Семейный альбом

У каждого из нас имеется множество фотографий, хранящих память о наших близких, друзьях, родственниках, памятных событиях и красивых местах, которые мы посетили. Современная техника позволяет получать качественные фотографии с высоким разрешением быстро и легко, к тому же фотокамера теперь всегда под рукой - все современные мобильные телефоны имеют возможность делать снимки. И было бы здорово иметь простую и функциональную систему хранения и просмотра этих фотографий.

В качестве инструмента создания будет использована визуальная среда разработки My Visual Database.

Оглавление

Проектирование

Общее описание

Создаваемая программа позволит не только просматривать ваши фотографии, но и снабжать их описанием, а также тегами - ключами, по сочетанию которых вы сможете быстро находить нужное изображение.

Вторая интересная функция - это составление альбомов, которые представляют собой презентацию: каждый экран может включать в себя несколько фотографий, декоративных изображений и текстовых описаний, которые будут храниться в базе данных. При запуске презентации экраны сменяются с заданным интервалом с применением визуальных переходов. Также можно настроить звуковое сопровождение, а, кликнув по фотографии, можно развернуть её на весь экран.

Бизнес модель

Добавление и хранение изображений: фотографии (файлы в формате jpg), декоративные элементы (файлы в формате png, jpg).

Изображения снабжены признаками: название, текстовое описание, дата создания, звуковое описание.

К изображениям можно прикреплять один или несколько тегов.

Тег имеет следующие свойства: название, категория. Варианты категорий тегов: места, время, персоны, эмоции, декор и т.д.

При добавлении фотографии она копируется в хранилище (специальную папку на диске) и переименовывается. Имя в хранилище совпадает с ID записи в базе.

Для изображения создаётся миниатюра: уменьшенная версия файла, которая используется в режиме просмотра списков изображений. Миниатюра хранится в хранилище, имеет название, совпадающее с названием основного изображения, но с префиксом “м”.

Изображения можно объединять в альбомы. Альбом имеет название и описание. Альбом состоит из страниц.

Страница содержит один или несколько элементов: изображение из базы данных или произвольный текст. Расположение элементов задаётся статически, при создании страницы.

У альбома создаётся миниатюра обложки - изображение первой страницы альбома.

Для каждой страницы альбома создаётся миниатюра, которая используется при редактировании альбома.

Альбом можно просматривать как презентацию: для каждой страницы можно задать время отображения, для альбома можно задать звуковое сопровождение, смена страниц происходит через заданный визуальный эффект.

Для результата поиска изображений можно создавать динамический альбом - презентацию по фиксированному шаблону.

Для изображений можно задавать область подсказки - прямоугольная область, при наведении на которую появляется рамка и надпись. Например, это можно использовать, чтобы обозначать, кто изображен на фотографии. Подсказки отображаются в режиме полноэкранного просмотра отдельного изображения. Область подсказки задается относительно размера изображения, поэтому легко масштабируется вместе с изображением.

Модель данных

При проектировании создание модели данных является самым главным этапом. Для его выполнения иногда используют специальные инструменты проектирования, но в большинстве случаев хватит текстового описания сущностей, а также их свойств. Такое описание легко можно составить в любом текстовом редакторе или набросать карандашом на листе бумаги.

Изображение

- Название
- Описание
- Дата создания

Тег

- Категория тега
- Название

Категория тега

- Название

Теги изображения

- Изображение
- Тег

Альбом

- Название
- Описание
- Звуковое сопровождение (плей-лист)
- Время отображения страницы по умолчанию

Страница альбома

- Альбом
- Порядковый номер
- Время отображения страницы
- Визуальный эффект перехода

Визуальный эффект перехода

- Название

Элемент альбома

- Изображение
- Позиция X
- Позиция У
- Ширина
- Высота
- Текст
- Название шрифта
- Размер шрифта
- Цвет шрифта

Подсказки к изображениям

- Изображение
- Позиция X
- Позиция У
- Ширина
- Высота
- Текст

Представление данных

Главная форма - это форма альбомов.

На форме альбомов расположена область просмотра миниатюр обложек альбомов и панель поиска, панель инструментов с кнопками управления: добавление, редактирование, удаление, переход в режим редактирования базы изображений, запуск альбома в режиме презентации.

На форме базы изображений расположена область просмотра миниатюр изображений и панель поиска, панель инструментов с кнопками управления: добавление, редактирование, удаление, переход в режим работы с альбомами.

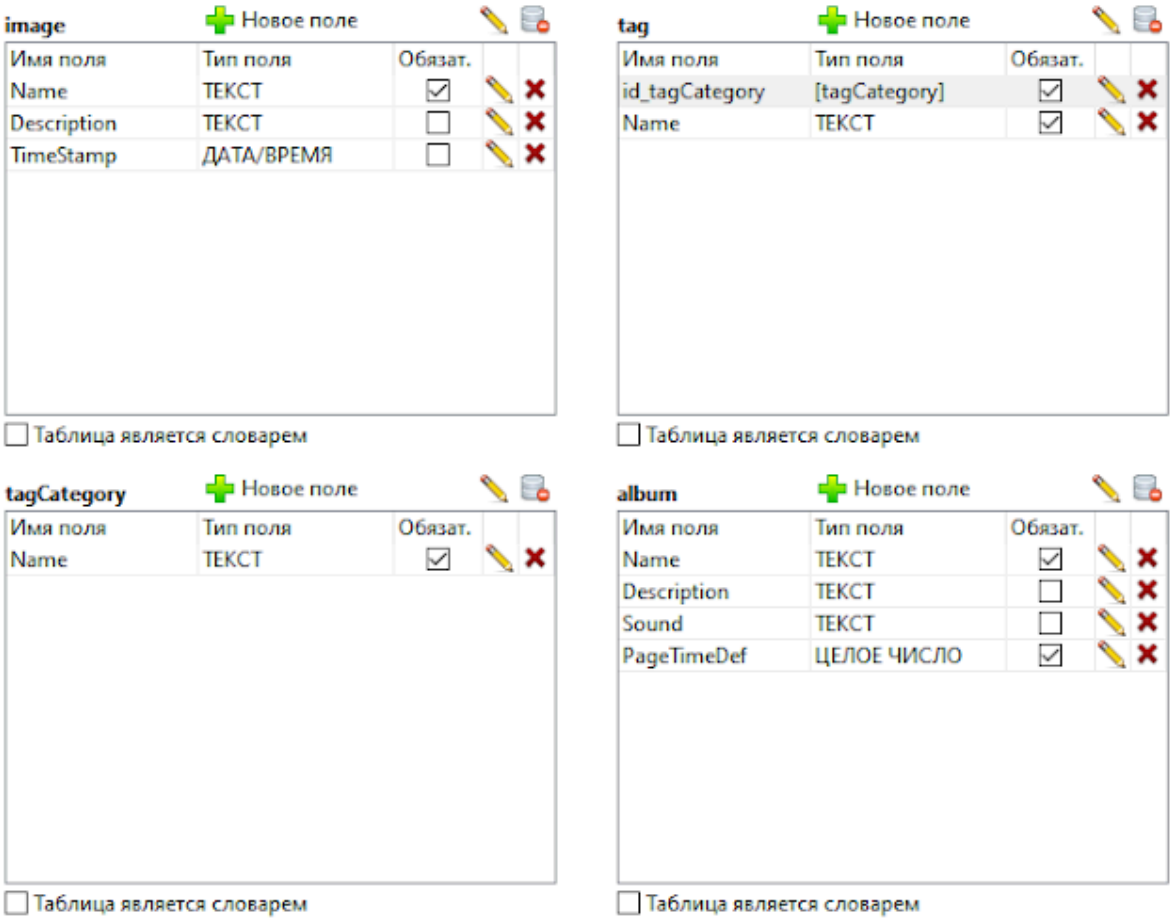
Реализация проекта

My Visual Database (MVDB) имеет удобную визуальную систему создания базы данных (БД). Для её использования не требуется знания SQL.

При создании структуры пригодится ранее созданная модель данных, в которой каждой сущности соответствует таблица БД, а каждому свойству сущности - поле в таблице.

При определенном опыте можно использовать MVDB для проектирования, но, к сожалению, в ней не хватает возможности задавать альтернативные имена таблицам и полям, а также возможности генерации текстового представления структуры БД. Это могло бы пригодиться как при создании документации, так и при генерации форм представления данных.

База данных



page Новое поле

Имя поля	Тип поля	Обязат.
id_album	[album]^	☑
pageNum	ЦЕЛОЕ ЧИСЛО	☑
pageTime	ЦЕЛОЕ ЧИСЛО	☐
id_vfx	[vfx]	☑

☐ Таблица является словарем

vfx Новое поле

Имя поля	Тип поля	Обязат.
Name	ТЕКСТ	☑

☐ Таблица является словарем

pageElement Новое поле

Имя поля	Тип поля	Обязат.
id_page	[page]^	☑
left	ЦЕЛОЕ ЧИСЛО	☑
top	ЦЕЛОЕ ЧИСЛО	☑
width	ЦЕЛОЕ ЧИСЛО	☑
height	ЦЕЛОЕ ЧИСЛО	☑
id_image	[image]	☐
caption	ТЕКСТ	☐
fontName	ТЕКСТ	☐
fontSize	ЦЕЛОЕ ЧИСЛО	☐
fontColor	ЦЕЛОЕ ЧИСЛО	☐

☐ Таблица является словарем

imageTag Новое поле

Имя поля	Тип поля	Обязат.
id_image	[image]^	☑
id_tag	[tag]^	☑

☐ Таблица является словарем

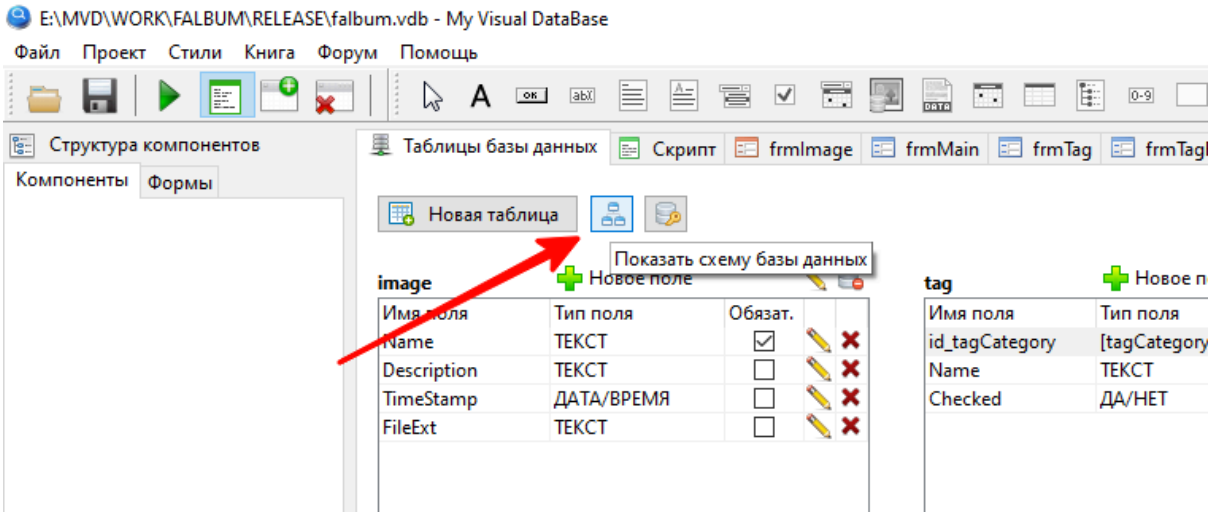
imageHint Новое поле

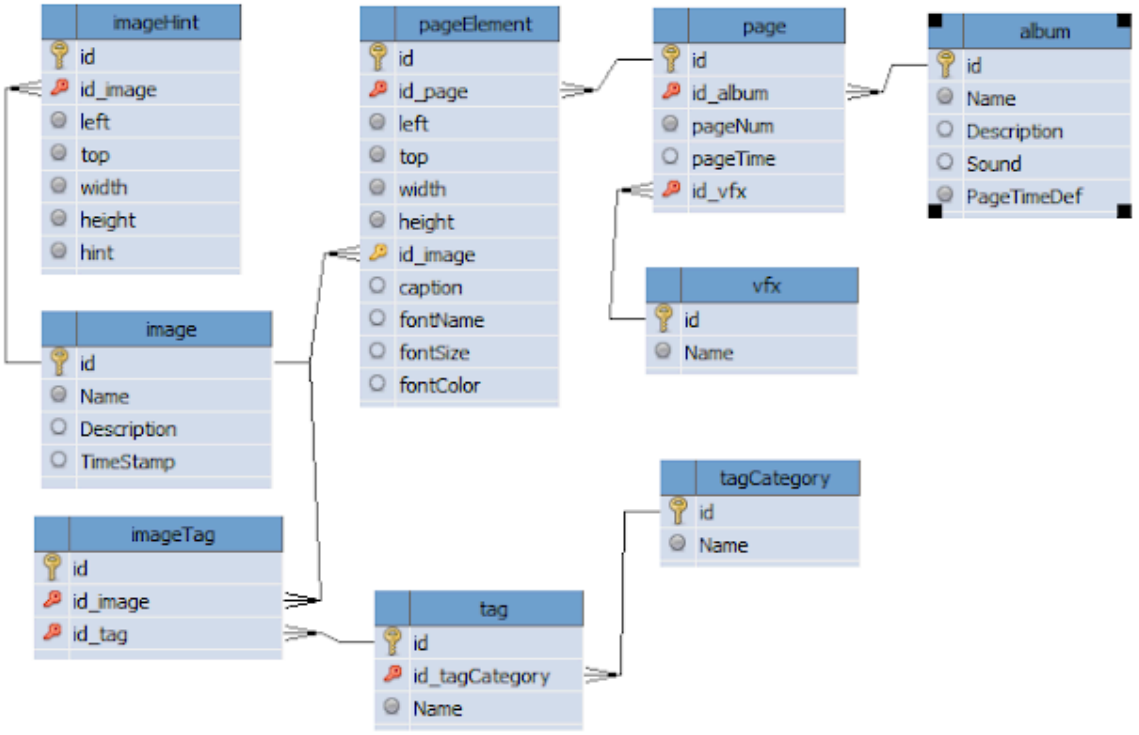
Имя поля	Тип поля	Обязат.
id_image	[image]^	☑
left	ЦЕЛОЕ ЧИСЛО	☑
top	ЦЕЛОЕ ЧИСЛО	☑
width	ЦЕЛОЕ ЧИСЛО	☑
height	ЦЕЛОЕ ЧИСЛО	☑
hint	ТЕКСТ	☑

☐ Таблица является словарем

Схема данных

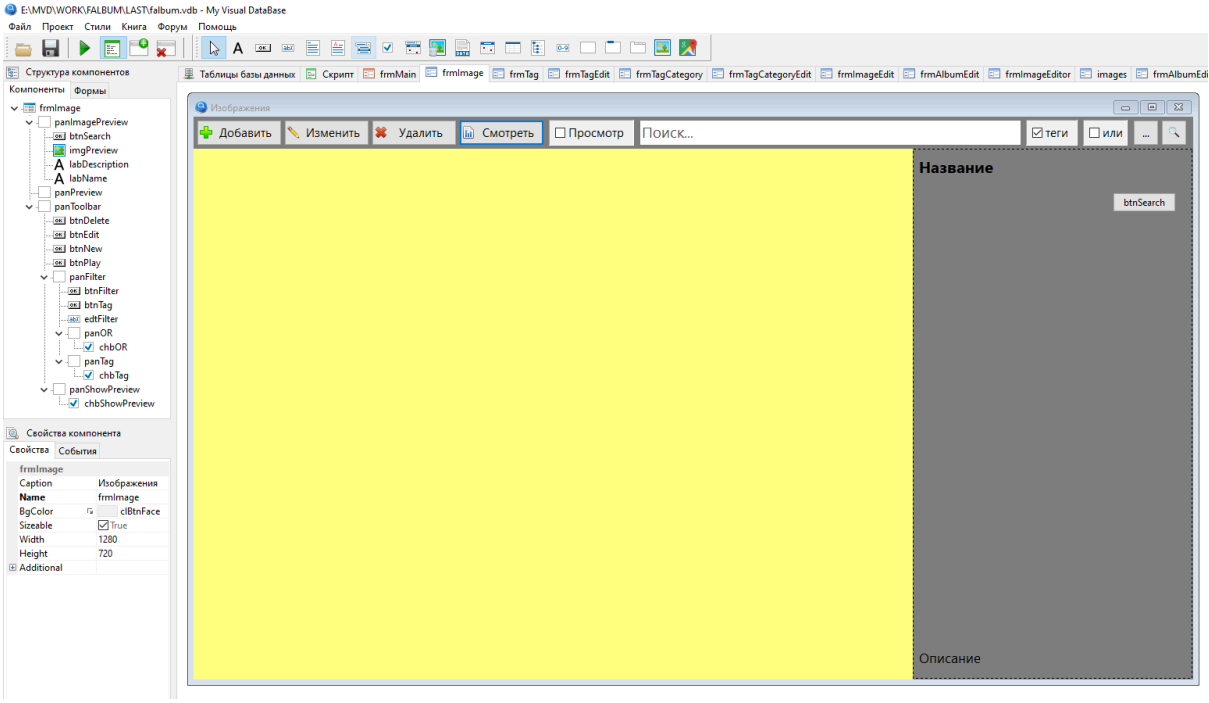
Данное представление удобно для целостного восприятия структуры. Получить её можно одним нажатием кнопки:



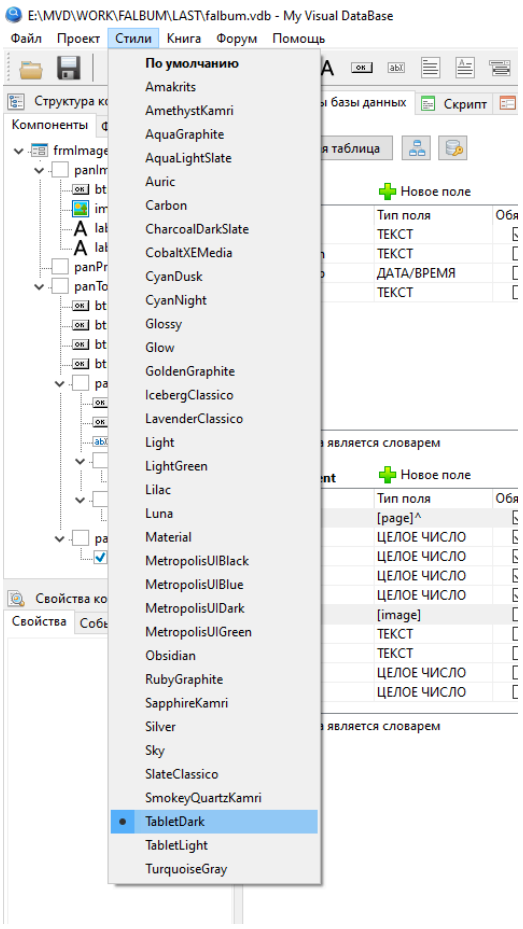


Промежуточный результат

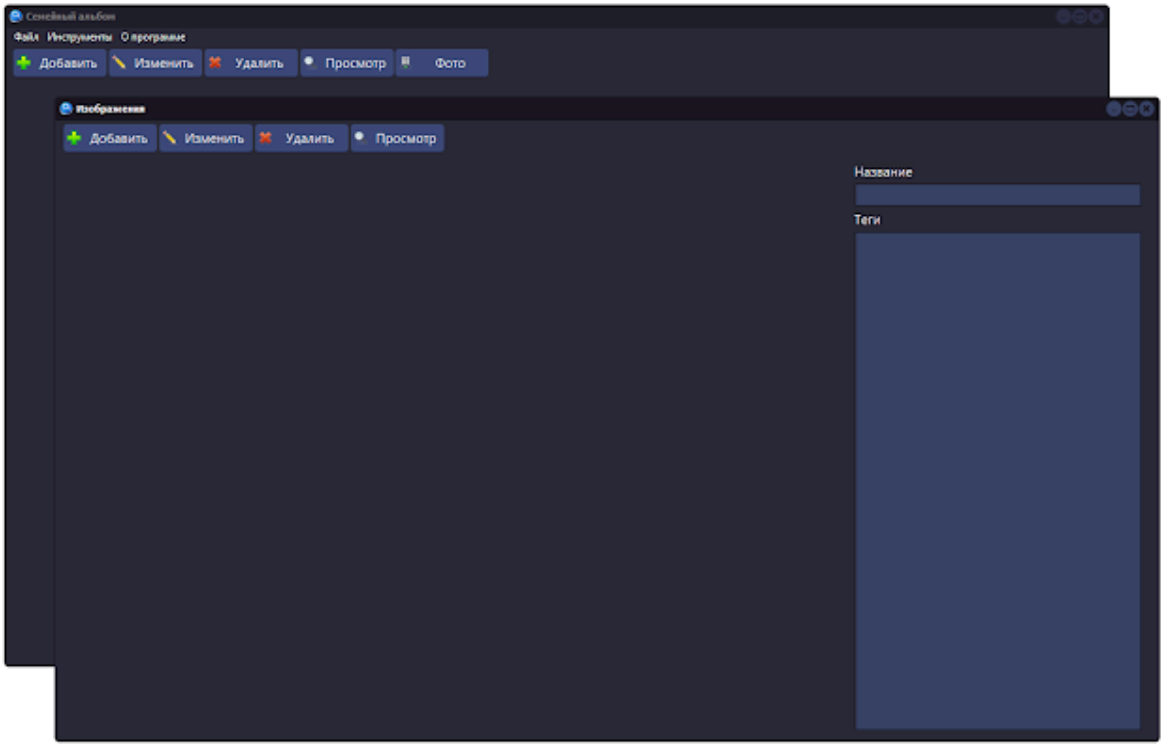
MVDB позволяет быстро получить готовое приложение за счет использования визуальных средств разработки: набора визуальных компонентов, редактируемого визуального представления формы и редактора свойств.



Подробно с компонентами и принципами их использования можно в моей книге “Создание приложений в My Visual Database. Начальный уровень.”, которую можно найти в он-лайн библиотеке “My Visual Database” (https://mvdlibrary.blogspot.com).



Несмотря на то, что у меня имеется ряд наработок для создания пользовательского интерфейса, подробно описанные в книге “Визуальное программирование”, я решил внести разнообразие, использовав встроенную возможность задавать стили оформления приложения. Для этого в проекте будут использованы стандартные компоненты и новые подходы, согласованные с концепцией стилей, сильная сторона которой - многообразие и простота применения - достаточно выбрать нужный стиль в главном меню среды разработки.



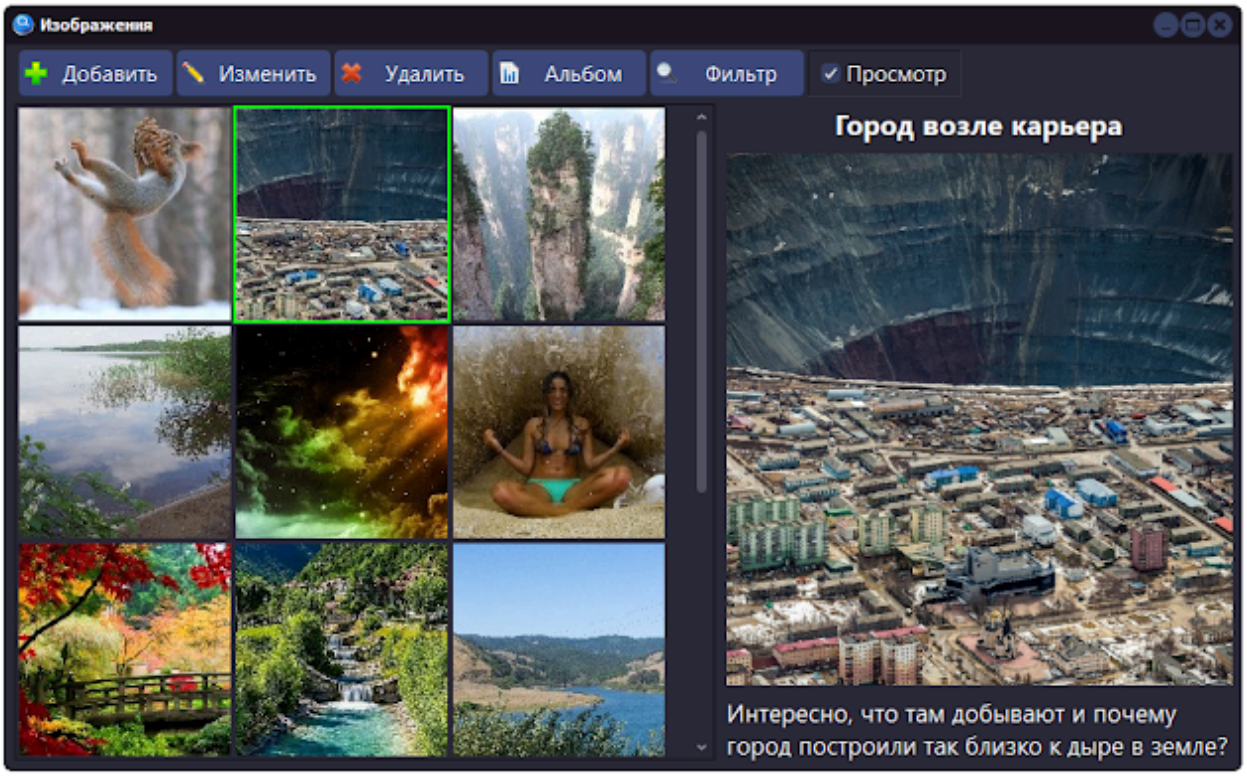
Создание структуры данных и форм в My Visual Database это, несомненно, быстрый и увлекательный процесс, но в нашем случае необходимо реализовать достаточно сложный функционал, а именно:

- создание миниатюр для изображений
- создание процедуры отображения миниатюр
- редактор альбома
- редактор подсказок для изображений

Отображать миниатюры с помощью стандартных элементов интерфейса (таблица) возможно, но мне хочется сделать адаптивное отображение: чтобы количество элементов по горизонтали и вертикали менялось в зависимости от размера формы. Уровень техники стремительно растёт, и разрешение FullHD - это не предел, в распоряжении пользователя может оказаться 2K или даже 4K дисплей. И наоборот - у кого-то в наличии только ноутбук, который едва вытягивает HD.

Редактор альбома по сложности соответствует программе MS PowerPoint, что также может несколько затянуть процесс разработки, поэтому первая версия программы скорей всего выйдет без альбомов, но с полнофункциональным хранилищем изображений и продвинутой системой поиска.

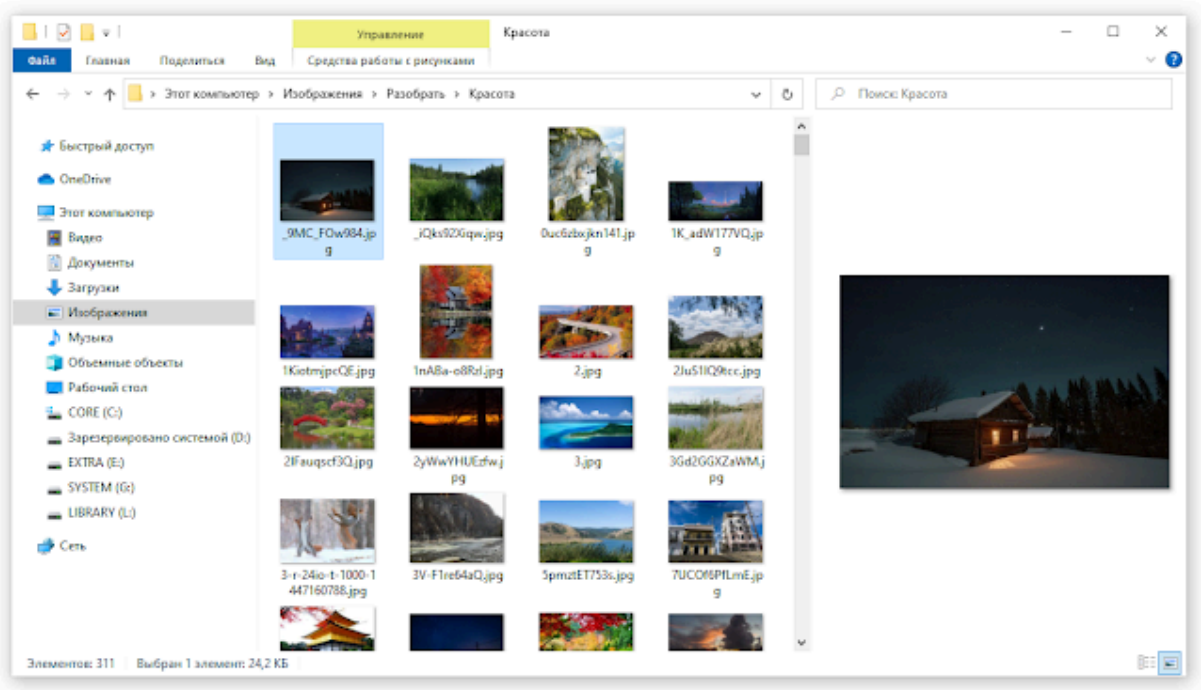
Галерея изображений



Просмотр изображений в виде галереи, состоящей из мини-образов. Такой функционал будет полезным не только в Семейном альбоме, но и в других программах, ориентированных на хранение изображений.

Миниатюры

Принцип работы галереи основан на том, что для каждого изображения создаётся миниатюра - уменьшенная копия изображения с заданным размером. Стандартная программа "Проводник" поступает также: для каждого изображения в папке она автоматически создаёт миниатюру, которые отображает в основной части экрана, если выбран режим просмотра "Крупные значки". А справа может быть отражена область просмотра выбранного изображения, размер которой можно менять.



К сожалению, автоматически создать миниатюры в My Visual Database не получится из-за ограниченных возможностей работы с компонентами изображений. Поэтому миниатюры в “Семейном альбоме” пользователь будет создавать сам, вручную определяя область, которая будет отображаться на уменьшенном изображении. Впрочем, это даже хорошо, так как галерея должна состоять из квадратных изображений, а исходные картинки могут быть с любым соотношением сторон. Поэтому только пользователь может решить, что же именно должно отображаться на миниатюре.

Хранение данных

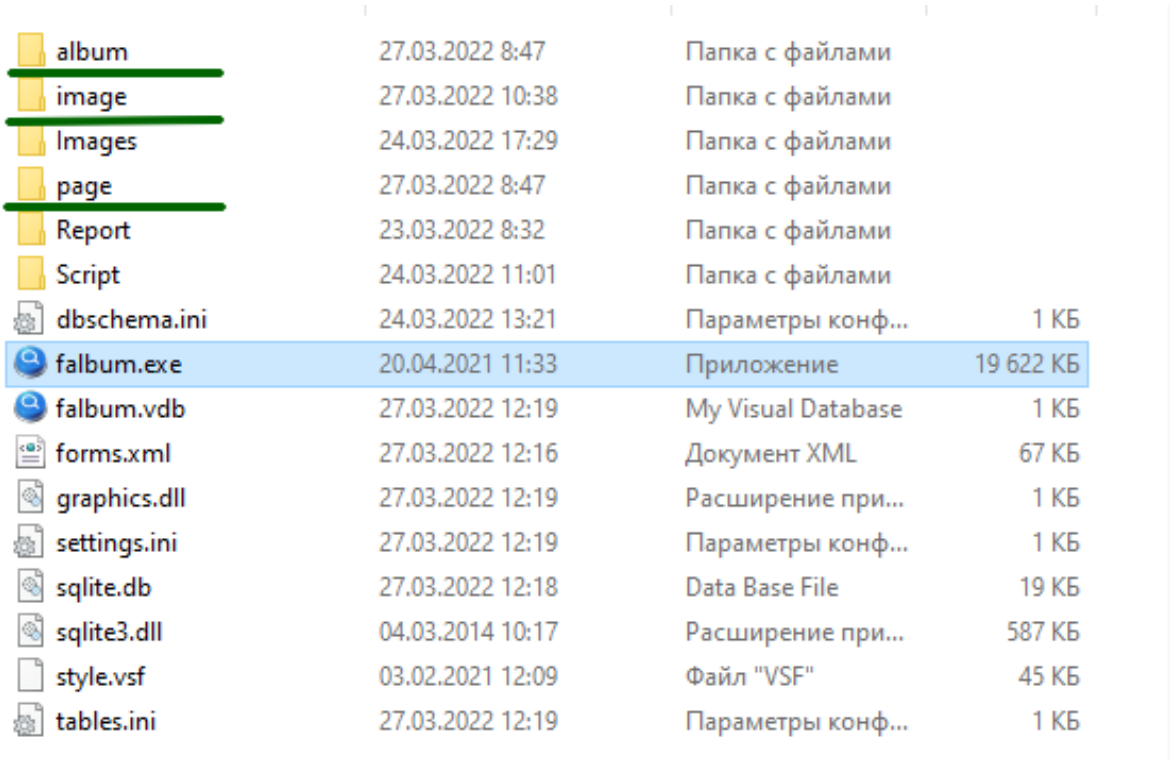
Вторым важным моментом является способ хранения изображений. Хотя базы данных SQLite и MySQL позволяют хранить картинки непосредственно в базе, но для хранения большого объёма данных этот способ не подходит, так как он будет заметно замедлять работу системы. Поэтому исходные картинки, а также миниатюры будут храниться в файловом хранилище - папке на диске рядом с базой данных.

Чтобы упростить систему поиска, название оригинального файла изображения будет состоять из ID записи и расширения, а у миниатюры к ID добавляется тип миниатюры.

Тип миниатюры - это возможность расширять функциональность системы хранения в дальнейшем, чтобы её можно было использовать в других проектах. Например, для отдела кадров может понадобиться дополнительный вид миниатюры в виде фотографии в личном деле с соотношением сторон 3x4, а для дерева структуры подразделений нужны миниатюры в виде кружков.

