Praktyczna aplikacja bazodanowa : Program do fakturowania cz. II

SZKIELET APLIKACJI

Interfejs użytkownika

Nadszedł czas byśmy wreszcie uruchomili zintegrowane środowisko programowania Delphi. Za jego pomocą przygotujemy wszystkie niezbędne formularz i raporty oraz wprowadzimy odpowiedni kod sterujący wykonywanymi operacjami. Rozpoczniemy od utworzenia formularza głównego, uruchamianego zaraz po starcie aplikacji. Będzie on pełnił rolę "centrum dowodzenia", które udostępnia różne funkcje programu. Jednak zanim przystąpimy do kodowania, musimy na chwilę cofnąć się do etapu projektu i przyjrzeć się ogólnej koncepcji naszej aplikacji pod kątem jej wewnętrznej struktury i właściwości interfejsu użytkownika

Konieczne jest rozważenie takich właściwości interfejsu jak :

- egronomiczność
- intuicyjność
- atrakcyjność
- ajednorodność

Co rozumiemy przez ergonomiczność interfejsu? Oczywiście chodzi tu o łatwy dostęp do najważniejszych funkcji. W każdej aplikacji są opcje bardzo często wykorzystywane (u nas będzie to moduł odpowiedzialny za wystawianie faktur) i na nich powinniśmy skoncentrować uwagę. Należy je tak zaprojektować, aby korzystanie z nich było jak najwygodniejsze. Musimy pamiętać na przykład, że użytkownik będzie wystawiał ponad sto faktur dziennie, co oznacza ,że każde niepotrzebne kliknięcie przycisku lub zbędny ruch myszą będą przez niego traktowane jako uciążliwość. Warto również zadbać o to, by dla każdej operacji dostępnie za pomocą myszy istniał również odpowiednik klawiaturowy. Praca w aplikacjach bazodanowych z reguły polega na wpisywaniu tekstu, co oznacza ,że każde oderwanie rąk od klawiatury wymaga dodatkowego czasu i przeniesienie uwagi na mysz czy inne urządzenie. Intuicyjność interfejsu w pewnym sensie decyduje o ergonomiczności. Wyróżniamy ją jednak tutaj , aby podkreślić szczególne znaczenie tego czynnika dla użytkownika dopiero poznającego aplikację. Ergonomiczność , tak jak ją opisaliśmy wcześniej, jest istotna szczególnie z punktu widzenia doświadczonego użytkownika aplikacji, który dąży do maksymalnej optymalizacji wykonywanych przez siebie czynności. Z kolei mówimy ,że aplikacja ma intuicyjny interfejs, jeśli "dana funkcja programu jest tam, gdzie się jej można najbardziej spodziewać". Rozważmy ten problem na przykładzie. Gdy wyświetlana jest lista kontrahentów, warto umieścić obok niej przycisk Drukuj, który umożliwi wydruk list ogólnych, list zobowiązań, obrotów itp. Natomiast analogiczny przycisk obok listy towarów powinien uruchamiać wydruk stanów magazynowych, wysokości sprzedaży itp. "Nieintuicyjny" program mógłby wyglądać następująco - lista kontrahentów umożliwiałaby co prawda przejście do kartoteki określonego klienta, ale wydruk dostawców wymagałby powrotu do menu głównego, wejścia do menu Wydruki i wybranie opcji Dostawcy. Atrakcyjność interfejsu nie wymaga chyba specjalnego definiowania. Zależy nam na tym, aby formularze były czytelne, przejrzyste i "przyjemne dla oka", a nie odstręczające. W tym miejscu warto jednak ostrzec przed "wodotryskami" - nadmiar kolorów, animacji i efektów specjalnych może utrudnić korzystanie z programu. Przerost formy nad treścią jest równie niewskazany , jak zupełny brak formy. Ostatni element związany z interfejsem to jego jednorodność. Jeżeli na przykład na jednym formularzu znajduje się przycisk Zapisz, to na innym formularzu powinien on mieć takie same rozmiary, taką samą ikonę i znajdować się w takim samym miejscu. Ułatwi to użytkownikowi naukę korzystania z programu i jego dalsze stosowanie

Wewnętrzna struktura aplikacji

Wiele elementów aplikacji , choć ukrytych przed użytkownikiem, ma kluczowe znaczenie dla jej poprawnego działania i dalszej rozbudowy. Aplikacja kliencka powinna spełniać dwa podstawowe kryteria:

- działać poprawnie
- cechować się skalowalnością

Poprawność działania aplikacji to warunek minimum jej użyteczności. Jeśli program bazodanowy nieprawidłowo analizuje zgromadzone dane lub ,co gorsza, powoduje zapis błędny bądź niespójnych informacji w tabelach , dyskwalifikuje go to całkowicie. Warto tutaj przypomnieć ,że choć aplikacja kliencka może (a wręcz powinna) weryfikować pewne aspekty poprawności danych, to generalnie mechanizmy te należy definiować na możliwe najniższym poziomie. Jeżeli na przykład między tabelami Kontrah i TranNagl ma zachodzić relacja typu jeden - do - wielu, warto ją zdefiniować na poziomie struktury bazy danych, a nie dopiero w aplikacji klienckiej. Takie podejście ma same zalety:

- podnosi bezpieczeństwo danych
- zmniejsza ryzyko utraty integralności referencyjnej
- upraszcza kod aplikacji, co jednocześnie poprawi jej niezawodność

Jednak pewne elementy można sprawdzić tylko na poziomie aplikacji klienckiej, co oznacza ,że w kodzie opracowywanych modułów formularzy muszą się znaleźć odpowiednie mechanizmy weryfikujące. Skalowalność aplikacji ma dwa aspekty : ilościowy i jakościowy .W pierwszym przypadku - skalowalności ilościowej chodzi przede wszystkim o to, aby zwiększająca się objętość danych nie powodowała drastycznego spowolnienia działania aplikacji użytkowych. Odpowiednie decyzje związane ze skalowalnością ilościową powinny być podejmowane już w fazie projektowanie. Jeżeli przewidujemy, że nasza aplikacja będzie wykorzystywana tylko przez kilku użytkowników i znajdzie się w niej maksymalnie kilkaset tysięcy rekordów, wystarczy utworzenie systemu jednowarstwowego, w którym większość zadań związanych z przetwarzaniem danych wykonuje aplikacja kliencka. Gdy aplikacja jest przeznaczona dla wielu użytkowników i chcemy w niej gromadzić miliony rekordów, konieczne jest zastosowanie rozwiązania dwuwarstwowego (aplikacja klient - serwer) lub nawet trójwarstwowego (aplikacja kliencka - serwer aplikacji - serwer baz danych). Skalowalność jakościowa polega na takim zaprojektowaniu struktury bazy danych i aplikacji, aby łatwo je było rozbudowywać. O skalowalności bazy danych zadbaliśmy , na przykład definiując tabelę TranNagl. Wstawiliśmy do niej dodatkowe pole, które umożliwi podział transakcji na sprzedaż , zakup i inne dokumenty. Skalowalność aplikacji wiąże się również z tym by dodawanie kolejnych elementów nie powodowało zaburzenia działania dotychczasowych funkcji i nie wymagało przebudowy całego programu. Podczas projektowania kolejnych elementów programu przez cały czas musimy mieć na uwadze powyższe zalecenia

Aplikacja Faktury - schemat działania

Podczas projektowania interfejsu użytkownika warto zaplanować sobie wcześniej, w jaki sposób będą na siebie oddziaływać poszczególne moduły programu. Taki schemat ułatwi w przyszłości tworzenie kodu. Aplikacja będzie rozpoczynała działanie od wyświetlenia formularza z głównym menu. Menu to pozwoli na bezpośrednie wyświetlenie formularzy z listami kontrahentów, towarów i faktur. Dopiero po wyświetleniu jednej z tych list użytkownik będzie mógł wybrać rekord konkretnego kontrahenta (towaru lub faktury) i przejść do formularza, w którym będą dostępne informacje szczegółowe. Takie rozwiązanie daje istotne korzyści: wyświetlanie rekordów w postaci tabeli ułatwia ich przeglądanie i wyszukiwanie; z drugiej strony , zastosowanie formularza prezentującego pojedynczy rekord pozwala na jednoczesne uzyskanie wszystkich zapamiętanych w nim informacji. Formularz główny będzie

również pozwalał na wyświetlanie formularzy z opcjami konfiguracji. Wydruki zbiorcze związane z fakturami (np. sprzedaż za dany okres, obroty z kontrahentem) są dostępne z formularza, w którym widzimy listę faktur. Z kolei w formularza, w którym edytujemy pojedyncze faktury, możemy uruchomić ich wydruk. Dzięki takiemu rozwiązaniu realizujemy postulat intuicyjności interfejsu. Ponieważ formularz główny będzie punktem wyjścia do wszystkich funkcji programu, tworzenie przykładowej aplikacji rozpoczniemy właśnie od niego

Formularz główny

Jeżeli myślisz o komercyjnym potraktowaniu tworzonej aplikacji, właściwie zaprojektowanie formularza głównego jest bardzo istotna. Nie na próżno mówi się ,że najważniejsze jest pierwsze wrażenie. Musisz zadbać przede wszystkim o wygodny dostęp do najczęściej wykorzystywanych opcji programu, ale nie zapomnieć o estetyce. Pierwszy ekran , który pojawi się przed oczami użytkownika musi go zachęcać do dalszej pracy z programem. Często aplikację rozpoczyna ekran powitalny (splash screen). Ma on dwojakie zastosowanie - pozwala zareklamować się twórcom programu i jednocześnie wypełnić jakoś czas potrzebny na wczytanie modułów startowych aplikacji. Tworzenie takiego ekranu jest bardzo proste (najczęściej jest to formularz bez żadnych aktywnych kontrolek), pominiemy więc ten etap i od razu przystąpimy do tworzenia formularza z głównym menu programu

