поддерживающую использование API функций.

## Установить необходимые инструменты разработки.

## Анализ требований и проектирование приложения:

## Определить функциональность приложения.

## Разработать архитектуру приложения, учитывая использование API функций.

## Реализация приложения с использованием API функций:

## Написать код приложения, включая использование API функций.

## Тестировать приложение на корректность работы с API.

## Тестирование и оптимизация:

## Провести тестирование приложения, включая взаимодействие с API.

## Оптимизировать код при необходимости.

## Документирование и представление результатов:

## Составить отчет о выполненной работе, описать использованные методы и API функции.

## Представить разработанное приложение и его функциональность.

## Представление результатов:

## Подготовить демонстрацию приложения с использованием API функций.

## Представить результаты работы в курсовой работе или презентации.

## 2.2 Диаграмма вариантов использования системы

## Диаграмма вариантов использования системы представляет собой визуальное представление функциональных требований к системе. Она отображает взаимодействие между системой и ее окружением, а также демонстрирует, как различные актеры (пользователи или другие системы) взаимодействуют с системой для достижения определенных целей. Диаграмма представлена на рисунке 1.

Изображение выглядит как текст, круг, диаграмма

Автоматически созданное описание

Рисунок 1 Диаграмма вариантов использования системы

## 2.3 Ход выполнения работы по созданию приложения

Для выполнения работы нужно открыть среду визуального программирования C++ builder embarcadero 2009 года без каких-либо библиотек, работающих с современными стандартами.

Изображение выглядит как снимок экрана, текст, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

Рисунок

После этого мы создаём новый проект, выбирая шаблон VCL Forms Application

Изображение выглядит как текст, снимок экрана, программное обеспечение, дисплей

Автоматически созданное описание

Рисунок

После нажатия ОК, откроется визуальная среда программирования нашим новым проектом. Сразу его сохраняем в нашу рабочую папку для того, чтобы при процессе компиляции, мы могли запустить приложение самостоятельно. Для сохранения нашего проекта мы нажимаем сочетание клавиш на клавиатуре Shift + Ctrl + S. После нажатия данного сочетания клавиш мы выбираем директорию, в которой сохраняем проект.

Изображение выглядит как текст, программное обеспечение, снимок экрана, компьютер

Автоматически созданное описание

Рисунок

После того, как мы сохранили файл, содержащий логику нашего приложения, нам снова открывается диалоговое окно, в котором мы выбираем место сохранения остальных файлов проекта. Мы выбираем предыдущую директорию, в которую мы сохраняли файл, содержащий логику нашего приложения.

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

Рисунок

После этих действий мы можем начинать расставлять визуальные интерактивные объекты, с которыми пользователь будет взаимодействовать при использовании нашего приложения. Для начала разместим поле для редактирования текста Edit. Это поле позволит пользователю самостоятельно вводить текстовые данные. В нашем случае мы размещаем первый Edit для того, чтобы пользователь смог ввести URL, по которому будет делаться запрос.

Изображение выглядит как снимок экрана, текст, Шрифт, логотип

Автоматически созданное описание

Рисунок

После этого мы изменим свойства этого объекта, выделив его левой кнопкой мыши. Установим свойство Text, чтобы по умолчанию при запуске приложения, в этом поле был текст, который мы укажем. Я установил значение URL своей API, чтобы можно было легко производить отладку приложения без лишних телодвижений. Ссылка http://nkse.geardung.space/api/v1/group/all установлена по умолчанию в приложении. Теперь добавим надпись для этого поля, чтобы пользователь понял, для чего это поле ввода. Для этого мы перетаскиваем Label из палитры объектов.

Изображение выглядит как текст, снимок экрана, Шрифт, линия

Автоматически созданное описание

Рисунок

После этого устанавливаем для этого объекта нужные поля. Для начала мы установим свойство Caption. Это свойство отвечает за отображаемое содержимое объекта. Его мы устанавливаем, как endpoint URL. Далее мы растягиваем границы объекта Label1 вдоль объекта Edit1. Это позволит сделать текст по центру. Чтобы текст встал по середине, необходимо изменить свойство alignment на taCenter. После этих действий наше приложение будет иметь вид, изображённый на рисунке ниже.