Расширение файлов миниатюр (формат хранения) будет фиксированным - JPG хорошо подходит для таких целей. А вот расширение оригинального файла будем

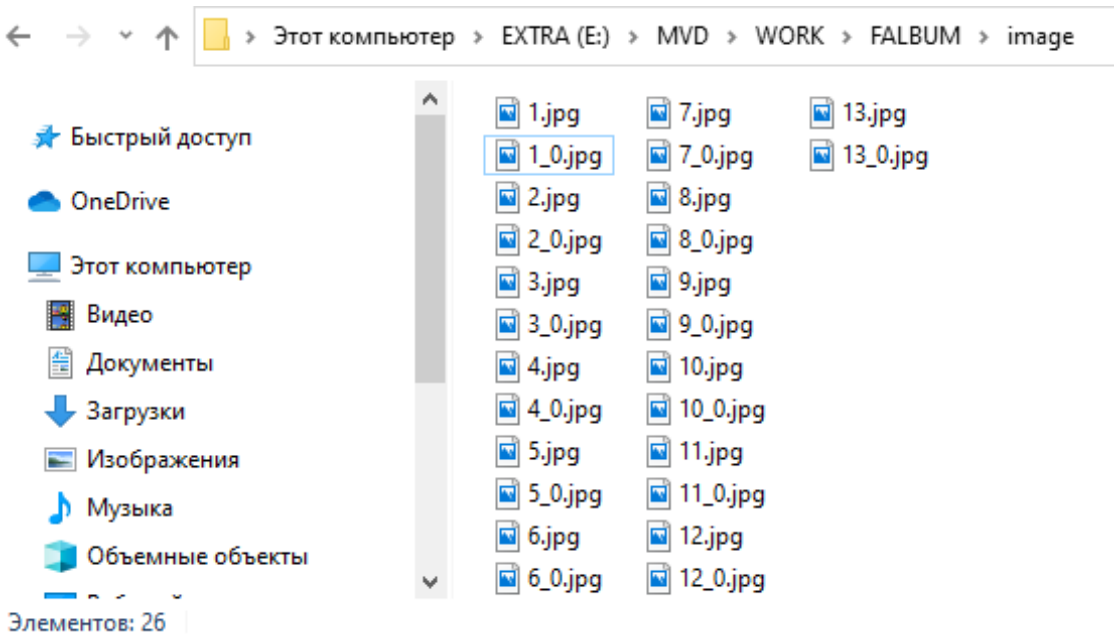
записывать в базу данных - оно понадобится для загрузки изображения в визуальные компоненты.



Семейный альбом для хранения изображений будет использовать три папки, которые называются также, как соответствующие таблицы базы данных:

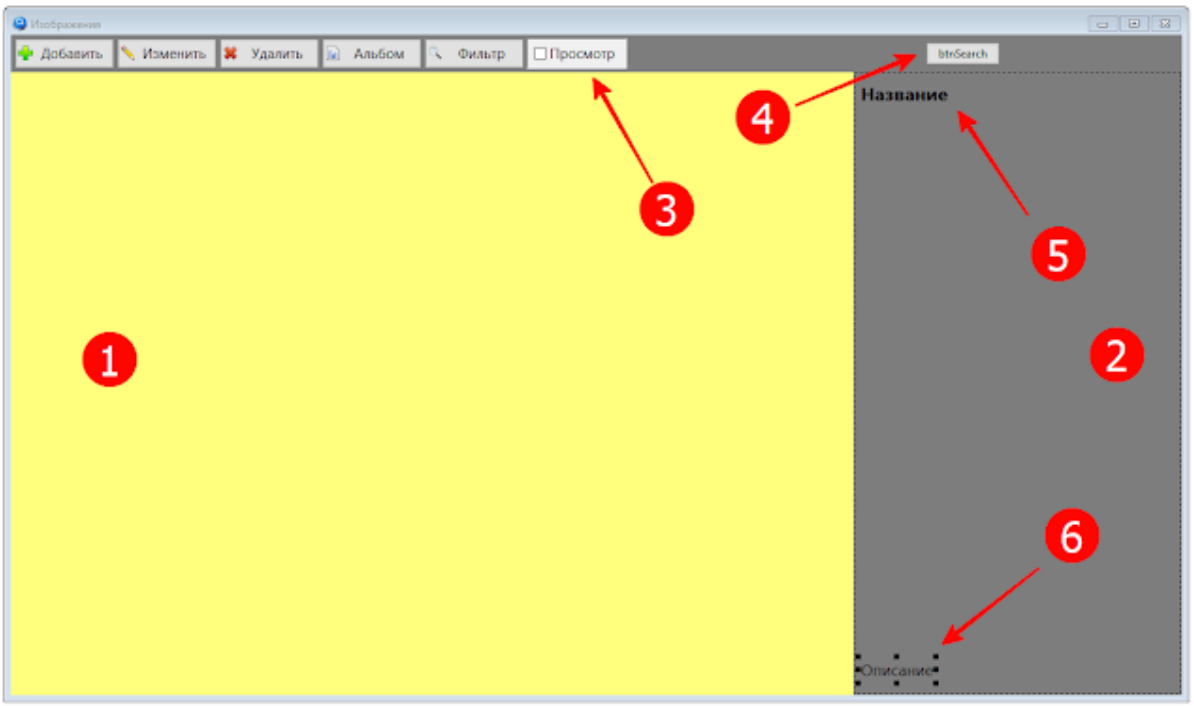
- **album** - миниатюры обложек альбомов
- **image** - оригинальные изображения и миниатюры фотографий и декоративных изображений
- **page** - миниатюры страниц альбомов

Тип миниатюры будет один - квадратная картинка для отображении в галерее размером 150x150 пикселей. Тип закодирован числом 0, которое отделяется от названия символом подчеркивания. Поэтому папка с файлами изображений выглядит так:



Форма отображения

На форме находится панель отображения миниатюр (1) и панель предварительного просмотра (2). Кроме обычных кнопок редактирования базой данных, на панели инструментов давлен чекбокс управления видимостью панели предварительного просмотра (3) и невидимая кнопка btnSearch, обработчик нажатия которой выполняет построение галереи изображений. На панели предварительного просмотра кроме самого изображения находятся пара меток для отображения названия (5) и описания (6).



Чекбокс был стилизован под кнопку с помощью панели. Такое решение обусловлено тем прискорбным фактом, что у стандартных кнопок (TdbButton) нет возможности задать свойство Down. Возможно в будущем панель инструмента мы реализуем с помощью компонента TToolbar, но пока, чтобы чрезмерно не усложнять программный код, ограничимся чекбоксом на панели.

Использование кнопки с названием btnSeach - это дань стандарту, который также может нам пригодиться в будущем.

Чтобы заготовка формы приобрела нужные кондиции (добавить сплиттер для управления размером панели предварительного просмотра, сделать адаптивные размер у метки, отвечающей за описание фотографии) необходимо её "допилить" с помощью скрипта.

```
procedure InitForm;
var
  tmpSplitter:TSplitter;
begin
  frmImage.panToolbar.Align := alTop;
  frmImage.panImagePreview.Align := alRight;
  frmImage.panPreview.Align := alClient;
  // добавить разделитель области предварительного просмотра
  tmpSplitter := TSplitter.Create(frmImage);
  tmpSplitter.Name := 'splMain';
  tmpSplitter.Parent := frmImage;
  tmpSplitter.Left := frmImage.panImagePreview.Left - 1;
  tmpSplitter.Align := alRight;
  tmpSplitter.Width := 7;
  // сделать адаптивные размеры надписи с описанием изображения
  frmImage.labName.align := alTop;
  frmImage.labDescription.align := alBottom;
  frmImage.imgPreview.align := alClient;
  // отработать скрытие панели
  frmImage_chbShowPreview_OnClick( frmImage.chbShowPreview );
end;
```

Построение галереи

Для построения данных используется простой запрос к базе. В дальнейшем этот запрос будет усложняться для фильтрации данных по заданным критериям.

```
procedure frmImage_btnSearch_OnClick (Sender: TObject; var Cancel: boolean);
var
  tmpSQL: string;
  tmpDataSet: TDataSet;
  tmpTop: integer;
  tmpLeft: integer;
  tmpImage:TdbImage;
```

```
  tmpScrollBox: TScrollBox;
begin
  Control_ClearChild( frmImage.panPreview );
  // для вертикальной прокрутки добавляем область прокрутки :)
  tmpScrollBox := TScrollBox.Create(frmImage);
  tmpScrollBox.Parent := frmImage.panPreview;
  tmpScrollBox.Align := alClient;
  tmpScrollBox.Name := 'scbPreview';
  //
  frmImage.panPreview.Tag := 0; // ID выделенного элемента
  tmpTop := 0;
  tmpLeft := 0;
  // построение превьюшек на панели
  tmpSQL := 'SELECT id, name, description FROM image';
  SQLQuery(tmpSQL,tmpDataSet);
  try
    while not tmpDataSet.EOF do
    begin
      // добавляем картинку
      tmpImage := TdbImage.Create(frmImage);
      tmpImage.Parent := tmpScrollBox;
      tmpImage.Top := tmpTop + 2;
      tmpImage.Left := tmpLeft + 2;
      tmpImage.Width := 150;
      tmpImage.Height := 150;
      tmpImage.Proportional := False;
      tmpImage.Stretch := False;
      // назначаем обработчики на кнопки
      AssignEvents(tmpImage);
      tmpImage.dbOnClick := 'PreviewImage_OnClick';
      tmpImage.dbOnDoubleClick := 'PreviewImage_OnDoubleClick';
      // используем некоторые свойства изображений для хранения данных
      tmpImage.Tag := tmpDataSet.FieldByName('id').asInteger;
      tmpImage.TagString := tmpDataSet.FieldByName('name').asString;
      tmpImage.Hint := tmpDataSet.FieldByName('description').asString;
      // загружаем изображение из файлового хранилища
      ImageEdit_LoadImageFromBase( tmpImage, tmpDataSet.FieldByName('id').asInteger,
PI_SMALL, TM_IMAGE);
      tmpDataSet.Next;
      // вычисляем следующую позицию для изображения
      tmpLeft := tmpLeft + 154; // шаг сетки
      // если изображение выходит за ширину панели + ширину скрола, то
      if frmImage.panPreview.Width < tmpLeft + 154 + 20 then
      begin // переходим на новую строку
        tmpLeft := 0;
        tmpTop := tmpTop + 154;
      end;
    end;
```

```
  except
    tmpDataSet.Free;
  end;
end;
```

В этой процедуре добавляются обработчики на клик и двойной клик по изображению. Обратите внимание, что для задания обработчика вместо свойства **onDoubleClick** (которое почему-то недоступно в скриптах) используется аналогичное свойство **dbonDoubleClick**.

> При назначении обработчиков через свойства вида db* необходимо использовать процедуру **AssignEvents().**

Обычный клик используется для выделения изображения (отображения рамки и загрузки данных об изображении на панель предварительного просмотра), а двойной клик - для редактирования изображения.

Для загрузки изображения с диска вызывается процедура ImageEdit_LoadImageFromBase(), которой в качестве параметра передаются компонент для загрузки, id записи и дополнительные параметры, о которых мы поговорим позже. Это связано с тем, что необходимо рассмотреть комплекс скриптов, обеспечивающих создание миниатюр.

Адаптивный размер

При изменении ширины панели производится перемещение миниатюр так, чтобы они занимали всю ширину области прокрутки. Если элементы не помещаются в области просмотра, то автоматически появляется вертикальная полоса прокрутки.

```
procedure frmImage_panPreview_OnResize (Sender: TObject);
// изменение размера
var
  i: integer;
  tmpImage:TdbImage;
  tmpFrame:TShape;
  C: TControl;
  tmpPanel: TdbPanel;
  tmpTop: integer;
  tmpLeft: integer;
  tmpForm:TAForm;
  tmpScrollBox: TScrollBox;
begin
  CForm(Sender,tmpForm);
  tmpPanel := TdbPanel(Sender);
  tmpTop := 0;
  tmpLeft := 0;
  FindC(tmpForm,'scbPreview',tmpScrollBox,False);
```

```
  // обработка нужна только для заполненой панели
  if tmpScrollBox <> nil then
  begin
    // двигаем картинки
    for i := tmpScrollBox.ControlCount - 1 downto 0 do
    begin
      C := tmpScrollBox.Controls[i];
      if C is TdbImage then
      begin
        c.Top := tmpTop + 2;
        c.Left := tmpLeft + 2;
        tmpLeft := tmpLeft + 154;
        if tmpPanel.Width < tmpLeft + 154 + 20 then
        begin
          tmpLeft := 0;
          tmpTop := tmpTop + 154;
        end;
      end;
    end;
    // двигаем рамку, если она есть.
    FindC(tmpForm,'shpSelector',tmpFrame,False);
    if tmpFrame <> nil then
    begin
      tmpImage := TdbImage(frmImage.panPreview.Tag);
      tmpFrame.Top := tmpImage.Top - 2;
      tmpFrame.Left := tmpImage.Left - 2;
    end;
  end;
end;
```

Редактор миниатюры

Миниатюрное изображение оригинала может быть использована в различных целях. Например, для отображения в галерее фотоальбома или как аватар сотрудника в программе кадрового учёта, для отображения в генеалогическом дереве и так далее.

Ключевые свойства компонента TdbImage

Компонент TdbImage унаследован от стандартного компонента TImage, который умеет "масштабировать" загруженное в него изображение, уменьшая или растягивая его до размера холста. При этом используются алгоритмы сглаживания, чтобы картинка выглядела естественно. Какие же свойства управляют этим процессом?

Stretch

Свойство **stretch** позволяет растягивать изображение, делая его больше или меньше оригинала.

- False - изображение отображается без масштабирования
- True - изображение растягивается на весь размер компонента; при этом оно может быть увеличено или уменьшено.

Обратите внимание, что изображение при масштабировании исказилось, так как при изменении размера нарушается пропорция: соотношение ширины и высоты изображения. В некоторых случаях это не критично (например, при растягивании монохромного изображения или градиента), но для фотографий и рисунков требуется сохранить пропорции. За это отвечает свойство Proportional, которое дополняет свойство Stretch

Proportional

False - пропорции не сохраняются, изображение растягивается по размеру компонента
True - пропорции сохраняются

Center

Это свойство позволяет центрировать изображение

- False - изображение отображается от левого верхнего угла компонента
- True - изображение отображается по центру компонента

Алгоритм

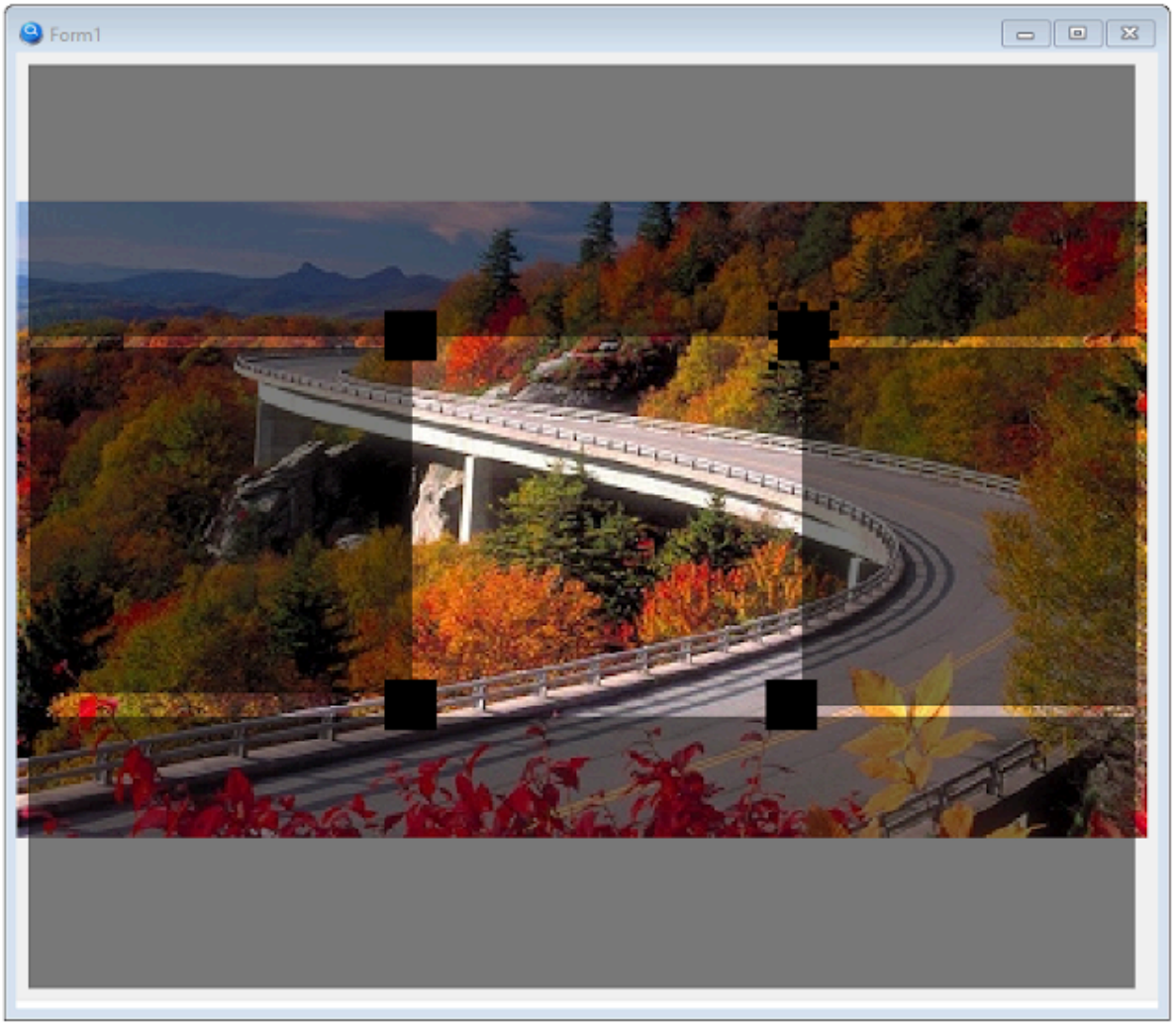
Для получения миниатюры нам потребуется два компонента TdbImage. Первый компонент будет содержать исходное изображение. Второй компонент предназначен для миниатюры. У обоих компонентов свойства будут установлены таким образом, чтобы масштабировать изображение с сохранением пропорций: Stretch = True, Proportional = True и Center = True.