Nowa aplikacja

W tej części rozpoczniemy tworzenie nowej aplikacji. Zadanie to jest bardzo proste - wystarczy uruchomić środowisko Delphi

1.Uruchom program Delphi. Na ekranie pojawi się pusty formularz główny nowego projektu .Z lewej strony ekranu powinny być widoczne okna Object TreeView oraz Object Inspector. Ponad pustym formularzem aplikacji znajduje się główne menu programu Delphi oraz paski narzędzi i przybornik z komponentami

2.W okienku Object Inspector przejdź do zakładki Properties, zaznacza właściwość Caption i w miejsce dotychczasowego tytułu formularza wpisz Moje faktury

Menu główne

Na przygotowanym formularzu możemy wprowadzić opcje menu i podmenu, które posłużą do uaktywniania różnych funkcji programu. Utworzymy trzy menu z menu z poleceniami : menu Transakcje z podmenu Sprzedaż, menu Słowniki z dwoma podmenu Kontrahenci i Słownik, wreszcie - menu Narzędzia zawierające polecenia Konfiguracja. Wykonaj poniższe kroki:

1.Upewnij się ,że na palecie komponentów aktywna jest zakładka Standard, kliknij ikonę komponentu MainMenu, a następnie kliknij w lewym górnym rogu formularza. Na formularzu pojawi się komponent MainMenu1

2.Dwukrotnie kliknij komponent MainMenu1. Pojawi się okno dialogowe edycji menu głównego

3.Gdy aktywne jest okno dialogowe Form1.MainMenu1, na karcie Properties okna dialogowego Object Inspector przejdź do właściwości Caption, wpisz Transakcje i naciśnij klawisz Enter. Do menu zostanie dodana nowa pozycja Transakcje, poniżej widoczne będzie zaznaczenie umożliwiające wprowadzenia polecenia menu

Po wstawieniu nowego menu jego właściwość AutoHotKeys ma wartość maAutomatic. Oznacza to ,że kompilator automatycznie dobierze klawisz skrótu dla danej opcji menu. Jako litera skrótu dobierana jest zawsze pierwsza wolna litera z nazwy opcji - w przypadku pozycji Transakcje będzie to litera T (po

uruchomieniu programu zostanie ona wyróżniona podkreśleniem; aby uaktywnić ją za pomocą klawiatury należy nacisnąć klawisze [lewy Alt + T]. Jeżeli na tym samym poziomie menu pojawi się kolejna opcja na literę T, na przykład Towary, kompilator przypisze klawisz skrótu do litery) itd. Jeżeli sami chcemy wybierać klawisze skrótu lub z jakichś względów chcemy z nich całkowicie zrezygnować, musimy zmienić właściwość AutoHotKeys na maManual. Aby zdefiniować literę skrótu, podczas wpisywania nazwy opcji menu we właściwości Caption należy poprzedzić odpowiednią literą znakiem &

4.Kliknij pozycję menu Transakcje , a następnie zaznacza wolną pozycję podmenu. Naciśnij Enter, aby przejść do właściwości Caption kolejnego polecenia menu, wpisz Sprzedaż i naciśnij Enter.

5.Naciśnij strzałkę w prawo. Menu Transakcje zostanie zwinięte, a na prawo od niego pojawi się niebieskie zaznaczenie i ramka

6.Naciśnij klawisz Enter, aby przejść do właściwości Caption polecenia menu i wpisz Słowniki

Podczas tworzenia menu, Delphi automatycznie generuje definicje pozycji menu typu TMenuItem. Każda z tych pozycji uzyskuje nazwę zgodną z tekstem wpisanym w jej właściwości Caption, zakończoną kolejną liczbą. Z uwagi na wymogi kompilatora z nazwy tej są usuwane niektóre niedozwolone znaki; dotyczy to między innymi spacji oraz polskich znaków diakrytycznych. I tak np. opcja menu o nazwie Środki trwałe będzie reprezentowana przez zmienną rodkitrwae1. Dla zachowania czytelności kodu w takich przypadkach warto skorygować właściwość Name tworzonej pozycji menu, wpisując na przykład SrodkiTrwale1

Jeśli zauważysz ,że w czasie wpisywania niektórych polskich znaków w polu Caption zamiast nich uaktywniają się różne opcje Delphi. Aby to usunąć, należy wprowadzić odpowiedni klucz w rejestrze ([HKEY_CURRENT_USER\Software\Delphi\x.0\Editor\Options] "NoCtrlAltKeys" = "1") .

