Структура проекта:

Файлы с разрешением \*.dpr – это текстовые файлы, которые используется для хранения информации о формах и модулях. В них содержатся операторы инициализации и запуска программ на выполнение.

Файлы с разрешением \*.dfm –двоичные или текстовые файлы, которые создается для хранения информации о формах. Каждому файлу формы соответствует файл модуля (.pas)

Каждой создаваемой форме соответствует текстовый файл модуля с разрешением \*.pas, используемый для хранения кода. Многие из функций и процедур Delphi хранятся в модулях.

Подробнее рассмотрим структуру модулей.

UnitAuth.pas– cодержит в себе обработчики формы логику авторизации пользователей в системе.

UnitSuperUser.pas– содержит в себе логику для работы пользователя с ролью заведующий библиотекой.

UnitLibrarianAccount.pas– содержит в себе обработчик формы для пользователя библиотекарь.

UnitReaderAccount.pas– содержит в себе обработчик формы для пользователя читатель.

UnitExemplar.pas – Файл, содержащий форму для добавления и редактирования информации о экземпляре книги.

UnitBook.pas – Файл, содержащий форму для создания, редактирования и просмотра информации о книгах.

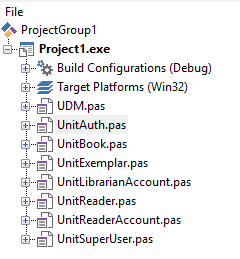


Рисунок. 3.2.1. Структура модулей

Рисунок. 3.2.2. Структура взаимосвязей модулей

Единственной не визуальной формой является DataModule. Она представляет собой форму-компонент для размещения на нем компонентов, предназначенных для соединения с базой данных.

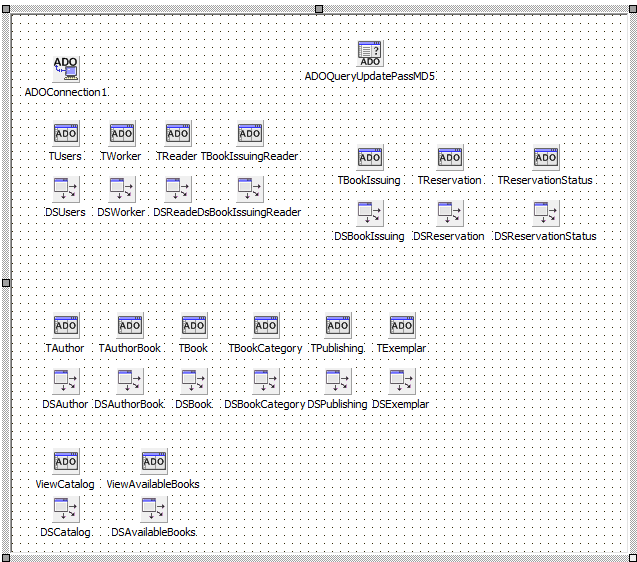


Рисунок. 3.2.2. Data Module

Схема соединения приложения с базой данных изображена на рисунке 3.2.3.

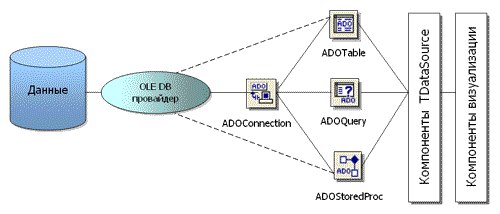


Рисунок. 3.2.4. Схема соединения

ADOConnection является компонентом, предназначенным для соединения с базой данных. Свойство компонента строка соединения формируется автоматически. В качестве поставщика данных используется «Microsoft OLEDB Providerfor SQL Server». В наименовании сервера указывается именованный экземпляр, созданный при установке MSSQLSQLServer, либо сервер, используемый по умолчания на локальной машине – (local). Для входа на сервер базы данных используется средство встроенной безопасности Window NT. Имя базы данных «LibraryDB».

После установления соединения с сервером базы данных необходимо настроить компоненты для отображения информации получаемой и базы данных.

ADOTable является компонентом, отвечающим за получение данных. Компонент из базы данных получает полную копию таблицы и хранит ее на всем протяжении работы приложения, а также по мере изменений содержания таблицы, синхронизирует ее состояние с состоянием реальной в базе данных.

Компонентом для отображения информации из набора данных является DBGrid. Однако для его соединения с набором данных необходима связь. Эту задачу решает компонент DataSource, он является посредником между наборами данных и конечными визуальными компонентами. Все конечные визуальные компоненты содержат свойство DataSource.

Что бы настроить корректное отображение информации в таблице достаточно в свойстве DataSource указать необходимый набор данных.

В качестве компонентов для визуалиазции: DBGrid, DBEdit,DBLookupComboBox. В этих компонентах уже заложена логика отображения информации из базы данных, достаточно только указать им нужный источник данных. Для добавления новых записей в набор данных с данными компонентами используются такие функции как Insert/Addend, для редактирования записи Editи для сохранения изменений Post.