Для получения копии изображения используем метод холста CopyRect(). Он копирует на холст изображение, находящееся на другом холсте. Чтобы фокус получился, компоненты TdbImage надо разместить на панелях, и в качестве источника копирования использовать холст родительской панели. Объяснение этому простое: для того, чтобы изображение, загруженное в компонент TdbImage стало видимым, оно отрисовывается на холсте родительского компонента, в нашем случае - панели. И отрисовывается оно с учетом свойств масштабирования, описанных выше. А холст TdbImage содержит исходное изображение.

Также необходимо каким-то образом определить область копирования, желательно, чтобы механизм был интуитивно понятным. Например, затемнить картинку, а область копирования сделать прозрачной, снабдив её "хелперами" - маленькими квадратиками

для перетаскивания мышкой, с помощью которых можно будет менять размер прозрачного "окна".

Для затенения придется использовать 4 изображения, в которые загружена полупрозрачная картинка формата png, а для хелперов - четыре непрозрачные метки черного цвета. Также понадобится ещё одно "изображение без изображения" - полностью прозрачная картинка, которую мы будем перетаскивать по экрану для выбора области копирования, а все остальные элементы (полупрозрачные шторки, хелперы) будут перемещаться вслед за ней, создавая иллюзию прозрачной "дырки".



Чтобы инструмент был универсальным, все его настройки должны быть описаны константами. Кроме того, он должен уметь создавать сразу несколько миниатюр разных размеров из одного заданного изображения.

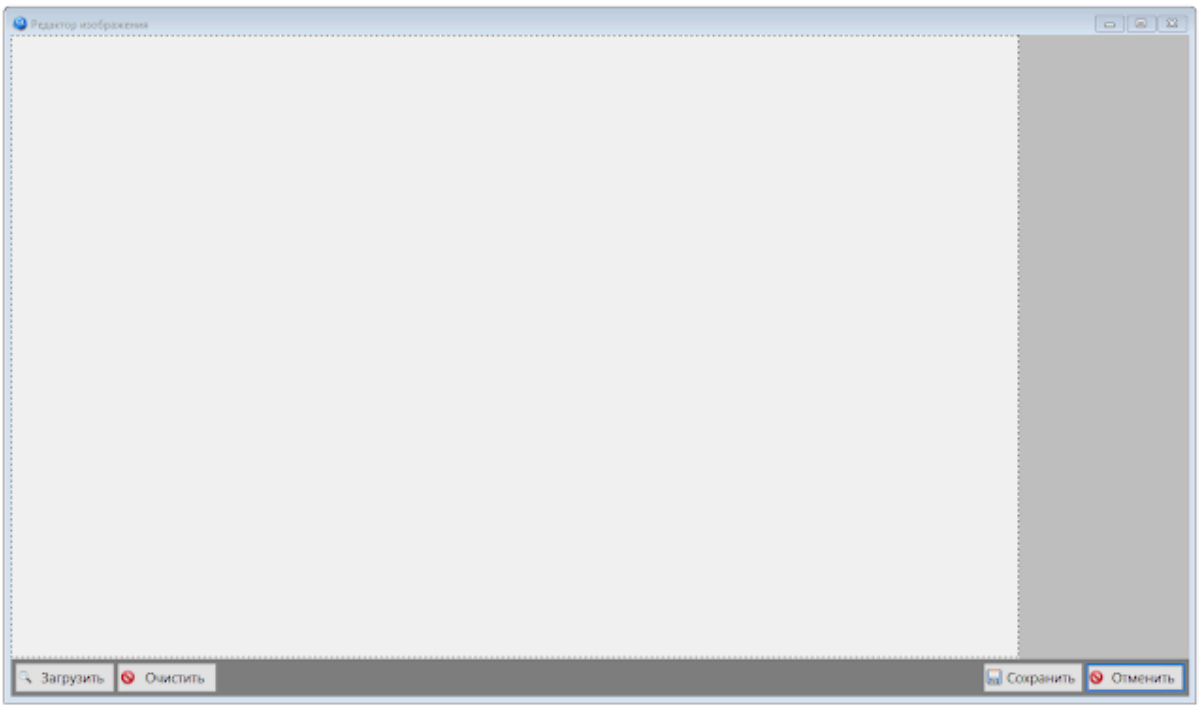
Виртуальные классы

Так как в My Visual Database нет возможности создавать свои классы, то придется создать "виртуальный класс" - библиотеку скриптов, обеспечивающую требуемый функционал для экземпляра компонента доступного класса.

В рамках данной книги нет смысла комментировать все методы виртуальных классов, поэтому я буду рассказывать только о ключевых.

Форма

Нам понадобится форма, на которой разместим ключевые элементы: панель исходного изображения, панель миниатюры и вспомогательные кнопки для управления процессом. Все остальные элементы будут создаваться программой, скрипты которой мы разместим в отдельном модуле. Это сделано для того, чтобы редактор миниатюр можно было легко интегрировать в любой проект, созданный на платформе My Visual Database.



TImageEdit

Собственно говоря, редактор изображения, в который будет осуществляться загрузка картинки. Там же будут реализованы методы сохранения изображения, методы манипуляции со шторками и хелперами для управления областью копирования.

Методы виртуального класса:

- ImageEdit_GetFileExt() - расширение редактируемого файла
- ImageEdit_GetImageDir() - получить путь к изображению по типу -
- ImageEdit_AjustHelpers() - установить хелперы по рамке
- ImageEdit_Create() - создание редактора картинок
- ImageEdit_CreatePreviews() - обновить миниатюры
- ImageEdit_Frame_OnMouseDown() - начало перемещения
- ImageEdit_Frame_OnMouseMove() - перемещение рамки селектора
- ImageEdit_Frame_OnMouseUp() - завершение перемещения
- ImageEdit_GetPhoto() - снятие изображения с канвы по заданным рамкой координатам
- ImageEdit_Helper_OnMouseDown() - начало перемещения хелпера
- ImageEdit_Helper_OnMouseMove() - перемещение хелперов
- ImageEdit_Helper_OnMouseUp() - окончание перемещения хелпера
- ImageEdit_Init() - установить режим редактирования
- ImageEdit_LoadImage() - загрузка изображения для редактирования
- ImageEdit_LoadImageFromBase() - загрузить изображение с сервера
- ImageEdit_PhotoPanel_OnResize() - изменение размера панели - корректировка рамки
- ImageEdit_Reset() - сбросить в исходное состояние
- ImageEdit_RestoreImages() - загрузка фотографий из файлового сервера
- ImageEdit_SaveImages() - сохранение фотографий на файловом сервере
- ImageEdit_SetFrame() - установить хелперы и настроить экран для редактирования фотографии
- ImageEdit_SetFrameDef() - сбросить в состояние по умолчанию
- ImageEdit_SetTargetImage() - установка целевой картинки

Изображения хранятся в папках, названия которых совпадают с названиями таблиц, а имя файла состоит из ID записи и расширения. Для оригинальных файлов сохраняется исходное расширения, для файлов миниатюр расширение jpg.

```
procedure ImageEdit_LoadImageFromBase(AImage: TdbImage; AID: integer;
APhotoType: integer; AImageType: integer);
// загрузить изображение с сервера
// если файл существует, то грузим его; если нет, то вставляем изображение по умолчанию
// AImage - куда загрузить
// AID - ID изображения
// APhotoType - типоразмер
// AImageType - тип картинки
var
```

```
  tmpFileName: string;
  tmpFolder: string;
  tmpFileExt: string;
begin
  tmpFolder := ImageEdit_GetImageDir(AImageType);
  // у оригинального файла неизвестно расширение,поэтому его нужно вытягивать
поиском
  if APhotoType = IE_FT_ORIGINAL then
  begin
    // тут надо делать проверку типа, но в нашем проекте оригинальные картинки
хранятся только в фотографиях
    tmpFileExt := SQLExecute('SELECT FileExt FROM image WHERE id ='+IntToStr(AID));
    tmpFileName := tmpFolder + IntToStr(AID) + tmpFileExt;
  end
  else // у остальных типоразмер входит в название, формат файла - jpg
    tmpFileName := tmpFolder + IntToStr(AID) + '_' + IntToStr(APhotoType) + '.jpg';
  // если файл существует, то грузим его
  if FileExists(tmpFileName) then
    AImage.Picture.LoadFromFile(tmpFileName)
  else // если нет, то вставляем изображение по умолчанию
    SetImage(AImage, IE_IMG_DEFAULT);
end;

procedure ImageEdit_SaveImages(AImageEdit: TdbImage;);
// сохраненние фотографий на файловом сервере
var
  tmpTIPanel: TdbPanel;
  tmpForm: TAForm;
  tmpName: string;
  tmpFileName: string;
  tmpID: Integer;
begin
  tmpID := AImageEdit.Tag;
  CForm(AImageEdit, tmpForm);
  tmpName := DeleteClassName(AImageEdit.Name);
  FindC(tmpForm, T_TARGET_IMAGE_PANEL + tmpName, tmpTIPanel);
  tmpFileName := tmpTIPanel.TagString + inttostr(tmpID) + AImageEdit.TagString;
  SaveImageToFile(AImageEdit, tmpFileName, True);
  // если картинка пустая, то файл не сохраняется, иначе сохраняем остальное
  if FileExists(tmpFileName) then
  begin
    TargetImagePanel_SaveImages(tmpTIPanel, tmpID);
  end;
end;
```

TTargetImage

Модуль для получения миниатюры. Принимает изображения от TImageEdit, работает с файловым хранилищем (сохраняет и загружает миниатюры в формате jpg).

Методы виртуального класса:

- TargetImage_Checker_OnClick() - клик по чекеру заголовка
- TargetImage_Create() - создание панели для отображения / сохранения изображения нужного размера
- TargetImage_Reset() - сбросить изображение в дефолтное
- TargetImage_RestoreImage() - загрузить изображение
- TargetImage_SaveImage() - сохранить изображение

Сохранение изображения происходит в два этапа. Сначала подготавливаем изображение: формируем имя файла и убираем рамку, если она была.

```
procedure TargetImage_SaveImage(APanel: TdbPanel; AID: integer);
// сохранить изображение
var
  tmpForm: TAForm;
  tmpName: string;
  tmpImage: TdbImage;
  tmpFrame: TdbImage;
  tmpTIPanel: TdbPanel;
  tmpFileName: string;
begin
  tmpTIPanel := TdbPanel(APanel.Parent);
  tmpName := DeleteClassName(DeleteSuffix(APanel.Name));
  CForm(APanel, tmpForm);
  FindC(tmpForm, T_IMAGE + tmpName, tmpImage);
  FindC(tmpForm, T_IMAGE + tmpName + SX_FRAME, tmpFrame, False);
  // собираем имя файла: папка + ID + расширение
  tmpFileName := tmpTIPanel.TagString + inttostr(AID) + APanel.TagString;
  // если есть рамка, то убирать
  if tmpFrame <> nil then
  begin
    tmpFrame.Visible := False;
    Application.ProcessMessages;
  end;
  SaveImageToFile(tmpImage, tmpFileName);
  // потом вернуть
  if tmpFrame <> nil then
    tmpFrame.Visible := True;
end;
```

Затем записываем картинку в файл

```
procedure SaveImageToFile(AImage: TdbImage; AFileName: string; AOriginalSize:
boolean = False);
// сохранение изображения с картинки в файл, в формате JPG
// картинка должна быть на панели того же размера, так как изображение
снимается с канвы панели.
// это позволяет сохранять
var
  tmpBuffer: TBitMap;
  tmpJPG: TJPEGImage;
  tmpPan: TdbPanel;
begin
  if AOriginalSize then // без масштабирования, в оригинальном разрешении
    AImage.Picture.SaveToFile(AFileName)
  else // с масштабированием, в размере видимого представления
  begin
    tmpBuffer := TBitMap.Create;
    tmpJPG := TJPEGImage.Create;
    tmpPan := TdbPanel(AImage.Parent); // удостовериться, что существует панель
    tmpBuffer.Width := AImage.Width;
    tmpBuffer.Height := AImage.Height;
    tmpBuffer.Canvas.CopyRect(0, 0, tmpBuffer.Width, tmpBuffer.Height, tmpPan.Canvas, 0,
0, tmpBuffer.Width,
      tmpBuffer.Height);
    tmpJPG.Assign(tmpBuffer);
    tmpJPG.SaveToFile(AFileName);
    tmpJPG.Free;
    tmpBuffer.Free;
  end;
end;
```

TTargetImagePanel

Панель для размещения модулей получения миниатюр. Адаптер - контейнер. Связующее звено между TImageEdit и набором TTatgetImage.

Методы класса

- TargetImagePanel_Clear() - очистить панель от виньеток
- TargetImagePanel_Reset() - сбросить все картинки на панели в исходное состояние
- TargetImagePanel_RestoreImages() - восстановление картинок из базы
- TargetImagePanel_SaveImages() - запись картинок в базу
- TargetImagePanel_SetMode() - установить режим работы - добавить нужные компоненты на панель целевых изображений

Все методы, за исключением последнего, работают со списком элементов TTargetImage, передавая ему соответствующую команду. А метод TargetImagePanel_SetMode() управляет самим списком: удаляет и создаёт необходимые элементы на панели.

```
procedure TargetImagePanel_SetMode(APanel: TdbPanel; AMode: integer);
// установить режим работы - добавить нужные компоненты на панель целевых изображений
var
  tmpImageEditor: TdbImage;
  tmpForm: TAForm;
  tmpName: string;
  tmpTIList: array of string;
  i: integer;
  tmpTIType: integer;
  tmpMaxWidth: integer;
  tmpParent: TWinControl;
  tmpPanel: TdbPanel;
begin
  tmpMaxWidth := 0;
  tmpName := DeleteClassName(DeleteSuffix(APanel.Name));
  CForm(APanel, tmpForm);
  FindC(tmpForm, T_IMAGE_EDIT + tmpName, tmpImageEditor);
  // если режим совпадает с текущим, состав виньеток не меняем, только сбрасываем изображения в исходные
  if APanel.Tag <> AMode then
  begin
    if AMode > (Length(ie_List) - 1) then
    begin
      RaiseException('TargetImagePanel_SetMode - режим не поддерживается: '+IntToStr(AMode) );
      Exit;
    end;
    // убираем все виньетки
    TargetImagePanel_Clear(APanel);
    // настроить папку
    APanel.TagString := ImageEdit_GetImageDir(AMode);
    // добавить виньетки
    // получить список форматов
    tmpTIList := SplitString( ie_List[ AMode ] ,',' );
    for i:=0 to Length(tmpTIList) - 1 do
    begin
      tmpTIType := StrToInt( tmpTIList[i] );
      // добавить целевую миниатюру на панель
      TargetImage_Create('TI'+IntToStr( tmpTIType ), ie_TICaption[tmpTIType], tmpImageEditor, APanel, ie_TIWidth[tmpTIType], ie_TIHeight[tmpTIType] );
      if ie_TIWidth[tmpTIType] > tmpMaxWidth then
      tmpMaxWidth := ie_TIWidth[tmpTIType];
    end;
```

```
    // запоминаем текущий режим здесь
    APanel.Tag := AMode;
  end;
  // установить ширину панели миниатюр
  // условность: редактор и панель миниатюр должны располагаться на одном родительском компоненте
  tmpParent := TWinControl( APanel.Parent );
  APanel.Width := tmpMaxWidth + IE_TI_MARGIN * 2;
  APanel.Left := tmpParent.Width - APanel.Width;
  tmpPanel := TdbPanel( tmpImageEditor.Parent );
  tmpPanel.Width := tmpParent.Width - APanel.Width;
  TargetImagePanel_Reset(APanel);
end;
```

Так как данные для инициализации хранятся в массивах, то процедуру TargetImagePanel_SetMode не придется переписывать при каждом изменении набора миниатюр или режимов редактирования.