7.Kliknij pozycję menu Słowniki a następnie kliknij wolną pozycję menu, która pojawia się poniżej

8.Przejdź do właściwości Caption zaznaczonego elementu i wpisz w niej Kontrahenci

9.Kliknij następną wolną pozycję menu i w jej właściwości Caption wpisz Towary i usługi

10.W analogiczny sposób utwórz menu Narzędzia ,a w nim polecenia Konfiguracja

11.Zamknij okno Form1.MainMenu1

Można powiedzieć ,że wstępna wersja formularza głównego jest już gotowa. Każda z utworzonych pozycji menu będzie powodowała uruchomienie jakiejś opcji programu ,na przykład wyświetlanie listy towarów lub kontrahentów. Zapiszmy naszą pracę na dysk

1.Zaznacz formularz i w Object Inspector zmień jego właściwość Name na MenuG1Form

2.Kliknij przyciks Save All na standardowym pasku narzędzi. Pojawi się okno dialogowe Save As. Należy na nim określić nazwę modułu w którym będzie przechowywany kod związany z formularzem głównym

3.W polu Nazwa pliku wpisz MenuG1

4.Odszukaj folder w którym chcesz przechowywać projekt aplikacji i kliknij Zapisz. Pojawi się kolejne okno dialogowe Save As, w którym należy podać nazwę całego projektu. Wprowadzoną tutaj nazwę uzyska również plik wykonywalny, który powstanie po skompilowanym kodu źródłowego

5.W polu Nazwa pliku wpisz Faktury i kliknij przycisk Zapisz

Informacje związane z formularzem są zapisywane w dwóch plikach. Kod języka ObjectPascal, obejmujący definicję klasę formularza, procedury obsługi, zmienne itp., jest zapisywany w pliku o rozszerzeniu .PAS, natomiast właściwości formularza i umieszczonych na nim komponentów są zapisywane w pliku o rozszerzeniu .DFM

Paski narzędzi

Pasek narzędzi nie jest niezbędnym elementem aplikacji, jednak jego zastosowanie ułatwia realizację postulatu ergonomiczności. Np. jeśli na pasku narzędzi umieścimy ikonę wywołującą formularz do wyświetlania faktur, użytkownik uzyska go za pomocą jednego kliknięcia - korzystając z menu musiałoby wykonać dwie operacje. Spróbujemy teraz utworzyć pasek narzędzi, którego przyciski będą wyświetlały następujące formularze:

- wystawianie faktury
- lista kontrahentów
- lista towarów i usług
- konfiguracja

Wykonaj następujące czynności:

1.Na palecie Win32 wybierz komponent ToolBar i umieść go na formularzu .W górnej części formularza pojawi się pusty pasek narzędzi o nazwie ToolBar1

2.Prawym przyciskiem myszy kliknij pasek narzędzi i wybierz polecenie New Button. Na pasku narzędzi pojawi się pusty przycisk.

3.Prawym przyciskiem myszy klikaj pasek narzędzi i wybieraj kolejno polecenia : New Separator, New Button, New Button, New Separator i New Button. Aby na utworzonym pasku narzędzi pojawiły się ikony przedstawiające poszczególne opcje, na formularzu musi się znaleźć komponent typu TImageList

4.Na palecie Win32 wybierz komponent ImageList i umieść go na formularzu (najlepiej w pobliżu komponentu MainMenu1)

5.Dwukrotnie kliknij komponent ImageList1. Na ekranie pojawi się okno dialogowe

6.Kliknij przycisk Add. Aby wypełnić pola przycisków, skorzystamy z przygotowanej uprzednio listy obrazów

7.W oknie dialogowym Add Images przejdź do folderu gdzie trzymasz ikony

8.Kliknij OK aby wczytać je do obiektu ImageList1. Do tworzenia ikon możesz użyć programu Image Editor dostarczanego wraz z Delphi. Możesz od razu utworzyć listę wszystkich ikon lub zdefiniować ikony pojedynczo, a następnie za pomocą przycisku Add dodawać je kolejno do obiektu ImageList

9.Zaznacz komponent ToolBar1

10.Przejdź do Object Inspectora i we właściwości Images ustaw wartość ImageList1

11.Kliknij Save As aby zapisać wprowadzone modyfikacje

Alias bazy danych

Aliasa bazy danych to umowan nazwa ,zastępująca pełną ścieżkę dostępu do plików. Jeśli chcemy zapewnić sobie swobodę w lokalizowaniu naszej aplikacji nie możemy na sztywno zakodować ścieżek dostępu, ponieważ w przypadku zainstalowania programu w innym katalogu nie można będzie

odnaleźć niezbędnych plików. Z drugiej strony zmuszanie użytkownika do zainstalowania programu w ściśle określonej lokalizacji jest rozwiązaniem co najmniej nieeleganckim. Aby uniknąć takich problemów stosuje się aliasy. Alias definiuje się tylko raz - w programie Database Desktop lub bezpośrednio w aplikacji użytkowej. Wszystkie następne odwołania do bazy danych nie podają jej lokalizacji, lecz posługują się aliasem. Jak można się domyślić, jeżeli z jakichś powodów lokalizacja bazy danych musi być zmieniona, wystarczy zmodyfikować alias, a wszystkie odwołania do bazy w dalszym ciągu będą działały prawidłowo. Aliasy można definiować bezpośrednio w programie Database Desktop lub w samej aplikacji użytkowej. Każde z tych rozwiązań ma swoje zalety i wady .Alias utworzony w programie Database Desktop jest przechowywany w pliku konfiguracyjnym IDAPI.CFG i mają do niego dostęp wszystkie aplikacje korzystające z aparatu BDE (Borland Database Enviroment), co jest niewątpliwie jego zaletą. Z drugiej strony aliasy BDE są dosyć kłopotliwe podczas przygotowania wersji instalacyjnej programu. Alias generowany w aplikacji użytkowej ma charakter doraźny - jest tworzony przez aplikację i przestaje istnieć natychmiast po jej zakończeniu (lub wcześniej, na skutek pewnych operacji w programie). Ma do niego dostęp tylko aplikacja, w której zostanie utworzony. Funkcjonalnie obydwa typy aliasów daję ten sam efekt - w komponentach operujących na tabelach nie musimy odwoływać się bezpośrednio do ścieżek dostępu, lecz stosujemy "wyrażenie zastępcze" - słowem ,alias. Aby utworzyć alias:

1.Przejdź do palety komponentów BDE

2.Przenieś na formularzu główny aplikacji komponent TDatabase .W tym celu kliknij go na palecie komponentów, a następnie kliknij formularz

3.Prawym przyciskiem myszy kliknij umieszczony na formularzu komponent Database1 i z menu podręcznego wybierz polecenie Database Editor. Pojawi się okno dialogowe MenuGlForm.Database1Database

4.W polu Name wpisz BAZAFAKT. Wpisana wartość będzie aliasem bazy danych używanym w tej aplikacji

5.Na liście DriveName odszukaj pozycję STANDARD. Sterowników standardowych należy używać , jeżeli baza danych korzysta z formatu Paradox lub dBaseIV

6. W polu Parameter overrides wpisz PATH = .\Dane. Wspomniane pole służy do wpisania wartości przesłaniających ustawienia domyślne. W tym przypadku, chcemy przesłonić domyślną wartość parametru PATH (pusty ciąg znaków), przypisując mu ścieżkę dostępu do plików naszej bazy danych. Kropka oznacza bieżący katalog. Wyrażenie ".\Dane" oznacza podkatalog Dane katalogu bieżącego. Wyjaśnijmy znaczenie pozostałych opcji widocznych w oknie MenuGLForm.Database1 Database .W polu Alias name można wyświetlić nazwę aliasu zdefiniowanego w aparacie BDE .W ten sposób nowo tworzony alias powstanie w oparciu o alias istniejący już w bazie danych. Jeżeli zaznaczone jest pole wyboru Login prompt w momencie łączenia się z bazą danych wymagającą zalogowania się, pojawia się okno dialogowe umożliwiające podanie nazwy użytkownika i hasła. Gdy ta opcja zostanie wyłączona , programista sam musi zapewnić przekazanie tych wartości (na przykład kodując je w programie) .Zaznaczone pole wyboru Keep inactive connection oznacza ,że połączenie bazą danych będzie utrzymywane, nawet jeżeli nie będzie ono korzystało z żadnego zestawu danych w bazie. Zapis PATH = .\Dane może stosować ,gdy w fazie projektowej chcemy mieć dostęp do bazy danych. Jednak w kodzie programu warto umieścić jawne przypisanie , deklarujące ,że katalog danymi jest podkatalogiem katalogu z plikiem wykonywalnym aplikacji. Nazwę katalogu , w której znajduje się plik wykonywalny, można uzyskać za pomocą wywołania ExtractFilePath(Application.ExeName)

7.Kliknij przycisk OK

8.Zaznacza formularza, przejdź do karty Event Object Inspector i dwukrotnie kliknij w polu obok zdarzenia OnCreate. Ścieżkę dostępu do danych należy ustalić zaraz po rozpoczęciu aplikacji

9.Uzupełnij procedurę zdarzenia , aby uzyskała postać :

```
procedure TMenuGlForm.FormCreate(Sender: TObject);

begin

with DataBase1 do begin

Connected := False;

Params.Clear;

Params.Add('Path=' + ExtractFilePath(Application.ExeName) + 'dane');

Connected := True;

end; end;
```

Powyższa procedura kolejno : rozłącza component DataBase1 ze źródłowoą bazą danych, czyści listę parametrów, wstawia nowy parametr PATH i ponownie łączy z bazą danych