Изображение выглядит как текст, снимок экрана, Прямоугольник, дисплей

Автоматически созданное описание

Рисунок

Теперь нам нужен объект, в котором мы сможем указывать параметры запроса и их значения. Для этого нам идеально подходит объект ValueListEditor. Объект изображён на рисунке ниже.

Изображение выглядит как текст, снимок экрана, дисплей, программное обеспечение

Автоматически созданное описание

Рисунок

Теперь мы настраиваем свойства этого объекта. В него нам нужно вписывать не только значения, но и названия самих параметров. Для этого мы выделяем объект и изменяем свойство KeyOptions. Там представлены параметры, которые можно разрешить и запретить для значений. Мы указываем keyAdd, чтобы можно было добавлять новые параметры, keyEdit чтобы можно было изменить название параметра, keyDelete, чтобы удалять лишние и пустые поля и keyUnique, которое будет проверять, чтобы параметры были уникальны и не повторялись.

Изображение выглядит как текст, Шрифт, линия, число

Автоматически созданное описание

Рисунок

После этих действий мы изменяем свойство TitleCaptions, которое отвечает за надписи в заголовке этого объекта. Также мы добавим стандартный параметр под названием token, отвечающий за авторизацию приложений при запросе. Чтобы это сделать, мы изменяем свойство Strings.

После этих действий вид нашего приложения изображён на рисунке ниже.

Изображение выглядит как текст, снимок экрана, программное обеспечение, дисплей

Автоматически созданное описание

Рисунок

Теперь мы добавим кнопку для выполнения запроса. Для этого мы добавим в приложение объект Button. Этот объект изображён на рисунке ниже.



Рисунок

Растягиваем кнопку по ширине объекта Edit1, чтобы не создавалось лишнее пустое пространство. Далее мы изменяем свойства кнопки, чтобы она была понятна пользователю. Выделив объект Button1 одним кликом мыши, мы переходим к свойству Caption. Он отвечает за видимую надпись на объекте. Мы изменяем его на «Выполнить запрос»

Теперь нам нужен объект, в который мы можем выводить данные, полученные в результате выполнения запроса по указанному в Edit1 URL с параметрами и значениями, указанными в ValueListEditor1. Для таких потребностей нам подходит объект Memo. Этот объект представлен на рисунке ниже.

Изображение выглядит как снимок экрана, Прямоугольник

Автоматически созданное описание

Рисунок

Растягиваем объект в оставшейся области приложения и устанавливаем свойства. Для начала мы установим свойство, которое запрещает редактирование текста, написанного внутри этого объекта. Устанавливаем право только на чтение содержимого. Для этого мы выбираем свойство ReadOnly и устанавливаем его в положение true. После этого, если текст ответа от API окажется большим, мы добавляем ползунки для перемещения по вертикали и горизонтали текста. Для этого мы изменяем свойство ScrollBars с ssNone на ssBoth. Мы добавили все необходимые для нашего приложения объекты.

Теперь мы компилируем приложение, пока-что без логики. Для этого нажимаем сочетание Shift + F9. После этого в директории нашего проекта появится папка Debug. В ней будет лежать наше приложение. Запущенное приложение на данном этапе изображено на рисунке ниже.

Изображение выглядит как текст, снимок экрана, дисплей, программное обеспечение

Автоматически созданное описание

Рисунок

Теперь мы начинаем писать код нашего приложения. Для начала сделаем так, чтобы при инициализации формы нашего приложения, объект Memo1 устанавливал свой текст как пустой. Это сделано, чтобы при запуске приложения не было видно стандартного текста Memo1. Для того, чтобы изменить код нашего приложения, мы должны снизу переключиться на вкладку Unit1.cpp. По умолчанию, внутри этого файла будет только одна функция, срабатывающаяся при инициализации формы.