Параметры инициализации

Все константы и переменные редактора вынесены в отдельный модуль IEConstVar.pas:

```
// Константы и переменные

const
  // Параметры редактора изображений
  IE_HELPER_SIZE = 10; // размер хелпера
  IE_HELPER_COLOR = clBlack; // цвет хэлпера
  IE_IMG_TRANSLUCENT = 'Black50'; // название компонента с полупрозрачным изображением, 50% черного
  IE_IMG_DEFAULT = 'DefImage'; // картинка по умолчанию
  // миниатюра
  IE_TI_CHECKBOX_SIZE = 24; // размер чекера выбора активной миниатюры
  IE_TI_MARGIN = 16; // отступ вертикальный
  IE_TI_CHECKER_NAME = 'PhotoEditMode_'; // название элемента с чекером
  //
  // для инициализации массивов
  IE_TI_COUNT = 2; // число форматов миниатюр
  IE_COUNT = 3; // число форматов редактирования
  //
  // для связи с основным приложением
  //
  // типы изображений (форматы редактирования)
  IE_IT_UNKNOW = -1; // начальное состояние
  IE_IT_IMAGE = 0; // картинки/фотографии
  IE_IT_ALBUM = 1; // обложки альбома
  IE_IT_PAGE = 2; // превью страниц альбома
```

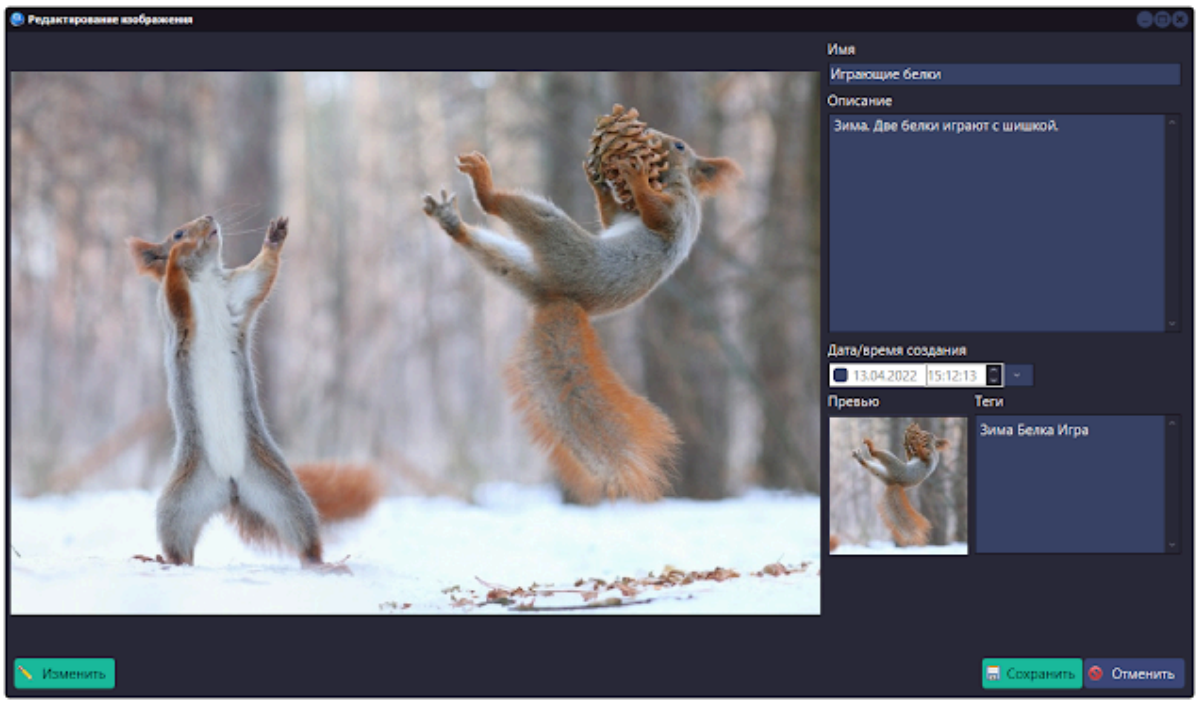
```
  // что загружаем - форматиы миниатюр
  IE_FT_ORIGINAL = -1; // оригинальное изображение
  IE_FT_SMALL = 0; // маленькая картинка
  IE_FT_MIDDLE = 1; // картинка побольше

var
  ie_MovedObject: TObject; // перемещаемый объект
  ie_pX: integer;
  ie_pY: integer;
  // форматы миниатюр
  ie_TIWidth: array of integer; // ширина миниатюры
  ie_TIHeight: array of integer; // высота миниатюры
  ie_TICaption: array of string; // название миниатюры
  // форматы редактирования
  ie_Tables: array of string; // таблица хранения
  ie_List: array of string; // список миниатюр, номера через запятую

begin
  // первичная инициализация
  //
  // форматы миниатюр
  SetLength(ie_TIWidth, IE_TI_COUNT);
  SetLength(ie_TIHeight, IE_TI_COUNT);
  SetLength(ie_TICaption, IE_TI_COUNT);
  //
  ie_TICaption[0] := 'Миниатюра для таблицы';
  ie_TIWidth[0] := 150;
  ie_TIHeight[0] := 150;
  //
  ie_TICaption[1] := 'Страница альбома';
  ie_TIWidth[1] := 250;
  ie_TIHeight[1] := 250;
  //
  // форматы редактирования
  SetLength(ie_Tables, IE_COUNT);
  SetLength(ie_List, IE_COUNT);
  //
  ie_Tables[0] := 'Image';
  ie_List[0] := '0';
  //
  ie_Tables[1] := 'Album';
  ie_List[1] := '1';
  //
  ie_Tables[2] := 'Page';
  ie_List[2] := '1';

end.
```

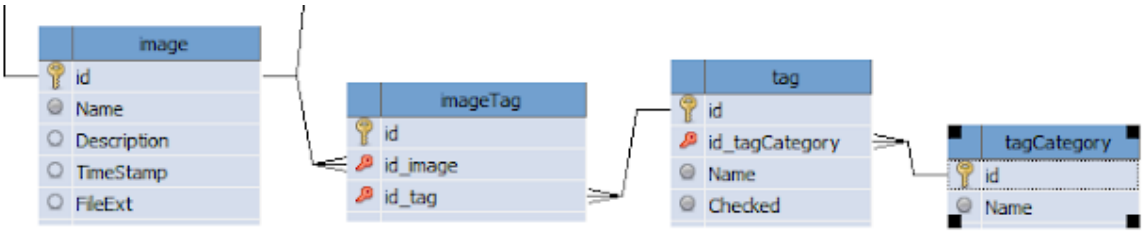
Поиск по тегам



Эффективность поиска зависит от применяемых алгоритмов. У каждого изображения в базе хранится название, описание, а также список тегов. Поиск по текстовому названию и описанию самый простой в реализации. Но он же самый медленный, так как для поиска обычно используется условие LIKE, с помощью которого находится фрагмент текста. Альтернативным вариантом поиска, который успешно применяется во многих социальных сетях, является поиск по тегам - ключевым словам, ассоциированным с искомыми данными.

Структура данных

Для организации поиска по тегам была создана необходимая структура таблиц:



Изображения (Image) связываются с тегами (tag) связью "многие-ко-многим" (imageTag). Для организации фильтра по тегам используется поле tag.Checked, что хорошо работает для однопользовательского приложения. Дополнительный справочник категорий тегов позволит навести порядок, если тегов станет очень много.

Редактирование тегов изображения

По канонам My Visual Database для редактирования таблицы **imageTag** нам пришлось бы добавлять на форму редактирования изображения дочернюю таблицу, в которой бы отображались теги. Но то не совсем удобно. Во-первых я не фанат дочерних таблиц, во-вторых возникнут сложности в организации процесса редактирования такой таблицы, так как редактировать данные в самой таблице при большом числе тегов будет неудобно. А в-третьих при добавлении изображения часто возникает необходимость добавления новых тегов. Поэтому для редактирования тегов изображения будем применять текстовый редактор и специальный скрипт.

Для текстового редактора подходят два компонента: **TdbMemo** и **TdbRichEdit**. Однако я остановил свой выбор на последнем, так как именно он позволяет выделять разными цветами отдельные слова набираемого текста.

По задумке пользователь набирает теги в редакторе, а после того, как он заканчивает редактирование, а этот момент определяется паузой в наборе текста, программа автоматически их проверяет. И если такого тега ещё нет в базе, то он выделяется другим цветом, чтобы пользователь сразу понимал: либо это новый тег, либо он ошибся в написании. А при сохранении данных для каждого нового тега открывается форма добавления для выбора категории тега.

Первая сложность возникла с поведением **TdbRichEdit** при использовании стиля, а именно - цвет фона документа не устанавливается в соответствии с выбранным стилем. Более того, у данного компонента нет свойства для управления цветом фона редактора. Выход был найден: при запуске приложения в компонент загружается HTML документ, в котором у тела прописывается свойство цвета фона (а вот с компонентом **TdbDateTimePicker** дела обстоят хуже - он светится ярким белым пятном на тёмной теме "TableDark" и скорей всего придётся его заменить на TdbEdit и пару скриптов...).

Вторая сложность, точнее нюанс поведения, состоял в том, что событие OnChange возникало при любом изменении в редакторе (для сравнения - в **TdbMemo** OnChange возникает только при ручном изменении текста пользователем), поэтому пришлось задействовать обработчик события OnKeyUp, в котором запускается 2-секундный таймер.

```
procedure frmImageEdit_redTags_OnKeyUp (Sender: TObject; var Key: Word; Shift, Alt, Ctrl: boolean);
// отпускание клавиши в редакторе тегов
begin
  // если таймера ещё нет, то создать его
  if TagTimer = nil then
  begin
    TagTimer := TTimer.Create(frmImageEdit);
    TagTimer.OnTimer := @TagTimer_OnTimer;
  end;
  // перезапустить таймер на 2 секунды
```

```
  TagTimer.Enabled := False;
  TagTimer.Interval := 2000;
  TagTimer.Enabled := True;
  // установить признак того, что данные изменил пользователь
  frmImageEdit.redTags.Tag := 1;
end;
```

А после окончания редактирования тегов запускается обработчик, который выполняет проверку тегов и их подсвечивание при необходимости.

```
procedure TagTimer_OnTimer (Sender: TObject);
// обработка списка тегов в редакторе тегов
var
  i: integer;
  Tags: array of string;
  s: string;
  tmpTag: string;
  tmpETag: string;
begin
  TagTimer.Enabled := False; // оставновить таймер
  s := frmImageEdit.redTags.Text; // взять данные о тегах
  frmImageEdit.redTags.Clear; // очистить редактор
  // запятые заменить пробелами, убрать перевод строки и двойные пробелы
  s := ReplaceStr(s,',',' ');
  s := ReplaceStr(s,chr(13),' ');
  s := ReplaceStr(s,'  ',' ');
  Tags := SplitString( s,' '); // получить список тегов в виде строкового массива
  for i := 0 to length(Tags) - 1 do
  begin
    tmpTag := Trim(Tags[i]); // на всякий случай, чтобы остался только текст
    if (tmpTag <> '') then
    begin
      // ищем тег в базе
      tmpETag := VarToStr( SQLExecute('SELECT name FROM tag WHERE upper( name ) =
upper("'+tmpTag+'")') );
      // если не найден, то
      if tmpETag = '' then
      begin // вставляем тег в написании пользователя, но с заглавной первой буквой,
жёлтым цветом
        tmpTag := UpperCase(copy(tmpTag,1,1)) + copy(tmpTag,2,length(tmpTag)-1);
        frmImageEdit.redTags.InsertTextEx(tmpTag,$00FFFF,11,0,'Segoe UI')
      end // если найден, то вставляем в написании из базы, белым цветом
      else
        frmImageEdit.redTags.InsertTextEx(tmpETag,$FFFFFF,11,0,'Segoe UI');
     // добавляем пробел
     frmImageEdit.redTags.InsertTextEx(' ',$FFFFFF,11,0,'Segoe UI');
    end;
  end;
```

```
end;
```

Также понадобится пара скриптов: для сохранения тегов в базу

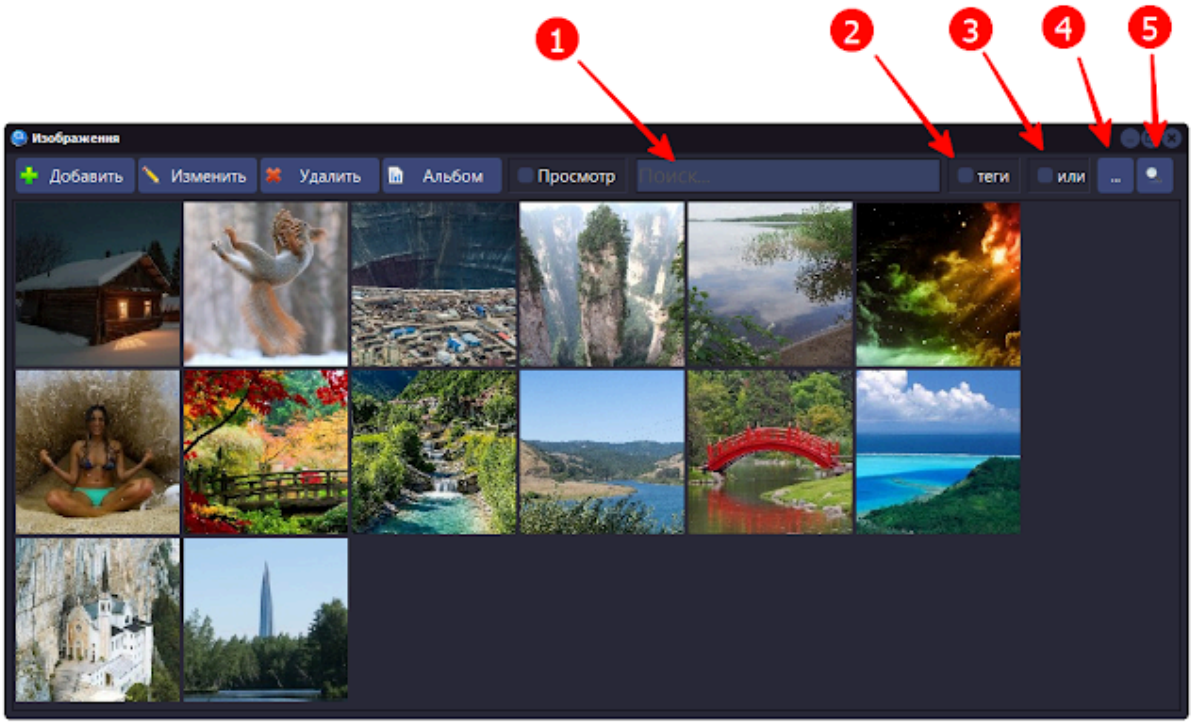
```
procedure frmImageEdit_btnSave_OnAfterClick (Sender: TObject);
// после успешного сохранения
var
  tmpSQL: string;
  s: string;
  tmpIDTag: string;
  tmpIDImage: string;
  Tags: array of string;
  i: integer;
begin
  // работа с тегами
  if frmImageEdit.redTags.Tag = 1 then
  begin
    // ID изображения
    tmpIDImage := IntToStr(frmImageEdit.btnSave.dbGeneralTableId);
    // удалить старые
    tmpSQL := 'DELETE FROM imageTag WHERE id_image = '+tmpIDImage;
    SQLExecute(tmpSQL);
    // сохранить новые
    s := frmImageEdit.redTags.Text;
    s := ReplaceStr(s,',',' ');
    s := ReplaceStr(s,chr(13),' ');
    s := ReplaceStr(s,'  ',' ');
    Tags := SplitString(s,' ');
    for i := 0 to length(Tags) - 1 do
    begin
      s := Trim(Tags[i]);
      if (s <> '') then
      begin
        // используем регистронезависимый поиск:
        tmpSQL := 'SELECT id FROM tag WHERE upper( name ) = upper('''+s+''')';
        // ID тега
        tmpIDTag := VarToStr( SQLExecute(tmpSQL) );
        // если такого тега нет в базе, то предлагаем пользователю добавить его
        if tmpIDTag = '' then
        begin
          // TODO: сделать опцию автоматического добавления, без участия
пользователя
          // добавляем новый тег
          frmTagEdit.TagString := s;
          frmTagEdit.NewRecord('tag');
          tmpIDTag := VarToStr( SQLExecute('SELECT id FROM tag WHERE upper( name ) =
upper('''+s+''')') );
```

```
      end;
      if tmpIDTag <> '' then
      begin
        tmpSQL := 'INSERT INTO imageTag (id_image,id_tag) VALUES ( '+tmpIDImage+',
'+tmpIDTag+') ';
        SQLExecute(tmpSQL);
      end;
    end;
  end;
 end;
end;
```

...и для восстановления внешнего вида тегов из базы при открытии формы редактирования: ведь в базе для каждого изображения хранятся не тексты тегов, а ссылки на них, что позволяет делать выборки с большими массивами данных достаточно быстро, а также использовать различные логические операции и релевантный поиск.

```
procedure frmImageEdit_OnShow (Sender: TObject; Action: string);
// отображение формы
var
  s: string;
  tmpSQL: string;
begin
  // загружаем картинкииз базы
  ImageEdit_LoadImageFromBase( frmImageEdit.imgImage,
frmImageEdit.btnSave.dbGeneralTableId, IE_FT_ORIGINAL, IE_IT_IMAGE);
  ImageEdit_LoadImageFromBase( frmImageEdit.imgPreview,
frmImageEdit.btnSave.dbGeneralTableId, IE_FT_SMALL, IE_IT_IMAGE);
  if Action = 'NewRecord' then
  begin
    frmImageEdit.edtName.Text := 'Новое изображение';
    frmImageEdit.redTags.InsertTextEx(' ',$FFFFFF,11,0,'Segoe UI');
  end
  else
  begin
    // сформировать список тегов
    tmpSQL := ' SELECT GROUP_CONCAT (tag.name," ") FROM imageTag LEFT JOIN tag
ON tag.id = imageTag.id_tag WHERE imageTag.id_image =
'+IntToStr(frmImageEdit.btnSave.dbGeneralTableId);
    s := VarToStr( SQLExecute(tmpSQL) );
    frmImageEdit.redTags.InsertTextEx(s,$FFFFFF,11,0,'Segoe UI');
  end;
  frmImageEdit.redTags.Tag := 0;
end;
```

