Министерство образования Республики Беларусь

Учреждение образования

«Белорусский государственный университет информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

Кафедра информатики

Курсовой проект по дисциплине:

«Программирование»

Пояснительная записка к курсовой работе

Тема работы:

«Библиотека мемов»

Исполнитель

студент гр. 653502 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Васильев В.В.

(подпись дата)

Руководитель

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Козуб В.Н.

(подпись дата)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(оценка)

Минск

2017 год

**Введение**

Мемы - крупицы культурной информации, существовавшие с начала времен. С развитием средств массовой информации мемооборот мировой мемооборот значительно вырос. Авторитет мемов признается уже всеми, существует даже специальная дисциплина - мематика.

В этих условиях, когда мемы появляются и уходят в закат так быстро, что за этим сложно уследить, я решил разработать программу, которая навсегда запечатлит мемы и предоставит к ним доступ. В итоге должно получится что-то вроде библиотеки (архива) мемов.

Для хранения информации будет использоваться база данных IBM db2. Для связи с ней - язык запросов sql.

Используемый язык программирования - c++.

**DB2**

**DB2** — база данных, разработанная IBM.

Начиная с версии 10 в комплекте с сервером DB2(который можно бесплатно скачать на сайте IBM) больше не идет Control Center - программа для работы с базой, которая куда более удобна и дружелюбна по сравнению с командной строкой сервера. Выяснив это (далеко не сразу) я решил использовать другой продукт IBM - Data Studio, выполняющий те же функции (хоть и были другие более удобные кандидаты).

Для хранения информации базы данных используют таблицы. DB2 поддерживает многие типа дынных, а также предостовляет очень тонкую настройку параметров хранения.

**Embarcadero RAD Studio XE**

Embarcadero RAD Studio — среда быстрой разработки приложений (RAD) фирмы Embarcadero Technologies. Для разработки я использовал Embarcadero RAD Studio XE8 2015г. Он предоставляет много инструментов для разработки программ.

И в этой среде, кроме базовый DB-компонентов есть интересная библиотека FireDAC, о которой подробнее напишу ниже.

**Структура программы**

**1. Общая схема**

Клиентское приложение прежде всего состоит из двух окон. В первом окне пользователь будет видеть полный список статей, а также инструменты их сортировки. Во втором окне юзер сможет просмотреть выбранную статью.

**2. Первое окно**

Итак в первом окне я использовал такие базовые компоненты как метки (TLabel), редактируемые поля (TEdit), поле со списком (TComboBox), кнопки (TButton), "вжимаемые" кнопки (TSpeedButton); для отображения списка статей был использован TDBGrid - специальный компонент для отображения таблиц баз данных. TDBGrid получает информацию из компонента TDataSource. И вот тут уже было два варианта. Сначала я собирался использовать первый способ подключения к базе данный - через dbExpress компоненты, однако позже заметил компоненты серии FireDAC.

"FireDAC is a Universal Data Access library for developing applications for multiple devices, connected to enterprise databases. With its **powerful universal architecture**, FireDAC enables native **high-speed direct access** from Delphi and C++Builder to InterBase, SQLite, MySQL, SQL Server, Oracle, PostgreSQL, DB2, SQL Anywhere, Advantage DB, Firebird, Access, Informix, DataSnap and more, including the NoSQL Database MongoDB.

FireDAC is a **powerful, yet easy-to-use** access layer that supports, abstracts, and simplifies data access, **providing all the features** needed to build real-world high-load applications. FireDAC provides a common API for accessing different database back-ends, without giving up access to unique database-specific features and without compromising on performance."

- так написано на сайте embarcadero. Эти слова ~~растопили мое сердце~~ убедили меня использовать именно FireDac.

Итак, создав в поле Data Explorer подключение к базе данных через FireDac, выносим его в форму в качестве компонента. Далее добавляем компонент TBDQuery, который отправляет sql запросы. В его свойство Connection помещаем созданное ранее подключение, а также соединяем его с TDataSourse.

При любом изменении параметров фильтра формируется sql запрос (просто строка, складываемая из кусков), который помещается в TBDQuery, которые его выполнит.

Стоит упомянуть "вжимаемые" кнопки, в которые охотно залезают картинки формата bmp.

Из списка статей в компоненте TDBGrid пользователь выбирает нужную и нажимает кнопку. После нажатия открывается второе окно.

**3. Второе окно**

Второе окно включает в себя несколько TDBMemo и TMemo для хранения заголовков и текста, а также TWebBrowser для отображения картинок по URL адресу.

Событие нажатия кнопки в первом окне активирует процесс извлечение информации из базы данных во второе окно. Во втором окне присутствуют те же цепочки компонентов Connection->Query->DataSourse->Информационное поле.

Для возвращения к первому окну присутствует кнопка возврата.

**Хранение данных**

Теперь немного конкретики. Для того, чтобы поместить информацию о мемах в базу данных, их необходимо классифицировать. Я выделил 14 тегов:

1. Вечные - мемы-долгожители.
2. Мертывые - мемы, утратившие актуальность.
3. Мемы из фильмов/сериалов.
4. Мемы с животными.
5. Игровые мемы.
6. Мемы-эмоции.
7. Исторические мемы.
8. Мемы со знаменитыми людьми.
9. Мемы из вирусных роликов.
10. Религиозные мемы.
11. Мемы из аниме.
12. "Золотой фонд мемов" - лучшие мемы (наверное).
13. Мемы-фразы.
14. Мемы-картинки.

Можно заметить, что 1 и 2 теги несовместимы, как и 13 и 14. Я решил не рассматривать аудио- и видео-мемы, т.к. ~~мне лень~~ в моем запасе не так много времени.

Для хранение буду использовать две талицы с одним общим параметром - id. В первой хранятся данные, которые необходимы для сортировки - названия статей, год появления и теги. Во второй находятся данные самих статей. Вышеупомянутый id служит для свзязи двух таблиц.

Для хранения текста использован varchar, для хранения картинок сначала хотел использовать CLOB (Character Large Object), однако позже ~~из-за возникновения маленькой проблемы~~ я перешел на varchar, в котором храниться URL нужной картинки, специально подготовленной и загруженной на какой-нибудь хостинг.

**Выводы**

В данной курсовой работе была разработана библиотека мемов, способная хранить информацию и выдавать её пользователю.

В ходе курсового проектирования были изучены база данных db2, интерфейс IBM Data Sudio для работы с ней, а также специальные библиотеки для работы с базами данных FireDac.

Ошибок в ходе тестирования выявлено не было.

В будущем я бы хотел переосмыслить существующую классификацию мемов, добавить возможность воспроизведения мелодий и видео. Также сейчас интерфейс программы выглядит серым и невзрачным, это нужно исправить.

**Список использованных источников**

* [https://ru.wikipedia.org/](https://ru.wikipedia.org/wiki/Telegram_(мессенджер))
* [https://ibm.com](https://ibm.com/)
* [https://www.embarcadero.com](https://www.embarcadero.com/)
* [https://community.embarcadero.com](https://community.embarcadero.com/)
* [http://www.cyberforum.ru](http://www.cyberforum.ru/)
* [https://www.youtube.com](https://www.youtube.com/)
* [https://github.com](https://github.com/)
* <http://www.sql.ru/>
* [https://www.informit.com](https://www.informit.com/)
* [http://h-l-l.ru](http://h-l-l.ru/)
* [http://cubook.supernew.org](http://cubook.supernew.org/)
* [http://radio-hobby.org](http://radio-hobby.org/)