В ней мы прописываем Memo1->Text = “”;. Эта строчка кода при инициализации будет очищать поле Memo1. Далее мы должны прописать логику при нажатии кнопки. Для этого мы должны вернуться обратно на вкладку с дизайном приложения. А затем сделать двойной щелчок на кнопку. После этого нам откроется снова код нашего приложения, где автоматически создастся функция, обрабатывающая нажатие кнопки Button1. В ней мы и пропишем логику приложения.

Для начала мы получим полную ссылку запроса. Для этого над понадобится сложить значение Edit1 с параметрами из ValueListEditor1. Объявляем переменную URL с типом UnicodeString. Назначаем переменной URL значение из поля для ввода Edit1. Далее мы создаём цикл for со счётчиком от одного до количества записей в ValueListEditor1. При каждой итерации мы просматриваем каждую запись в ValueListEditor1. Мы проверяем наличие и параметра, и значения. Затем, если все условия соблюдены, мы добавляем к концу переменной URL такую конструкцию: &<параметр>=<значение>.

Таким образом мы получаем полную ссылку для запроса. Теперь нужен объект, который будет выполнять это действие. Мы добавляем объект IdHTTP в любое место нашего приложения. Этот объект невидим.

Теперь мы объявляем переменную result типом UnicodeString. Теперь делаем запрос по готовой ссылке при помощи конструкции IdHTTP1->Get(URL); Это позволит нам сделать запрос на готовую ссылку с параметрами и получить ответ, который мы запишем в переменную result. После того, как мы получим результат, мы его запишем в свойство Text у объекта Memo1.

Теперь наша программа работает и исправно выдаёт ответ.

## 2.4 Отладка приложения

## Для отладки приложения мы воспользуемся, встроенным в c++ builder дебаггером кода.Заключение

Текст

**Приложение А**

**Техническое задание**

Работа выполняется в рамках курсовой работы по модулю «МДК 03.01. «Технология разработки программного обеспечения»».

2 Основание для разработки

2.1 Программа создана в ходе выполнения курсовой работы.

Тема курсовой работы и основание для выполнения программы были утверждены 15.01.2024 г.

2.2 Наименование работы:

«Создание приложения с использованием API функций в приложении визуальной среды».

2.3Исполнители: Студент группы П-41-20, Слюсарь А. Ю.

2.4Соисполнители: нет.

3 Назначение разработки:

Создание приложения для произведения API запросов.

4) Технические требования

4.1) Состав выполняемых функций. Разрабатываемое ПО должно обеспечивать:

- Выполнение запросов по необходимым адресам;

- Сбор параметров для запросов;

Основной режим использования системы — ежедневная работа.

4.2) Требования к надежности.

Для обеспечения надежности необходимо проверять корректность получаемых данных.

4.3) Условия эксплуатации и требования к составу и параметрам технических средств.

Для работы системы должен быть выделен ответственный оператор.

Требования к составу и параметрам технических средств уточняются на этапе эскизного проектирования системы.

4.4) Требования к информационной и программной совместимости.

Программа должна работать на платформах Windows 7/10.

4.5) Требования к транспортировке и хранению.

Система не требует особых условий эксплуатации

Программная документация поставляется в электронном и печатном виде.

4.6) Специальные требования.

Программное обеспечение должно иметь дружественный интерфейс, рассчитанный на пользователя (в плане компьютерной грамотности) средней квалификации.

При поддержке программы, дальнейших расширений функционала не планируются и наращивание системы не ожидается

Язык программирования — С++

5) Требования к программной документации

Основными документами, регламентирующими разработку будущих программ, должны быть документы Единой Системы Программной Документации (ЕСПД): руководство пользователя, руководство администратора, описание применения.

6)Технико-экономические показатели

Эффективность системы определяется удобством проведения тестирования для определения уровня знаний, а также экономической выгодой, полученной от внедрения аппаратно-программного комплекса.

7) Порядок контроля и приемки

После передачи Исполнителем отдельного функционального модуля программы Заказчику, последний имеет право тестировать модуль в течение 7 дней. После тестирования Заказчик должен принять работу по данному этапу или в письменном виде изложить причину отказа от принятия. В случае обоснованного отказа Исполнитель обязуется доработать модуль.