Система поиска



Панель поиска разместилась на панели инструментов. Она включает в себя поле для ввода текста (1), чекер включения режима поиска по тегам (2), чекер включения логического "ИЛИ" при поиске по тегам (3), кнопку выбора тегов из справочника (4) и, собственно, кнопку поиска (5).

```
procedure frmImage_btnSearch_OnClick (Sender: TObject; var Cancel: boolean);
// построение галереи
var
  tmpSQL: string;
  tmpDataSet: TDataSet;
  tmpTop: integer;
  tmpLeft: integer;
  tmpImage:TdbImage;
  tmpScrollBox: TScrollBox;
begin
  Control_ClearChild( frmImage.panPreview );
  // для вертикальной прокрутки добавляем область прокрутки :)
  tmpScrollBox := TScrollBox.Create(frmImage);
  tmpScrollBox.Parent := frmImage.panPreview;
  tmpScrollBox.Align := alClient;
  tmpScrollBox.Name := 'scbPreview';
  //
  frmImage.panPreview.Tag := 0; // ID выделенного элемента
  tmpTop := 0;
  tmpLeft := 0;
```

```
  // построение превьюшек на панели
  if frmImage.edtFilter.Text = '' then
    // все подряд
    tmpSQL := 'SELECT id, name, description FROM image LIMIT 30'
  else
  if not frmImage.chbTag.Checked then
    // по имени или по описанию по вхождению
    tmpSQL := 'SELECT id, name, description FROM image WHERE (name LIKE
"%'+frmImage.edtFilter.Text+'%") OR (description LIKE "%'+frmImage.edtFilter.Text+'%")
LIMIT 30'
  else
  if frmImage.chbOR.Checked then
  // "ИЛИ", упорядочены по релевантности
    tmpSQL := ' SELECT id, name, description FROM ( SELECT count(image.id) as c,
image.id, image.name, image.description FROM tag LEFT JOIN imageTag ON
imageTag.id_tag = tag.id LEFT JOIN image ON image.id = imageTag.id_image WHERE
tag.checked = 1 AND image.id IS NOT NULL GROUP BY image.id ORDER BY 1 DESC
LIMIT 30 )  '
  else
    // строгое совпадение "И"
    tmpSQL := 'SELECT id, name, description FROM ( SELECT count(image.id) as c,
image.id, image.name, image.description FROM tag LEFT JOIN imageTag ON
imageTag.id_tag = tag.id LEFT JOIN image ON image.id = imageTag.id_image WHERE
tag.checked = 1 AND image.id IS NOT NULL GROUP BY image.id ) WHERE c = (SELECT
count(*) FROM tag WHERE tag.checked = 1) LIMIT 30';
  SQLQuery(tmpSQL,tmpDataSet);
  try
    while not tmpDataSet.EOF do
    begin
      // добавляем картинку
      tmpImage := TdbImage.Create(frmImage);
      tmpImage.Parent := tmpScrollBox;
      tmpImage.Top := tmpTop + 2;
      tmpImage.Left := tmpLeft + 2;
      tmpImage.Width := 150;
      tmpImage.Height := 150;
      tmpImage.Proportional := False;
      tmpImage.Stretch := False;
      // назначаем обработчики на кнопки
      AssignEvents(tmpImage);
      tmpImage.dbOnClick := 'frmImage_PreviewImage_OnClick';
      tmpImage.dbOnDoubleClick := 'frmImage_PreviewImage_OnDoubleClick';
      // используем некоторые свойства изображений для хранения данных
      tmpImage.Tag := tmpDataSet.FieldByName('id').asInteger;
      tmpImage.TagString := tmpDataSet.FieldByName('name').asString;
      tmpImage.Hint := tmpDataSet.FieldByName('description').asString;
      // загружаем изображение из файлового хранилища
```

```
      ImageEdit_LoadImageFromBase( tmpImage, tmpDataSet.FieldByName('id').asInteger,
IE_FT_SMALL, IE_IT_IMAGE);
      tmpDataSet.Next;
      // вычисляем следующую позицию для изображения
      tmpLeft := tmpLeft + 154; // шаг сетки
      // если изображение выходит за ширину панели + ширину скрола, то
      if frmImage.panPreview.Width < tmpLeft + 154 + 20 then
      begin // переходим на новую строку
        tmpLeft := 0;
        tmpTop := tmpTop + 154;
      end;
    end;
  finally
    tmpDataSet.Free;
  end;
end;
```

Если набрать текст в строке поиска и нажать кнопку "Поиск", то выборка данных будет осуществляться через SQL запрос с поиском текста по вхождению в полях **image.name** и **image.description**.

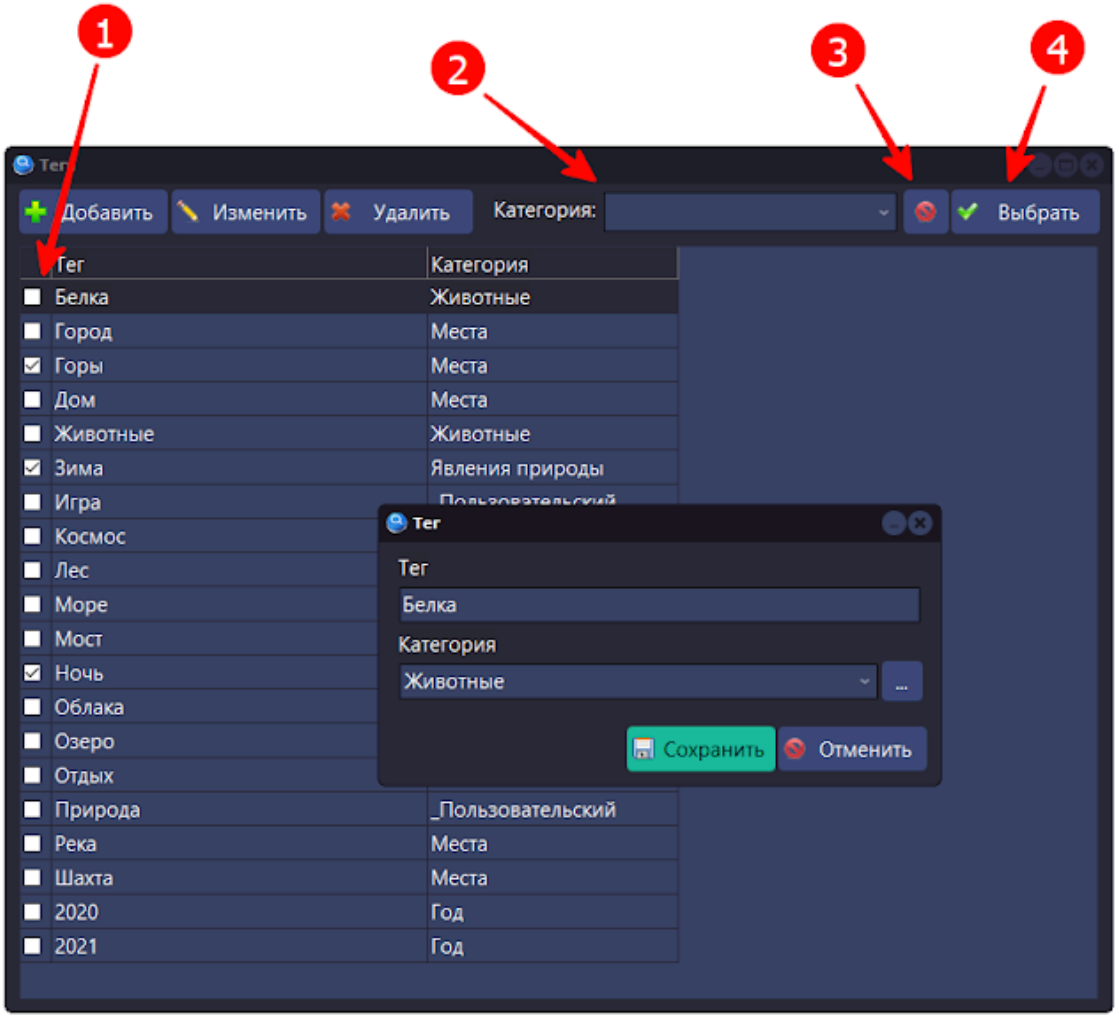
При активации чекера "Теги" набранный в строке поиска текст интерпретируется как набор тегов, разделенных пробелом или запятой. Результат включает в себя картинки у которых присутствуют все указанные теги.

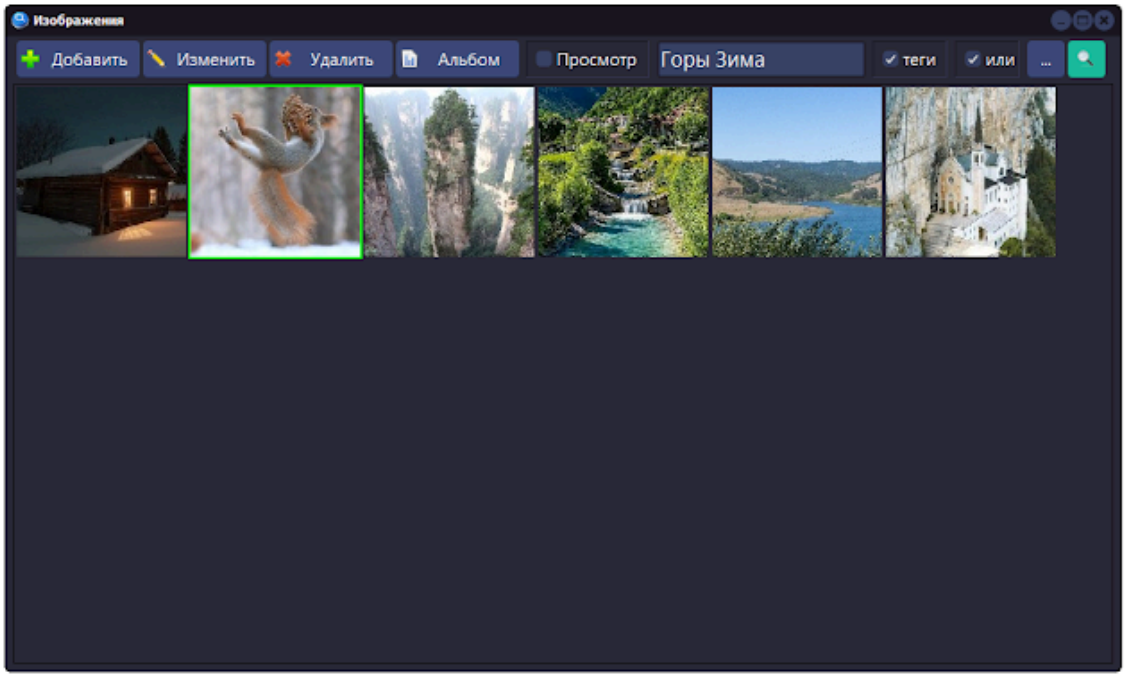
При активации чекера "Или" результат будет включать изображения, в которых есть хотя бы один искомый тег. Сами изображения будут упорядочены в порядке релевантности (сначала с большим числом совпадений).

При нажатии кнопки "Выбрать теги из справочника" открывается справочник тегов, который одновременно является настраиваемым фильтром.

Для выбора тегов используется колонка с чекерами (1), при этом теги можно фильтровать по категориям (2). Для сброса всех чекеров предусмотрена кнопка "Очистить выбор" (3). А для подтверждения выбора кнопка "Выбрать" (4), при нажатии которой выбранные теги переносятся в фильтр поиска.

Запрос выборки данных содержит фиксированное ограничение на количество записей, чтобы построение галереи занимало конечное и относительно маленькое время. В дальнейшем необходимо будет добавить специальную кнопку "Показать ещё" для загрузки очередной партии картинок. Запрос будет аналогичным, но включать команду OFFSET - смещение от начала выборки.

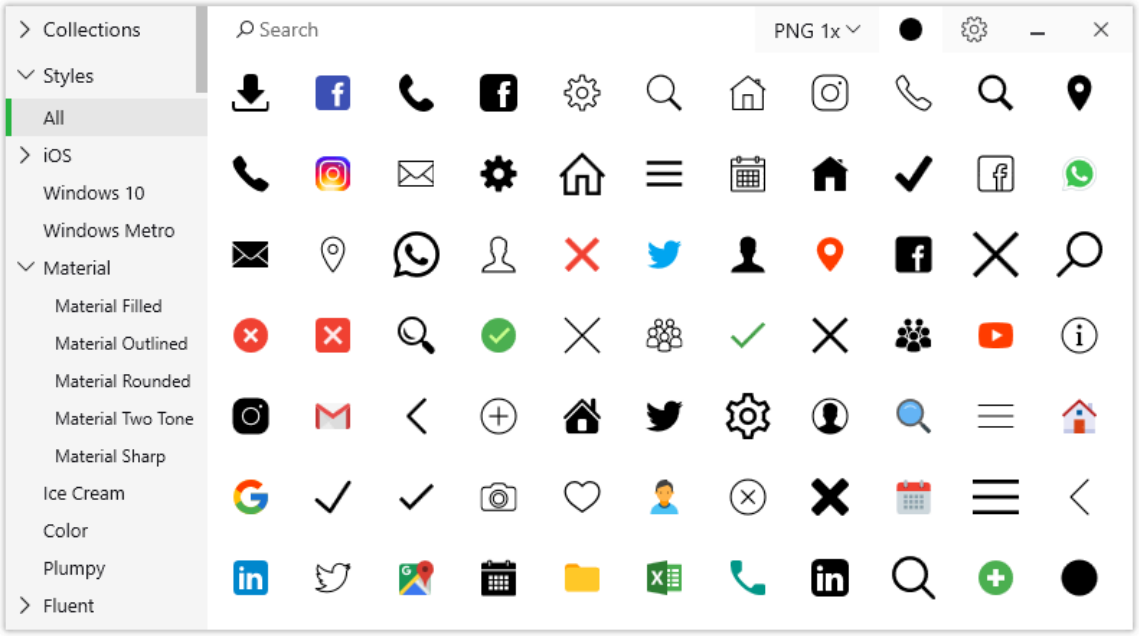

1
2
3
4
Добавить
Изменить
Удалить
Категория:
Выбрать
Тег
Категория
Белка
Животные
Город
Места
Горы
Места
Дом
Места
Животные
Животные
Зима
Явления природы
Игра
Космос
Лес
Море
Мост
Ночь
Облака
Озеро
Отдых
Природа
_Пользовательский
Река
Места
Шахта
Места
2020
Год
2021
Год
Тег
Белка
Категория
Животные
Сохранить
Отменить


Изображения
Добавить
Изменить
Удалить
Альбом
Просмотр
Горы Зима
теги
или

Картинки на кнопках

После того, как часть основного функционала реализована, можно передохнуть и подумать об улучшении внешнего вида программы. Например, заменить стандартные изображения на кнопках на более современные и соответствующие выбранному цветовому оформлению.

Задача решается в два этапа: создание (поиск) изображений и загрузка их в программу. Первый этап я рекомендую пройти с помощью удобного и бесплатного инструмента: библиотеки изображений Pichon for Windows, которую можно найти на сайте https://icons8.com



Так как кнопки в выбранной теме отображаются двумя контрастными цветами, то нам понадобится два набора изображений: более светлые для тёмного фона и более тёмные для светлого. Собственно говоря, выбор цвета изображений определяется стилем: я буду использовать цвет текста, которым отображаются надписи на кнопках.

Для стиля **TabletDark** это белый и чёрный. Для этих целей хорошо подходят изображения, у которых можно задавать цвет (Material), в частности Material Two Tone - эти изображения кроме основного тона содержат полутона - полупрозрачные области изображений.

Сами изображения можно хранить в специальном файле graphics.dll, который создается автоматически и содержит все графические данные, используемые в режиме конструктора форм. Этот способ хорошо описан в книге “Современный UI”. Но в этот раз изображения будут загружаться в момент запуска приложения из специальной папки, что больше соответствует принципам стилизации программы. Замечу, что стиль конечного приложения зависит от наличия и содержания файла style.vsf, то есть, меняя этот файл можно менять стиль приложения без его повторной компиляции. То же самое можно будет проделывать и с картинками кнопок, просто заменяя файлы в папке с картинками.

Виртуальный класс Images

Для реализации всех функций работы с изображениями создадим виртуальный класс и набор скриптов. Назначение класса - работа с наборами изображений, которые будут загружаться в программу в момент запуска приложения.