10.Kliknij przycisk Save All aby zapisać wprowadzone modyfikacje

Formularz do edycji konfiguracji programu

Jak na razie nasze programowanie, rozumiane w tradycyjnym znaczeniu tego słowa jako pisanie kodu, ograniczyło się do wstawienia kilku wierszy. Jedną z zalet środowiska Delphi jest automatyzacja wielu procesów. Między innymi w trakcie tworzenia formularza głównego powstał obiekt dziedziczący po klasie TForm. Następnie w trakcie definiowania elementów menu, zostały do niej dołączone definicje komponentów typu TMainMenu i TMenuItem. Jeżeli chcesz zobaczyć definicję modułu MainGl, wciśnij klawisz F12 lub kliknij przycisk Toggle Form/Unit. Niestety (a może na szczęście) kodowania nie da się uniknąć. Kolejne linie kodu wpiszemy, tworzy moduł do edycji ustawień konfiguracyjnych naszego programu. Rozpoczniemy od wygenerowania nowego formularza, przeznaczonego do przechowywania informacji konfiguracyjnych. Każda większa aplikacja ma pewien zbiór ustawień. Z reguły są one dostępne dla użytkownika, który może je dowolnie modyfikować. Jednocześnie większość tych ustawień jest zapamiętywana, co pozwala na ponowne uruchamiania programu z taką samą konfiguracją .Wspomniane ustawienia mogą być zapisane w wielu miejscach. W tym celu wykorzystuje się najczęściej : rejestr systemu operacyjnego , plik bazy danych lub zewnętrzny plik tekstowy .W naszym programie zastosujemy ostatnie z wymienionych rozwiązań. Informacje konfiguracyjne będą zapisywane w pliku tekstowym o rozszerzeniu .INI .Delphi posiada wbudowane mechanizmy do pracy tego typu plikami, dzięki czemu pobieranie z nich wartości oraz modyfikowanie ich jest bardzo proste. Co powinno się znaleźć w pliku konfiguracyjnym? Ogólnie rzecz biorąc można stwierdzić ,że im więcej ustawień jest dostępnych dla użytkownika, tym lepiej. Jednak w naszej aplikacji ograniczymy się tylko do kilku niezbędnych elementów. W ustawieniach konfiguracyjnych będziemy przechowywali informacje umieszczane w nagłówku faktury, takie jak : domyślna seria i dopisek do numeru faktury ,dane adresowe sprzedawcy oraz miejsce wystawienia faktur. Pierwsze dwie wartości ułatwią użytkownikowi wystawianie faktur - jeżeli stosowana jest na przykład numeracja FA/1/2012, FA/2/2012 itd. warto zautomatyzować wpisywanie skrajnych wartości. Podobne znaczenie mają dane firmy - będą one automatycznie wstawianie do każdej drukowanej faktury

1.W menu File -> New, wybierz polecenie Others. Pojawi się okno dialogowe New Items

2.Przejdź do zakładki Dialogs, upewnij się , czy zaznaczona jest opcja Copy i dwukrotnie kliknij ikonę Standard Dialog (Horizontal). Na ekranie pojawi się nowy formularz. Będie na nim umieszczona ramka typu TBevel oraz dwa przyciski TButton. Właściwość BorderStyle tego okna ma wartość bsDialog, co oznacza ,że będzie ono wyświetlane w formie okna dialogowego (bez możliwości minimalizacji oraz maksymalizacji)

3.We właściwości Caption przycisku CancelButton wpisz Anuluj

4. We właściwości Caption nowego formularza wpisz Konfiguracja programu, a jego właściwości Name wpisz KonfigForm

5.Zapisz utworzony moduł formularza pod nazwą Konfig

6.Na formularzu wstaw cztery komponenty TLabel oraz siedem komponentów TEdit (z panelu Standard)

7.Zmień właściwości Caption kolejnych komponentów TLabel na:

Domyślna seria faktur: ; Domyślny dopisek dop numeru: ; Dane firmy do faktury : ; Miejsce wystawienia : ;

8.Rozmieść etykiety i pola tekstowe. W razie potrzeby zwiększ rozmiary formularza i ramki oraz przesuń przyciski OK i Anuluj

Tekstowy plik konfiguracyjny

Spróbujemy teraz utworzyć plik typu INI, w którym umieścimy ustawienia konfiguracyjne. Budowa takiego pliku jest bardzo prosta. Aby przypisać wartości poszczególnym ustawieniom, stosuje się znak równości, np.

dane1 = Serwis Komputerowy "KOMKOM"

W celu uproszczenia organizacji danych dzieli się je na sekcje; poszczególne sekcje są ciągami znaków umieszczonymi w nawiasach kwadratowych np.

[Faktury]

Jak wynika z utworzonego przed chwilą formularza, w projektowanym pliku powinno się znaleźć siedem pozycji. Do utworzenia pliku tekstowego użyjemy edytora Delphi, choć równie dobrze możemy posłużyć się np. Notatnikiem. Pliku INI nie musisz tworzyć w opisany tutaj sposób. Możesz pod razu wprowadzić kod procedur opisanych w dalszych częściach, skompilować i uruchomić program, przejść do formularza konfiguracyjnego, wypełnić odpowiednie pola i kliknąć OK. Jeżeli plik INI nie istnieje zostanie utworzony przez metodę Create. Z kolei metoda WriteString wstawi do nowego pliku odpowiednie klucze

1.W menu File -> New, wybierz polecenie Other. Pojawi się okno dialogowe New Items

2. Na karcie New odszukaj ikonę Text File i kliknij ja dwukrotnie .W oknie edytora Delphi pojawi się nowa zakładka File1.txt

3.W nowym pliku tekstowym wpisz poniższy tekst. Zwróć uwagę ,że przed i po znaku równości nie powinno być spacji:

[Faktury]

seria = FA

dopisek = 2012

[Nazwa]

dane1 = Serwis Komputerowy "KOMKOM"

dane2 = 01-124 Brutów, ul. Błotna 21a

dane3 = NIP : 666-555-44-33

dane4 = konto : BGG o/Brutów 1010999-123456789

[Inne]

miejsce-wystaw = Brutów

Jak widać, plik zawiera dwie sekcje, w pierwszej znajdują się dane firmy umieszczane na fakturze, w drugiej - informacje dotyczące numeru faktury. Oczywiście przedstawione tutaj opcje stanowią jedynie niezbędne minimum dla omawianej aplikacji, jednak przyjęcie tego rozwiązania nie ogranicza dalszej rozbudowy pliku INI i obsługującego go formularza (Jeżeli liczba opcji obsługiwanych przez ten formularza ma być duża , warto zastanowić się nad zastosowaniem komponentu typu TTabControl i rozmieszczenie opcji na różnych zakładkach)

4.Kliknij przycisk Sabe i zapisz plik w katalogu systemowym (np. C:\Windows) pod nazwą Faktury.ini.

Pliki INI domyślnie są przechowywane w katalogu systemowym. Jeżeli chcesz zmienić lokalizację pliku INI, musisz podać pełną ścieżkę dostępu

Odczyt danych z pliku INI

Kolejnym zadaniem jest odczyt z pliku INI i wyświetlenie ich w formularzu. Pierwsze pytanie, które musimy sobie postawić, brzmi : w jakim momencie odczytywać dane w pliku: odpowiednie działanie można wykonać w momencie tworzenia formularza (zdarzenie ONCreate), w momencie jego wyświetlenia (OnShow) lub w momencie uaktywnienia (OnActivate). Wybierzmy tą ostatnią opcję, gdyż daje nam ona gwarancję ,że w chwili przejścia do formularza konfiguracyjnego zawsze zobaczymy aktualną zawartość pliku INI. Oznacza to ,że musimy przygotować odpowiednią procedurę dla zdarzenia OnActivate formularza KonfigForm

1.Uaktywnij formularz KonfigForm

2.W Object Inspectorze przejdź do karty Events i odszukaj właściwość OnActivate, a następnie dwukrotnie kliknij pole wolne obok tej właściwości

3.Delphi wygeneruje powłokę procedury obsługi zdarzenia :

Procedurę TKonfigForm.FormActivate(Sender : TObject);

begin

end;

4.Uzupełnij treść procedury, by miała ona postać taka jak na poniższym wydruku:

procedure TKonfigForm.FormActivate(Sender ; TObject);

begin

KonfigINI := TINIFileCreate('Faktury.ini'); {Otwarcie pliku}

{Wczytywanie serii i dopisku do numeru faktury}

Edit1.Text := KonfigINI.ReadString('Faktury','seria','');

Edit2.Text := KonfigINI.ReadString('Faktury','dopisek','');

Edit3.Text := KonfigINI.ReadString('Nazwa','dane1','');

Edit4.Text := KonfigINI.ReadString('Nazwa','dane2','');

Edit5.Text := KonfigINI.ReadString('Nazwa','dane3','');

Edit6.Text := KonfigINI.ReadString('Nazwa','dane4','');

Edit7.Text := KonfigINI.ReadString('Inne','miejsce_wystaw','');

Pierwsza instrukcja procedury powoduje otwarcie lub utworzenie pliku (jeżeli nie istnieje)/ Następnie instrukcje przyporządkowują właściwościom Text odpowiednich pól TEdit wartości odczytanych z pliku INI. Do odczytu służy metoda ReadString należąca do kalsy TIniFile. Metoda ta przyjmuje trzy parametry: nazwę sekcji, nazwę klucza oraz wartość domyślną, która zostanie użyta w przypadku braku odpowiedniego klucza w pliku. W przedstawionym przykładzie wartość domyślna jest proponowana tylko dla serii faktur. Klasa TIniFile jest zdefiniowana w module IniFiles. Moduł ten nie jest domyślnie dołączany do nowych formularzy, musimy więc zadeklarować jego użycie w klauzuli uses. Ponieważ w naszym przypadku operacja na pliku INI ma charakter lokalny, deklaracja może zostać umieszczona w sekcji implementation klasy TKonfigForm.

5.Umieść kursor na końcu listy klauzuli uses, znajdujący się na początku pliku Konfig.pas i dołącz do niej moduł IniFiles. Musimy jeszcze zdefiniować atrybut KonfigINI. Zmienna ta będzie wykorzystywana przez dwie procedury, lecz nie ma potrzeby jej udostępniania poza modułem Konfig, wystarczy więc, jeżeli zadeklarujemy ją w sekcji private

6.Utwórz wolny wiersz poniżej komentarza {private declarations} i wpisze :

KonfigINI : TIniFile;

7.Kliknij przycisk Save, aby zapisać wprowadzone zmiany

Zapis danych do pliku INI

Chcąc uzyskać w pełni funkcjonalny formularz, musimy zadbać o to ,aby po wprowadzeniu zmian w wartościach konfiguracyjnych zostały one zapisane w pliku INI. Odpowiednie czynności będą wykonywane po kliknięciu przycisku OK opracowywanego formularza