8)Календарный план работы

Таблица А.1 календарный план работы

|  |  |  |  |
| --- | --- | --- | --- |
| Номер этапа | Название этапа | Сроки этапа | Чем оканчивается этап |
| 1 | Изучение предметной области. Проектирование системы. Разработка предложений по реализации системы | 15.01.24-1.02.24 | Предложения по работе системы. Акт сдачи-приемки |
| 2 | Создание программного обеспечения по учету товаров  Внедрение системы для склада | 2.02.24-  25.02.24 | Программный комплекс |
| 3 | Тестирование и отладка модуля. Внедрение системы во всех складах | 26.02.24-1.03.24 | Готовое приложение |

Приложение Б

Дизайн приложения

Изображение выглядит как текст, электроника, снимок экрана, программное обеспечение

Автоматически созданное описание

Рисунок Б1

Приложение В

Программный код

Программный код

//---------------------------------------------------------------------------

#include <vcl.h>

#pragma hdrstop

#include "Unit1.h"

//---------------------------------------------------------------------------

#pragma package(smart\_init)

#pragma link "IdBaseComponent"

#pragma link "IdComponent"

#pragma link "IdHTTP"

#pragma link "IdTCPClient"

#pragma link "IdTCPConnection"

#pragma resource "\*.dfm"

TForm1 \*Form1;

//---------------------------------------------------------------------------

\_\_fastcall TForm1::TForm1(TComponent\* Owner)

: TForm(Owner)

{

Memo1->Text = "";

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::Button1Click(TObject \*Sender)

{

UnicodeString url = Edit1->Text;

for(int x = 1; x < ValueListEditor1->RowCount; x++)

{

if (x == 1) url = url + "?";

if (! (ValueListEditor1->Cells[0][x] != "" & ValueListEditor1->Cells[1][x] != "")) continue;

url = url + "&" + ValueListEditor1->Cells[0][x] + "=" + ValueListEditor1->Cells[1][x];

}

UnicodeString result;

result = IdHTTP1->Get(url);

Memo1->Text = result;

}

//---------------------------------------------------------------------------

**Список используемой литературы**

1. Перлова О.Н. Проектирование и разработка информационных систем : учебник М.: Академия, 2020.- 256 с. Профессиональное образование
2. Поколодина Е.В. Ревьюирование программных модулей :учебник М.: Академия, 2020.-208 с.
3. Федорова Г.Н. Осуществление интеграции программных модулей: учебник М.: Академия, 2019. -288 с.-
4. Зверева В.П. Сопровождение и обслуживание программного обеспечения компьютерных систем :учебник М.: Академия, 2020.- 256с.
5. Федорова Г.Н. Разработка модулей программного обеспечения для компьютерных систем :учебник М.: Академия, 2020. -384 с.
6. Мартишин, С.А.Базы данных. Практическое применение СУБД SQL- и NoSQL- типа для проектирования информационных систем : Учебное пособие. - М.: ИНФРА-М, 2019. - 368 с.
7. Зубкова, Т. М. Технология разработки программного обеспечения: учебное пособие для СПО / Т. М. Зубкова. — Саратов : Профобразование, 2019. - 468 c.
8. Гвоздева, Т.В., Баллод Б.А. Проектирование информационных систем: Учебное пособие  - Ростов н/Д: Феникс, 2020.-508 с.
9. Мезенцев К.Н.. Автоматизированные информационные системы : учебник -  М.: Академия, 2022. - 176 с.
10. Музалевская А.А., Тарасова Н.В. Информатика: Учебник. 2-е изд. - М.: ИНФРА-М, 2021. - 384 с.
11. Стасышин В.М. Разработка информационных систем и баз данных [Электронный ресурс]: учебное пособие для СПО. Электрон.текстовые данные.- Саратов: Профобразование, 2020.- 100 c.
12. Стешин А. И. Информационные системы в организации: учебное пособие - 2-е изд. - Саратов: Вузовское образование, 2020. - 194 c

## Лист замечаний

## Рецензия преподавателя