Класс реализован через абстрактные методы (без создания экземпляра класса), их всего четыре:

Метод	Назначение
Images_ButtonsAssign()	Назначение картинок кнопкам на формах
Images_Init()	Инициализация данных
Images_Load()	Загрузка картинок
Images_Set	Назначить кнопке индексы картинок по названию файла

Хранение данных осуществляется в глобальных переменных, в массивах различных типов. Если бы MVDB поддерживал полноценный синтаксис языка Pascal, то вместо этого зоопарка мы бы использовали структурированные данные (Record), но, как говорится, что есть, то есть.

```
const
  // папки указаны относительно папки с приложением
  IMAGES_DIR = '\Images\';   // папка, в которой находятся изображения

  IMAGES_SELECTED_SUFFIX = '_S'; // суффикс в названии для изображения выбранной кнопки
  IMAGES_HOT_SUFFIX = '_H'; // суффикс в названии для изображения кнопки при наведении курсора
  IMAGES_DISABLE_SUFFIX = '_D'; // суффикс в названии для изображения задизейбленной кнопки
  IMAGES_PRESSED_SUFFIX = '_P'; // суффикс в названии для изображения нажатой кнопки

  IMAGES_FORMAT_COUNT = 2; // сколько форматов в библиотеке
  // перечень форматов
  IMAGES_FORMAT_BUTTON = 0; //
  IMAGES_FORMAT_BIGBUTTON = 1; //
  //
  // модели загрузки изображений кнопок
  IMAGES_SIMPLE_MODEL = 1; // упрощенная
  IMAGES_FULL_MODEL = 2; // полная


var
  ImagesDir: array of string; // папки хранения
  ImagesSize: array of integer; // размеры изображений
  ImagesList: array of TImageList; // хранилища картинок
  ImagesNames: array of TStringList; // имена загруженных картинок
```

Инициализация данных выполняется при запуске приложения. Она включает настройку папок для загрузки изображений и их размер. Особенность использования класса TImageList для хранения изображений в том, что в нем можно хранить изображения одного размера. Поэтому, для каждого размера потребуется отдельное хранилище.

```
procedure Images_Init;
// инициализация данных
var
  i: integer;
begin
  // инициализация массивов данных
  SetLength(ImagesDir,IMAGES_FORMAT_COUNT);
```

```
  SetLength(ImagesSize,IMAGES_FORMAT_COUNT);
  SetLength(ImagesList,IMAGES_FORMAT_COUNT);
  SetLength(ImagesNames,IMAGES_FORMAT_COUNT);
  // настройка, выполняемые для проекта
  ImagesDir[IMAGES_FORMAT_BUTTON] := IMAGES_DIR + 'Buttons\';
  ImagesSize[IMAGES_FORMAT_BUTTON] := 32;
  ImagesDir[IMAGES_FORMAT_BIGBUTTON] := IMAGES_DIR + 'BigButtons\';
  ImagesSize[IMAGES_FORMAT_BIGBUTTON] := 48;
  // загрузка данных
  for i := 0 to IMAGES_FORMAT_COUNT-1 do
  begin
    ImagesNames[i] := TStringList.Create;
    ImagesList[i] := TImageList.Create(MainForm);
    Images_Load(i); // загрузить изображения
  end;
  Images_ButtonsAssign( IMAGES_FORMAT_BUTTON,'','frmShow' );
  Images_ButtonsAssign( IMAGES_FORMAT_BIGBUTTON,'frmShow','' );
end;
```

Инициализация включает в себя не только настройку массивов под конкретный проект, но и загрузку данных с диска, а также автоматическое назначение картинок кнопкам.

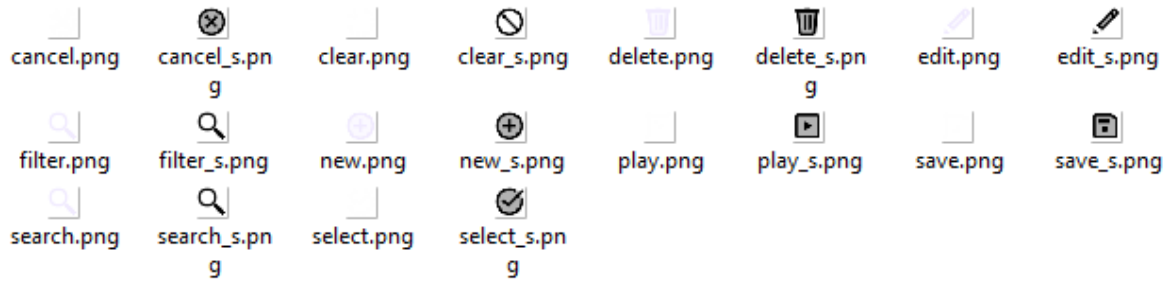
```
procedure Images_Load( AIndex:integer  );
// загрузка картинок
// сделана процедурой с параметрами, так как в приложении может быть несколько
размеров изображений,
// которые должны храниться в разных списках; картинки в формате png
var
  tmpList:TStringList;  // список файлов в папке с картинками
  tmpImageList:TImageList;
  tmpImageNames: TStringList;
  i:integer;
  tmpImageDir: string;
  s:string;
begin
  tmpImageDir := ExtractFilePath(Application.ExeName)+ImagesDir[AIndex];
  if DirectoryExists(tmpImageDir) then
  begin
    tmpImageList := ImagesList[AIndex];
    tmpImageNames := ImagesNames[AIndex];
    //
    tmpImageList.Masked:=false;
    tmpImageList.ColorDepth:=cd32bit;
    // размер картинок
    tmpImageList.Width := ImagesSize[AIndex];
    tmpImageList.Height := ImagesSize[AIndex];
    tmpList := TStringList.Create;
    try
```

```
    tmpList.Text := GetFilesList( tmpImageDir );
    for i := 0 to tmpList.Count - 1 do
    begin
      tmpImageList.AddPng( tmpList.strings[i] );
      s := UpperCase( ExtractFileName(tmpList.strings[i]) );
      s := copy(s,1,Length(s)-4);
      tmpImageNames.Add( s );
    end;
  finally
    tmpList.Free;
  end;
 end
 else
   RaiseException('Не найдена папка '+tmpImageDir);
end;
```

Картинки должны быть в формате PNG. Их названия должны быть подчинены определенным правилам: для обозначения изображений различных состояний кнопок используются суффиксы в названии файлов.

Суффикс	Изображение	Примечание
_S(elected)	Выбранная кнопка	Выбранная кнопка - это кнопка с фокусом, фокус попадает на кнопку после нажатия или с помощью кнопки **Tab**
_H(ot)	Кнопка с наведенным на неё курсором	В момент наведения, если кнопка доступна.
_D(isabled)	Недоступная кнопка	
_P(ressed)	Нажатая кнопка	В момент нажатия, после отпускания кнопка становится выбранной

В полной модели используются пять изображений для одной кнопки. В упрощенной модели - только две.



Назначение изображений производится по определенным правилам: название кнопки должно включать в себя трёхбуквенный префикс (имя класса) и имя кнопки. Если имеется картинка с таким именем, то происходит привязка массива изображения к данной кнопке и назначение соответствующих индексов. Присвоение идет по всем формам, но можно задать белый и черный список форм. Таким образом достигается максимальная универсальность кода.

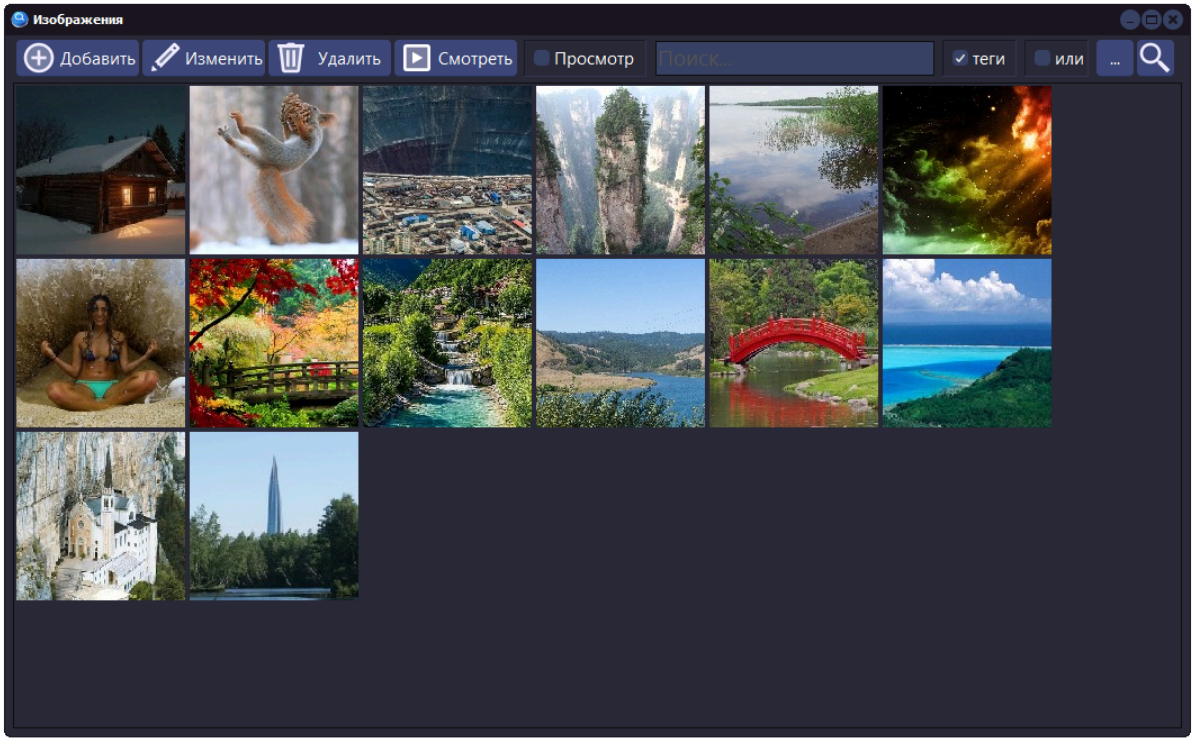
```
procedure Images_ButtonsAssign( AIndex: integer; AWhiteList: string; ABlackList:string );
// присвоение картинок копкам на формах
// AIndex - формат картинок
// AWhiteList - белый список, изменять только указанные формы
// ABlackList - черный список, исключить формы
var
  tmpForm: TAForm;
  tmpButton: TdbButton;
  tmpName: string;
  i: integer;
  j: integer;
  tmpByName: boolean;
  tmpFormList: string;
begin
  if AWhiteList <> '' then
  begin
    tmpByName := True;
    tmpFormList := ','+AWhiteList+',';
  end
  else
  begin
    tmpByName := False;
    tmpFormList := ','+ABlackList+',';
  end;
  for i := 0 to Screen.FormCount - 1 do
  begin
    tmpForm := TAForm(Screen.Forms[i]);
    if ( tmpByName and (pos( tmpForm.name, tmpFormList )>0) ) or ( not tmpByName and
(pos( tmpForm.name, tmpFormList )=0) )  then
    for j:=0 to tmpForm.ComponentCount - 1 do
    begin
      if tmpForm.Components[j] is TdbButton then
      begin
        tmpButton := TdbButton(tmpForm.Components[j]);
        tmpName :=  DeleteClassName(tmpButton.Name);
        Images_Set( tmpButton, tmpName, AIndex, IMAGES_SIMPLE_MODEL );
      end;
    end;
  end;
end;
```

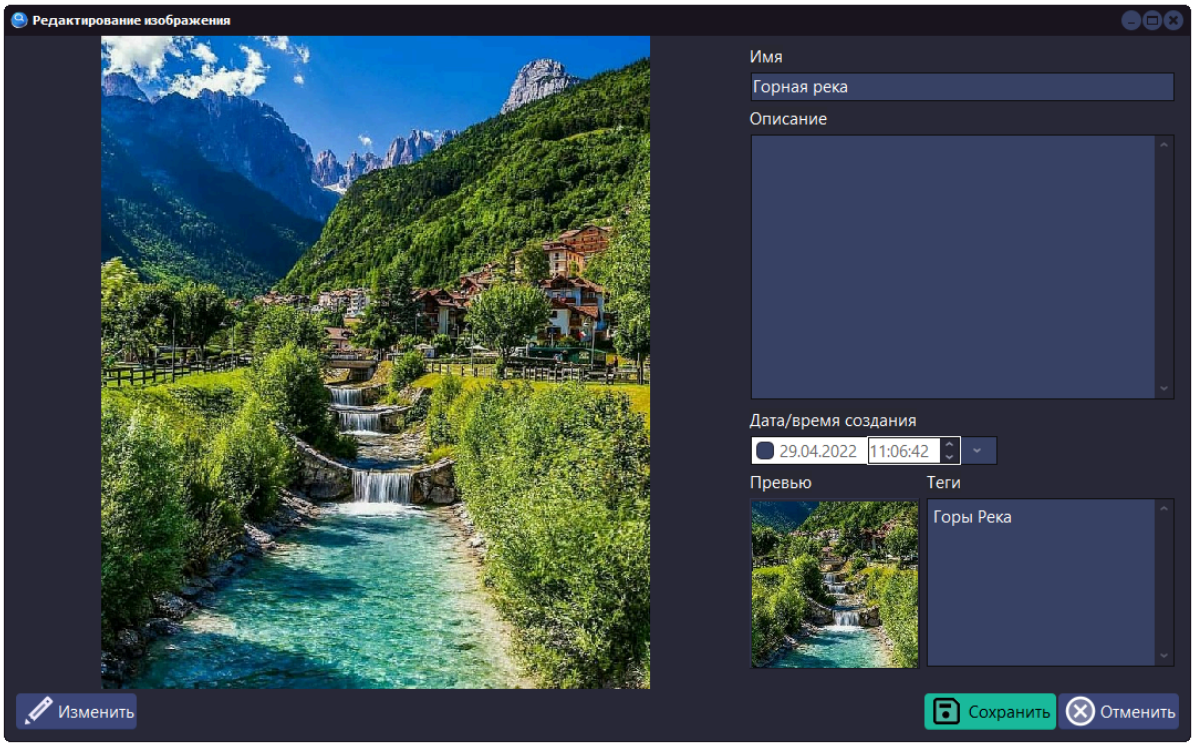
Последний метод - назначение картинки по имени. Он может потребоваться не только при инициализации кнопок, но и в ходе работы приложения, чтобы оперативно менять изображения на кнопке.

```
procedure Images_Set( AButton:TdbButton; AImageName:string; AIndexOfSize:integer;
AModel: integer );
// назначить индекс изображения по названию файла
// AButton:TdbButton; - кнопка
// AImageName:string; - имя изображения
// AIndexOfSize:integer; - индекс размера
// AModel: integer - модель
var
  tmpImageIndex: integer;
  tmpImageSelectedIndex: integer;
  tmpImageHotIndex: integer;
  tmpImageDisableIndex: integer;
  tmpImagePressedIndex: integer;
begin
  AImageName := UpperCase(AImageName);
  if AImageName <> '' then
  begin
    // основное изображение
    tmpImageIndex := ImagesNames[AIndexOfSize].IndexOf(AImageName);
    tmpImageSelectedIndex :=
ImagesNames[AIndexOfSize].IndexOf(AImageName+IMAGES_SELECTED_SUFFIX);
    tmpImageHotIndex :=
ImagesNames[AIndexOfSize].IndexOf(AImageName+IMAGES_HOT_SUFFIX);
    tmpImageDisableIndex :=
ImagesNames[AIndexOfSize].IndexOf(AImageName+IMAGES_DISABLE_SUFFIX);
    tmpImagePressedIndex :=
ImagesNames[AIndexOfSize].IndexOf(AImageName+IMAGES_PRESSED_SUFFIX);
    if tmpImageIndex >= 0 then
    begin
      AButton.Images := ImagesList[AIndexOfSize];
      AButton.ImageIndex := tmpImageIndex;
      case AModel of
      IMAGES_SIMPLE_MODEL: begin
        AButton.SelectedImageIndex := tmpImageSelectedIndex;
        AButton.HotImageIndex := tmpImageSelectedIndex;
        AButton.DisabledImageIndex := tmpImageIndex;
        AButton.PressedImageIndex := tmpImageSelectedIndex;
      end;
      IMAGES_FULL_MODEL: begin
        AButton.SelectedImageIndex := tmpImageSelectedIndex;
        AButton.HotImageIndex := tmpImageHotIndex;
        AButton.DisabledImageIndex := tmpImageDisableIndex;
        AButton.PressedImageIndex := tmpImagePressedIndex;
      end;
```

```
    end;
  end;
 end;
end;
```

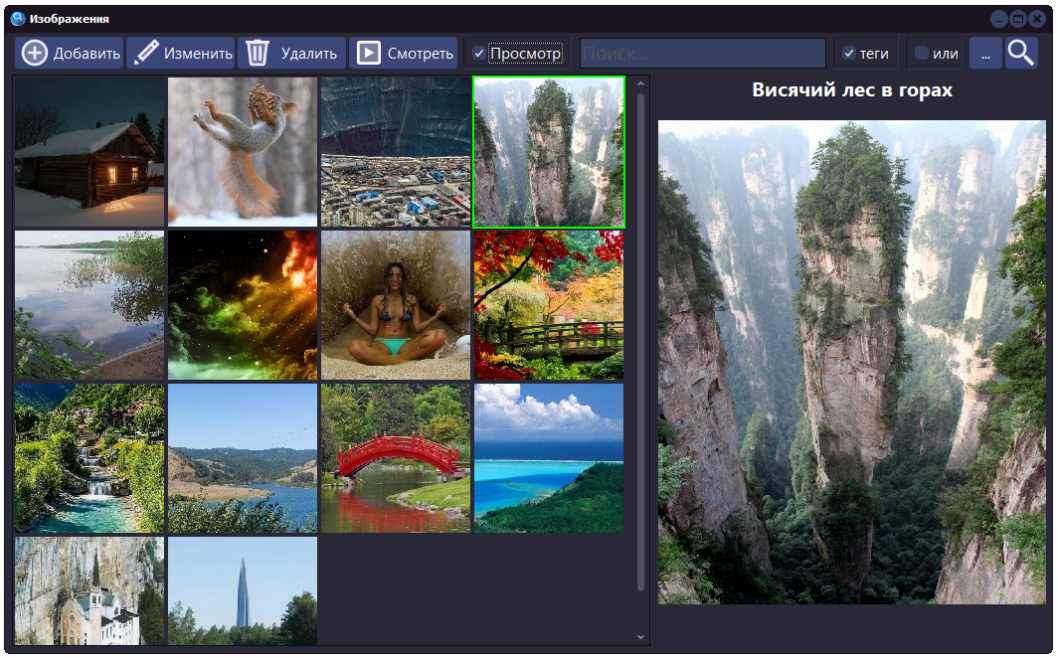
В результате приложение обзавелось стильными картинками на кнопках:



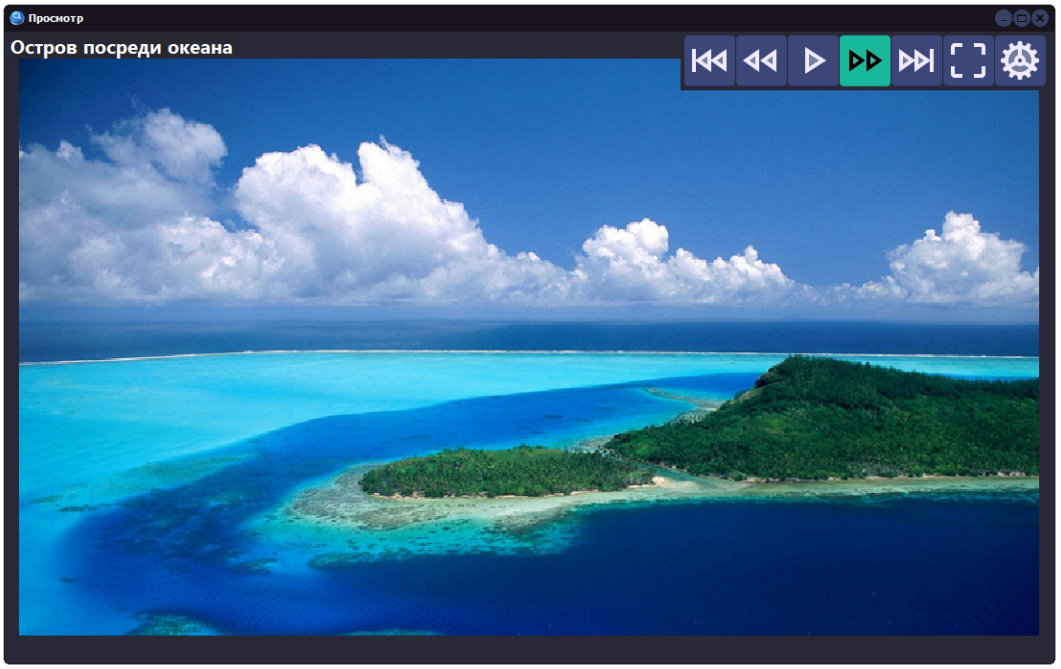


Режим презентации

После того, как нужные изображения найдены, их можно просмотреть в области предварительного просмотра, переключая изображение мышкой.



Но гораздо удобней использовать для этого отдельную форму, на которой будут находиться только просматриваемое изображение, название, описание и кнопки управления просмотром.



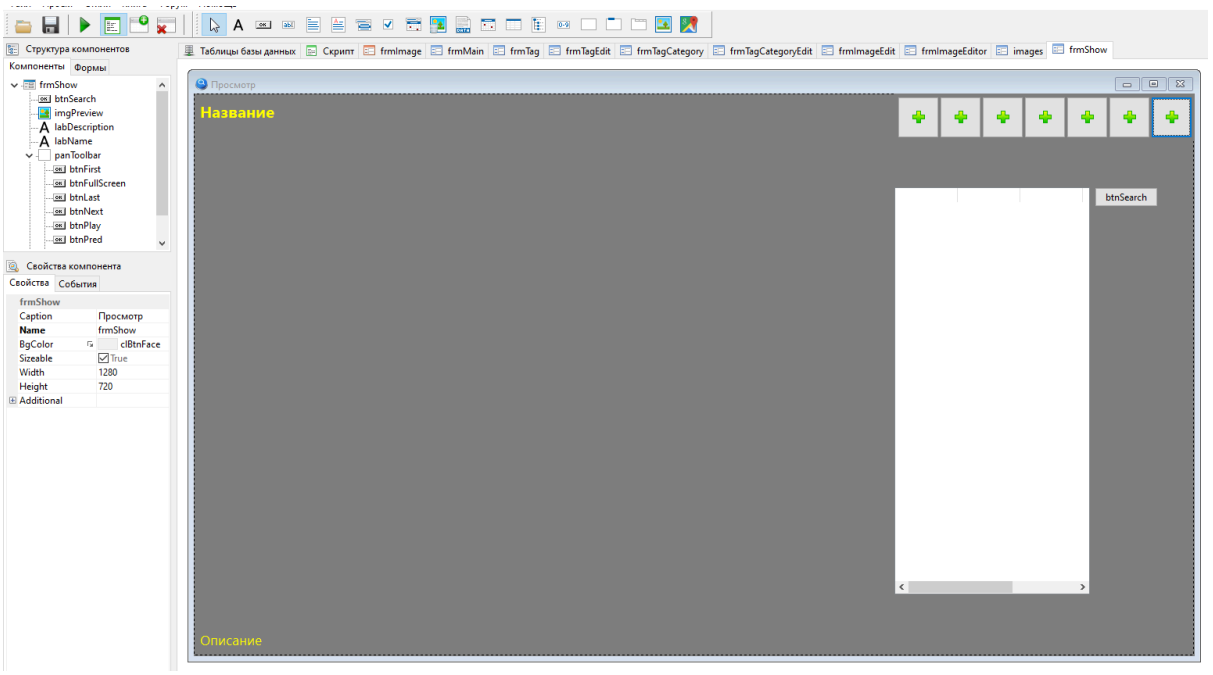
А самое главное - там будет кнопка непрерывного воспроизведения, при нажатии которой программа будет автоматически показывать следующее изображение из списка.

Для реализации режима презентации нам понадобится таймер, который будет создаваться при первом нажатии на кнопку воспроизведения:

```
procedure frmShow_btnPlay_OnClick (Sender: TObject; var Cancel: boolean);
// нажатие кнопки Play
begin
  if ShowTimer = nil then
  begin // первый запуск
    ShowTimer := TTimer.Create(frmMain);
    ShowTimer.Interval := 3000; // интервал смены картинки
    ShowTimer.OnTimer := @frmShow_ShowTimer_OnTimer;
    ShowTimer.Enabled := False;
  end;
  if ShowTimer.Enabled then
  begin
    ShowTimer.Enabled := False;
    Images_Set(frmShow.btnPlay, 'play', IMAGES_FORMAT_BIGBUTTON,
IMAGES_SIMPLE_MODEL);
  end
  else
  begin
    ShowTimer.Enabled := True;
    Images_Set(frmShow.btnPlay, 'pause', IMAGES_FORMAT_BIGBUTTON,
IMAGES_SIMPLE_MODEL);
  end;
end;
```

Здесь нам пригодится процедура Images_Set(), с помощью которой будет меняться внешний вид кнопки воспроизведения при её нажатии.

Для навигации по изображениям нам понадобится невидимая таблица. Использовать для этого TDataSet не получится, так как он однонаправленный, и перемещаться по нему по записям можно только от начала к концу.



Зато можно использовать тот же самый запрос, что и при формировании галереи. Для навигации используется свойство SelectedRow:

```
procedure frmShow_btnNext_OnClick (Sender: TObject; var Cancel: boolean);
// нажатие кнопки Next
begin
  if frmShow.tgrImageList.SelectedRow < (frmShow.tgrImageList.RowCount - 1) then
    frmShow.tgrImageList.SelectedRow := frmShow.tgrImageList.SelectedRow + 1;
  frmShow_LoadData;
end;
```

А для загрузки изображения используем метод виртуального класса ImageEdit:

```
procedure frmShow_LoadData;
// загрузка данных
var
  tmpRow: integer;
begin
  tmpRow := frmShow.tgrImageList.SelectedRow;
  if tmpRow <> -1 then
  begin
    frmShow.labName.Caption := frmShow.tgrImageList.cells[1,tmpRow];
    frmShow.labDescription.Caption := frmShow.tgrImageList.cells[2,tmpRow];
    ImageEdit_LoadImageFromBase( frmShow.imgPreview,
StrToInt(frmShow.tgrImageList.cells[0,tmpRow]), IE_FT_ORIGINAL, IE_IT_IMAGE);
  end;
end;
```

Тестирование

В результате тестирования выявлена серьёзная проблема - низкая производительность. При загрузке крупного изображения с диска возникают задержки в несколько секунд (!). И это ставит большой жирный крест на той части программы, которая отвечает за создание альбомов и презентаций: время загрузки страницы альбома будет неприемлемым. А об эффектах и говорить нечего: обычная смена свойства Visible у компонента с загруженной картинкой высокого разрешения может занимать секунду и более. Не помогли трюки с двойной буферизацией отображения формы - при смене видимости компонента картинка “мигает”. Из этого следует, что

Компоненты, используемые в **My Visual Database** не предназначены для эффективной работы с графическим контентом и могут использоваться только как вспомогательные элементы интерфейса с рядом ограничений.

Изменение планов

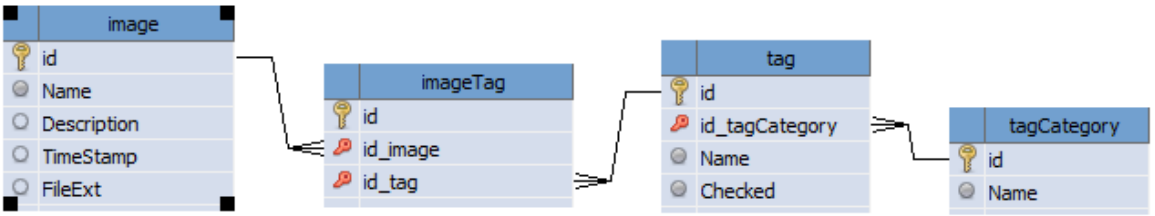
Конечно, неприятно признаваться, что планы по созданию альтернативного PowerPoint потерпели крушение, но такие драмы периодически случаются в мире программирования. Несмотря на то, что достичь ожидаемого результата не удалось, программа обладает несколькими универсальными решениями, которые можно использовать и в других приложениях: виртуальные классы **ImageEdit** и **Images, поиск данных по тегам.**

Проект навсегда останется в категории учебных. Для завершения работы из него необходимо удалить все неиспользуемые таблицы и формы.

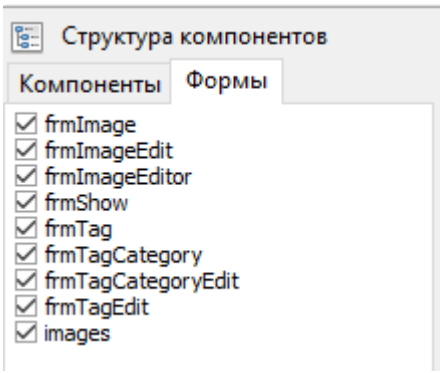
А реализацию полноэкранного просмотра и настройку параметров презентации я оставляю читателям данной книги ;-)

Финальная версия

Структура данных существенно упростилась:



После удаления несостоявшейся формы для просмотра перечня альбомов, список форм тоже поуменьшился.

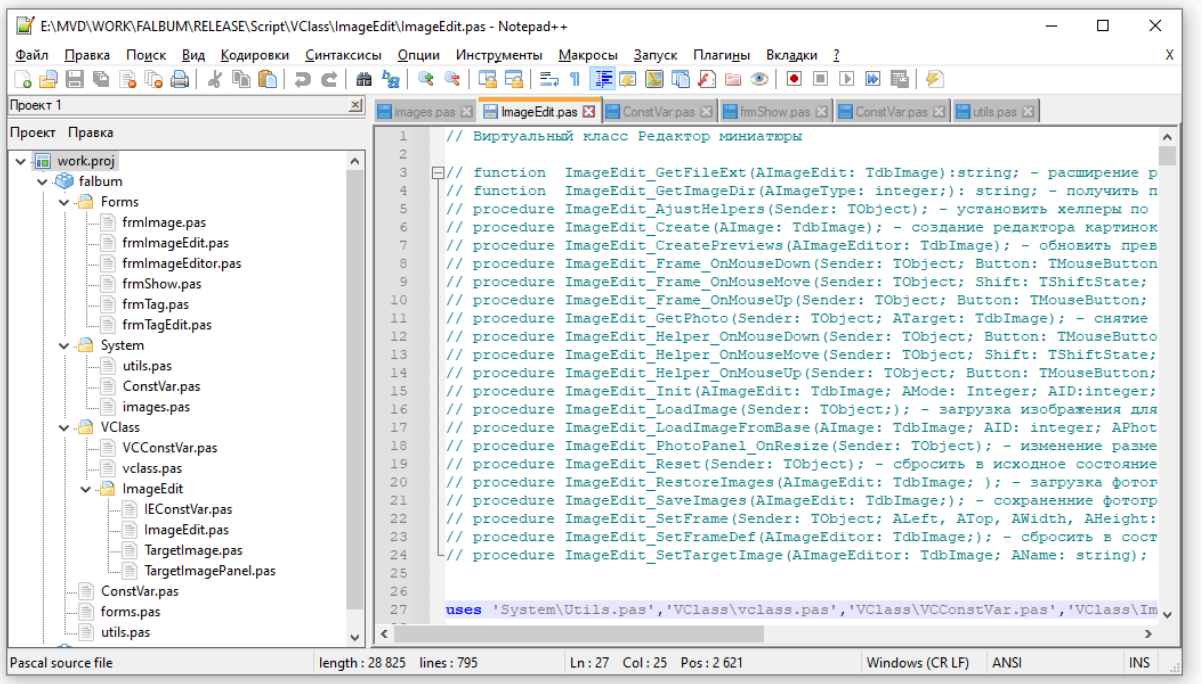


Файл Script.pas также приобрёл лаконичную форму:

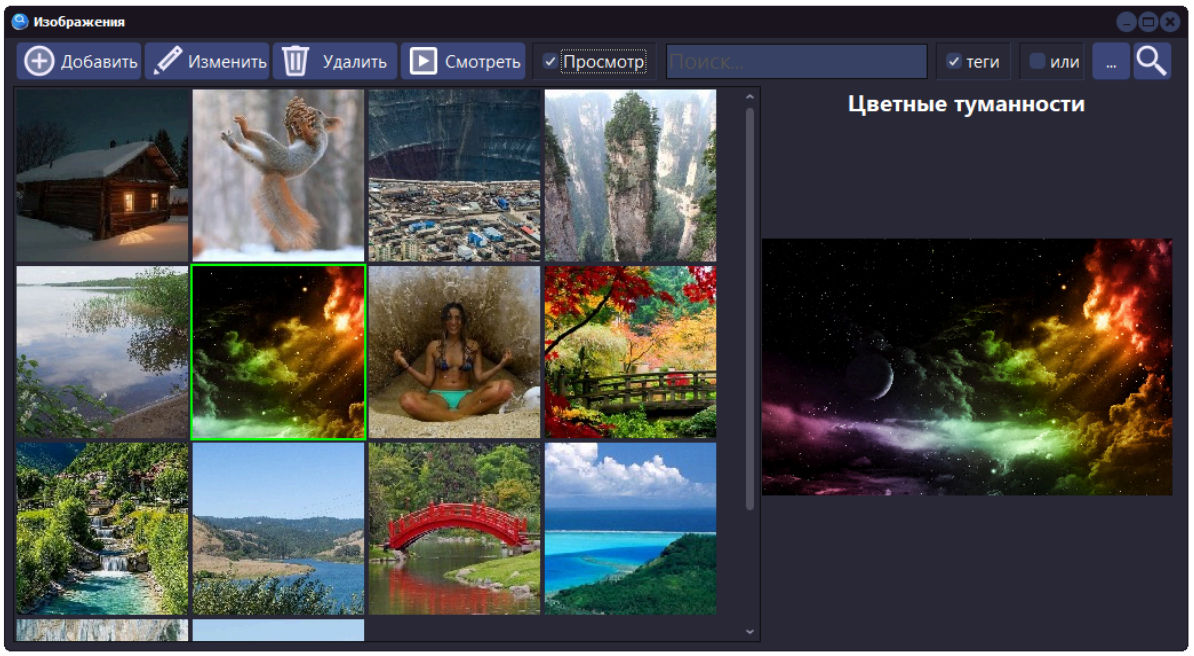
```
uses
  'ConstVar.pas',
  'Forms.pas',
  'system\Images.pas',
  'utils.pas';

begin
end.
```

А все скрипты удобно разместились в отдельных файлах, для редактирования которых я использую универсальный редактор программиста Notepad++.



Главной формой я сделал форму отображения галереи изображений:



Ссылки:

- Исходные коды проекта