По мере возрастания сложности запросов, появления больших межтабличных запросов, встроенного функционала стандартных компонентов ADOTable становится недостаточно и появляется необходимость использовать компонент ADOQuery.В отличие от ADOTable, запрос не содержит свойств для выбора таблицы, которую он будет отображать. Вместо этого он имеет поле SQL, которое должно содержать текст SQL-запроса к БД для выбора необходимых данных. Запросы, которые содержали в себе значительный объем текста были сохранены в БД в представлении.

Представление (VIEW) - объект данных который не содержит никаких данных его владельца. Это - тип таблицы, чье содержание выбирается из других таблиц с помощью выполнения запроса. Поскольку значения в этих таблицах меняются, то автоматически, их значения могут быть показаны представлением. Поэтому работа с представлениями с представлениями не отличается от работы с обычной таблицей в понимании использования компонентов ADOTable.

Основные процедуры:

Модуль «UnitFormAuth» предназначен для авторизации пользователя в системе и последующего доступа к ней в зависимости от роли авторизованного пользователя.

procedure TFormAuth.BitBtn1Click(Sender: TObject);

begin

ADOQuery1.Close;

// передача в параметры sql запроса данных введенных пользователем

ADOQuery1.Parameters.ParamByName('l').Value := EditLogin.Text;

ADOQuery1.Parameters.ParamByName('p').Value := EditPass.Text;

// выполнение запроса

ADOQuery1.Open;

if (ADOQuery1.RecordCount > 0) then

begin

if ADOQuery1.Fields[1].AsString = 'reader' then

begin

dm.TReader.Locate('id', ADOQuery1.Fields[0].AsInteger,

[loCaseInsensitive, loPartialKey]);

FormReader.Show;

end;

if ADOQuery1.Fields[1].AsString = 'librarian' then

begin

dm.TWorker.Locate('id', ADOQuery1.Fields[0].AsInteger,

[loCaseInsensitive, loPartialKey]);

FormLibrary.Show;

end;

if ADOQuery1.Fields[1].AsString = 'admin' then

begin

dm.TWorker.Locate('id', ADOQuery1.Fields[0].AsInteger,

[loCaseInsensitive, loPartialKey]);

FormSuper.Show;

end;

hide;

end

else

ShowMessage('Неверный логин или пароль');

end;

В компоненте ADOQuery1 в свойстве SQL содержится следующий текст запроса:

SELECT

user\_id

,type

FROM LibraryDB.dbo.Users

WHERE login =:l

AND passMD5 = CAST(HASHBYTES('MD5', :p) AS BINARY(16))

Здесь следует пояснить последнюю строчку. Парили в забе данных хранятся в зашифрованном виде с использованием шифромания MD5. Оператор HASHBYTES (Transact-SQL) – возвращает хэш MD2, MD4, MD5, SHA1 или SHA2 входного значения в SQL Server.

Модуль «UnitLibrarianAccount» содержит в себе обработчики формы личного кабинета библиотекаря. Здесь реализован весь функционал из перечня задач библиотекаря. Особенность данного модуля заключается в удобном поиске и просмотре информации из разных таблиц базы данных. Запросы используемые для отображения информации в данном модуле были довольно большие по объему, поэтому часть из них была сохранена на стороне базы данных в виде представления (view). Далее информация из данного представления использовалась как обычная таблица базы данных и подключалась через компонент ADOTable. Ниже приведен код содержимого представления «Catalog», предназначенного для отображения информации о всех книгах в библиотеке с подсчетом суммарного количества экземпляров каждой книги.

SELECT

b.ISBN ,bc.name AS 'category' ,b.Title ,FirstName + ' ' + SUBSTRING(a.LastName, 1, 1) + '.' + SUBSTRING(a.MiddleName, 1, 1) + '.' AS 'Author' ,p.name ,p.city ,b.year ,COUNT(\*) AS 'Count'

FROM LibraryDB.dbo.Exemplar e

LEFT JOIN Book b

ON e.ISBN = b.ISBN

LEFT JOIN BookCategory bc

ON b.category = bc.id

LEFT JOIN Publishing p

ON b.publishing\_id = p.id

LEFT JOIN AuthorBook ab

ON b.ISBN = ab.ISBN

LEFT JOIN Author a

ON ab.author\_id = a.id

GROUP BY b.ISBN

,bc.name,b.Title, FirstName + ' ' + SUBSTRING(a.LastName, 1, 1) + '.' + SUBSTRING(a.MiddleName, 1, 1) + '.' ,p.name ,p.city ,b.year

Следующее представление «AvailableBooks» - предназначено для отображения сводной информации об экземплярах книг, доступных для бронирования и выдачи. В таковым относятся те, которые в данный момент времени находятся не на руках у читателя, а в библиотеке, а также книги, которые не были ранее забронированы читателями и ожидающие их выдачич.