1.Przejdć ponownie do widoku formularza i dwukrotnie kliknij przycisk OK. Dwukrotne kliknięcie kontrolki typu TBitBtn oznacza przejście do edycji zdarzenia OnClick (twórcy Delphi słusznie założyli, że jest to najczęściej wykorzystania zdarzenie)

2.Uzupełnij kod procedury, aby wyglądała ona jak poniżej:

procedure TKonfigForm.OKBtnClick (Sender : TObject);

begin

with KonfigINI do begin

WriteString('Faktury', 'seria', Edit1.Text);

WriteString('Faktury', 'dopisek', Edit2.Text);

WriteString('Nazwa', 'dane1', Edit3.Text);

WriteString('Nazwa', 'dane2', Edit4.Text);

WriteString('Nazwa', 'dane3', Edit5.Text);

WriteString('Nazwa', 'dane4', Edit6.Text);

WriteString('Inne', 'miejsce_wystaw', Edit7.Text);

end; end;

Procedura WriteString przyjmuje trzy parametry : nazwę sekcji i klucza pliku INI oraz zapamiętywaną wartość. Zwróć uwagę na zastosowanie w powyższym kodzie konstrukcji with ... do. Dzięki niej unikamy wielokrotnego wpisywania nazwy klasy, z której pochodzi metoda WriteString (trzeba by pisać KonfigINI.WriteString)

3.Kliknij Save , aby zapisać wprowadzone zmiany

Wywołanie formularza konfiguracyjnego

Moduł Konfig.pas jest już gotowy, pozostało jeszcze napisać odpowiednie instrukcje, które będą wyświetlały formularz. Ponieważ wywołanie formularza KonfigForm będzie następowało z poziomu formularza głównego, musimy do niego powrócić

1.Na pasku narzędzi kliknij prycisk View Form i dwukrotnie kliknij pozycję MenuGlForm

2.Na formularzu kliknij menu Narzędzia i wybierz Polecenia Konfiguracja. Pojawi się powłoka procedury obsługi zdarzenia OnClick dla polecenia menu Konfiguracja1.

3.Uzupełnij tę procedurę , aby miała ona postać:

procedure TMenuGlForm.Konfiguracja1Click (Sender: TObject);

begin

WyswietlKonfiguracje;

end;

Oczywiście w tym miejscu można by od razu umieścić polecenia KonfigForm.ShowModal, ale ponieważ to samo działanie będzie realizować przycisk na pasku narzędzi, dobry styl programowania nakazuje utworzenie oddzielnej procedury. Procedura, której nagłówek nie jest generowany automatycznie, (jak to ma miejsce w przypadku wszystkich procedur obsługi zdarzenia), musi być w całości wpisana przez użytkownika. Należy również pamiętać o umieszczeniu jej deklaracji w definicji klasy

4.Kliknij przycisk Toggle Form/Unit i dwukrotnie kliknij ostatni przycisk paska narzędzi formularza (przedstawiający koło zębaty). Zostanie utworzona powłoka procedury zdarzenia OnClick dla przycisku paska narzędzi. Uzupełnij tę procedurę ,aby wyglądała następująco:

procedure TMenuGlForm.ToolButton6Click(Sender : TObject)

begin

WyswietlKonfiguracje;

end;

5.Ustaw kursor w wolnej linii przed słowem kluczowym end. na końcu modułu MenuGl.pas i wpisz poniższą procedurę:

procedure TMenuGlForm.WyswietlKonfiguracje(Sender : TObject)

begin

KonfigForm.ShowModal;

end;

6.Odszukaj definicję klasy TMenuGlForm i umieść nagłówek procedury przed słowem kluczowym private. Ten fragment kodu będzie wyglądał następująco:

procedure Konfiguracja1Click(Sender : TObject);

procedure WyswietlKonfigurcje(Sender : TObject);

private;

7.Kliknij przycisk Save, aby zapisać wprowadzone zmiany

8.Kliknij przycisk Run ,aby skompilować i uruchomić aplikację. Na ekranie pojawi się komunikat informujący ,że formularz MenuGlForm odwołuje się do formularza KonfigForm, lecz w tym pierwszym barak odpowiedniej deklaracji

9.Kliknij przycisk Yes, aby dołączyć brakujące deklarację

10.Ponownie kliknij przycisk Run

Jeżeli w kodzie nie ma błędów, program zostanie skompilowany i uruchomiony, Po przejściu do formularza z konfiguracją programu w poszczególnych polach, zobaczysz wartości pobrane z pliku INI

Podsumowanie

W ramach tego zadania opracowałeś menu główne oraz pasek narzędzi programu. Przygotowałeś również formularz służący do przeglądania i edycji danych konfiguracyjnych programu. Poznałeś też podstawowe zasady korzystania z plików w formacie INI. Właśnie w takim pliku są przechowywane dane wyświetlane w formularzu konfiguracyjnym. W kolejnej części rozpoczniemy tworzenie formularzy dla danych zgromadzonych w bazie(…)