SELECT

ex.id ,b.ISBN ,b.Title ,b.publishing\_id ,p.name AS 'publishing'

,b.year

,b.category AS 'category\_id'

,bc.name AS 'category'

,ex.placement

FROM LibraryDB.dbo.Exemplar ex

LEFT JOIN Book b

ON ex.ISBN = b.ISBN

LEFT JOIN publishing p

ON b.publishing\_id = p.id

LEFT JOIN BookCategory bc

ON b.category = bc.id

WHERE ex.id NOT IN (SELECT

bi.Exemplar\_id

FROM BookIssuing bi

WHERE bi.DateReturnReal IS NULL)

AND ex.id NOT IN (SELECT

r.Exemplar\_id

FROM Reservation r

LEFT JOIN ReservationStatus rs

ON r.status = rs.id

WHERE rs.name IN ('Забронирован', 'Ожидание выдачи'))

В данном модуле практически у каждой таблицы есть возможность поиска по ней. Однако от таблицы к таблице суть этого поиска не изменяется – везде применяется стандартное свойство компонента ADOTable - Filtered. Свойство Filter - наиболее часто используемый способ фильтрации записей, имеет тип String. Для фильтрации таблицы вначале задается условие фильтрации в этом свойстве, затем логическому свойству Filtered присваивается значение True, после чего таблица будет отфильтрована. Общий обработчик фильтрации по заданному полю имеет следующий вид:

procedure TDM.DoIncrementalFilter(DataSet: TDataSet;

const FieldName, SearchTerm: string);

begin

Assert(Assigned(DataSet), 'No dataset is assigned');

if SearchTerm = '' then

DataSet.Filtered := False

else

begin

DataSet.Filter := FieldName + ' LIKE ' + QuotedStr(SearchTerm + '\*');

DataSet.Filtered := True;

end;

end;

Для его использования достаточно задать набор данных для фильтрации, столбец по которому будет происходить поиск, а также искомое значение. Так, например, для поиска книг на форме был создан выпадающий список с предварительно заданным содержимым (свойство Items): Название, Категория, Автор, Издательство. Добавлен компонент TSearchBox и создан следующий обработчик поиска на заданному пользователем полю:

procedure TFormLibrary.SearchBoxCatalogInvokeSearch(Sender: TObject);

var

FieldName, SearchTerm: string;

begin

case ComboBoxBookSearch.ItemIndex of

0:

FieldName := 'Title';

1:

FieldName := 'category';

2:

FieldName := 'Author';

3:

FieldName := 'name';

end;

SearchTerm := SearchBoxCatalog.Text;

DM.DoIncrementalFilter(DM.ViewCatalog, FieldName, SearchTerm);

end;

Изначально определяется по какому полю происходит поиск, затем название FieldName записывается название столбца, по которому будет происходить поиск и в ранее описанную процедуру DoIncrementalFilter передаются все необходимые параметры. После этого набор данных будет отфильтрован и DBGrid отобразятся записи, удовлетворяющие условию фильтра.

Помимо просмотра и поиска информации роль рядового библиотекаря подразумевает и регистрацию читателя в библиотеке, а также выдачу и возврат книги.

Далее рассмотрим процедуру, позволяющую создать нового читателя в системе.

//

procedure TFormLibrary.CreateReaderClick(Sender: TObject);

begin

DM.TReader.Insert; //открываем набор данных для вставки записи

case FormAddReadTicket.ShowModal of

mrOk:

//отображает модально форму в которой пользователю необходимо будет заполнить поля и нажать на кнопку «Сохранить»

begin

DM.TReader.Post; //сохраним введенную информацию о читателе

DM.TUsers.Insert; // откроем набор данных для вставки записи

DM.TUsers.FieldByName('type').Value := 'reader'; //устанавливаем тип пользователя

DM.TUsers.FieldByName('user\_id').Value := DM.TReader.Fields[0].Value;

DM.TUsers.FieldByName('login').Value :=

FormAddReadTicket.EditLogin.Text;

DM.TUsers.FieldByName('password').Value :=

FormAddReadTicket.EditPass.Text;

DM.TUsers.Post;

// обновим пароль MD5

DM.ADOQueryUpdatePassMD5.ExecSQL;

end;

mrCancel:

begin

DM.TReader.Cancel;

DM.TUsers.Cancel;

end;

end;

end;

Приступим к рассмотрению одной из важных функций библиотекаря – выдача книги. Перед выполнением необходимо перевести набор данных в режим редактирования. Со стороны пользователя произойдет эт после нажатия на кнопку «Открыть лист выдачи». После того как состояние набора данных DM.TBookIssuing переведено в режим редактирования, и пользователь ввел всю необходимую информацию о выдаче, такую как код экземпляра, читатель и дата выдачи, необходимо применить сохраненный изменения вызовом метода Post у набора данных DM.TBookIssuing.

procedure TFormLibrary.ButtonBookIssueSaveClick(Sender: TObject);

begin

if DM.TBookIssuing.State in [dsInsert] then

begin

DM.TBookIssuing.FieldByName('DateIssue').Value := DateTimePicker1.DateTime;

DM.TBookIssuing.FieldByName('DateReturnExpected').Value :=

DateTimePicker2.DateTime;

DM.TBookIssuing.FieldByName('Worker\_id').Value := DM.TWorker.Fields

[0].Value;

DM.TBookIssuing.Post;

GroupBoxBookIssue.Enabled := false;

end;

end;