Paweł Burchert, Adam Pelikant
Wyższa Szkoła Informatyki, Katedra Inżynierskich
Zastosowań Informatyki, 93-008 Łódź, ul Rzgowska 17a
email: pawel.burchert@ruukki.com,
apelikan@wsinf.edu.pl

SYSTEM ZARZĄDZANIA RAPORTAMI

Streszczenie – Artykuł prezentuje metodę administrowania dużym systemem raportowania dla potrzeb polskiego oddziału firmy Ruukki. Prezentowane rozwiązanie oparto na środowiskach MS SQL i VB, dla systemu generowania dynamicznych raportów za pomocą Crystal Reports. Głównym zadaniem oprogramowania jest kontrola uprawnień do generowania raportów oraz monitorowanie (audyt) aktywności poszczególnych operatorów systemu. Aplikacja została wdrożona dla potrzeb dużego spółki Ruukki Polska.

1 Wstęp

Jedną z ważniejszych metod prowadzenia polityki gospodarczej i zarządzania firmą jest analiza danych bieżących i historycznych. Najbardziej podstawowym źródłem danych do analiz jest tworzenie różnych typów raportów. Odpowiada to analizie różnych przekrojów w wielowymiarowych kostkach OLAP. Trudno przypuszczać aby zestaw tworzonych raportów był statyczny i nie podlegał modyfikacji w czasie. Dla małych jednostek gospodarczych można te zadania scedować na informatyka lub ich zespół. Jednak takie rozwiązanie w przypadku dużych firm nie jest do przyjęcia. O wiele bardziej sensowne jest przeniesienie możliwości modyfikacji raportów (choć w ograniczonym zakresie) na poziom wyspecjalizowanych operatorów, związanych z konkretnymi wydziałami przedsiębiorstwa. Takie działanie wymaga jednak opracowania metody tworzenia quasi dynamicznych, modyfikowalnych raportów. Częstym rozwiązaniem tej kwestii jest stosowanie Crystal Reports. Należy podkreślić, że rozwiązanie tu zaproponowane można polecać jako wzorcowe dla wszystkich przedsiębiorstw niezależnie od ich wielkości.

Pomimo tych zalet, problemy pojawiają się w momencie wprowadzania nowego lub modyfikacji istniejącego szablonu, tak aby wszyscy uprawnieni operatorzy mieli aktualny ich zestaw praktycznie w każdej chwili działania. Rozwiązaniem jest zbudowanie centralnej ich składnicy (repozytorium) i redystrybucja na żądanie operatora do stacji roboczych.

Problem ten w praktyce pojawił się w wyniku trudności z zapanowaniem nad istniejącym systemem raportowania w firmie Ruukki Polska, a zbudowanym z zastosowaniem Crystal Reports. Trudności objawiały się koniecznością instalacji każdej nowej wersji raportu na odrębnych stacjach roboczych w momencie utworzenia nowego lub zmiany istniejącego raportu. Często również dochodziło do sytuacji, gdy któraś ze stacji roboczych została pominięta przy ich instalowaniu czy uaktualnianiu.

Kolejnym zadaniem było opracowanie systemu nadawania praw do generowania raportów z podziałem na prawa indywidualne i grupowe. Przy tej okazji rozwiązane zostało również zadanie monitorowania (audytu) aktywności operatorów. Zbierane są informacje związane z typem generowanego raportu i czasem wygenerowania dla każdego z nich. Opisane w artykule rozwiązania są silnie zakotwiczone w praktyce biznesowej dużych konsorcjów przemysłowych.

2 Środowisko programistyczne

Ponieważ infrastruktura informatyczna firmy, w której miał być wdrażany system była oparta o produkty Microsoft, wybór platformy był dość oczywisty. Ponadto istniejące raporty były wykonane w oparciu o Crystal Reports, a dane biznesowe były gromadzone na serwerze MS SQL.

Pomimo tych uwarunkowań możliwe było dokonanie wyboru dowolnej innej platformy. Jednak szereg zalet tych produktów zadecydował o pozostaniu przy narzędziach oferowanych przez te firmę. Pierwszym była duża plastyczność narzędzia raportującego – Crystal Reports w wersji 10 Developer. Wersja 10 została wybrana dlatego, że nie nie było w sprzedaży wersji 9, z którą to wersją VS 2004 współpracuje dość dobrze.

Crystal Reports pozwala na tworzenie dynamicznych raportów (szablonów). Wykorzystywać przy tym można zarówno narzędzia wizualne, jak i wbudowany język skryptowy oraz programowanie w językach wyższego rzędu np. VB. Zawarte w nim środowisko graficzne pozwala stworzyć własny wygląd raportu oraz umożliwia tworzenie wykresów oraz dołączanie ich do raportów źródłowych. Możliwy jest również eksport danych do plików różnego rodzaju[1].

Podobnie wybór serwera bazy danych, wynikający ze stanu zastanego, jest uzasadniony wieloma pozytywnymi cechami tego produktu takimi jak: duża szybkość przetwarzania (najszybszy serwer BD), dobry poziom zabezpieczeń, łatwość utrzymania i administracji, duża niezawodność oraz silne zintegrowanie z mechanizmami wymiany danych. W tej chwili aplikacja współpracuje z MS SQL 7 Enterprise i MS SQL 2000.

[1] Np. MS Excel,Word, Adobe PDF

Największa swoboda wyboru dotyczyła narzędzi programistycznych do tworzenia końcówek klienckich. Wybrana została platforma .NET silnie zintegrowana zarówno z serwerem MS SQL przez platformę dostępu i programowania ADO.NET, jak i z narzędziem raportującym. Ponieważ platforma .NET jest produktem pozwalającym na wykorzystanie wielu języków (programowanie mieszane) pozostał otwarty wybór konkretnego z nich. Wybrany został język VB.NET jako najbardziej podstawowy. Przetworzenie kodu na poziom innego języka np. najbardziej popularny C# nie nastarcza żadnych problemów formalnych lub informatycznych. Wobec możliwości programowania mieszanego (z wykorzystaniem wielu języków jednocześnie) problem wyboru konkretnego z nich jest raczej problemem mniej istotnym, wtórnym.

Ważna z punktu widzenia metody tworzenia aplikacji jest natomiast wybór między cienkim (thin), a grubym (thick) klientem. Aby poprawić wydajność zdecydowano się aby możliwie dużo przetwarzania zostało wykonywane po stronie serwera bazy danych, zmierzając raczej w kierunku pierwszego z rozwiązań.

3 Struktura baz danych

Podstawą funkcjonowania aplikacji jest baza danych stworzona na platformie MS SQL Server 2000. Zbudowany został prosty schemat tabel odwzorowujący wymagania związane z zakresem funkcjonowania aplikacji. Baza Reports zawiera połączone kluczami w schemat relacyjny tabele: Użytkownicy, Grupy, GrupyUzytkownicy, HelpTekst, Raporty, PawaInd, PrawaGrup. Zawarta jest w niej niezwiązana tabela dla potrzeb monitorowania – Historia. Diagram bazy danych przedstawia rys. 1.

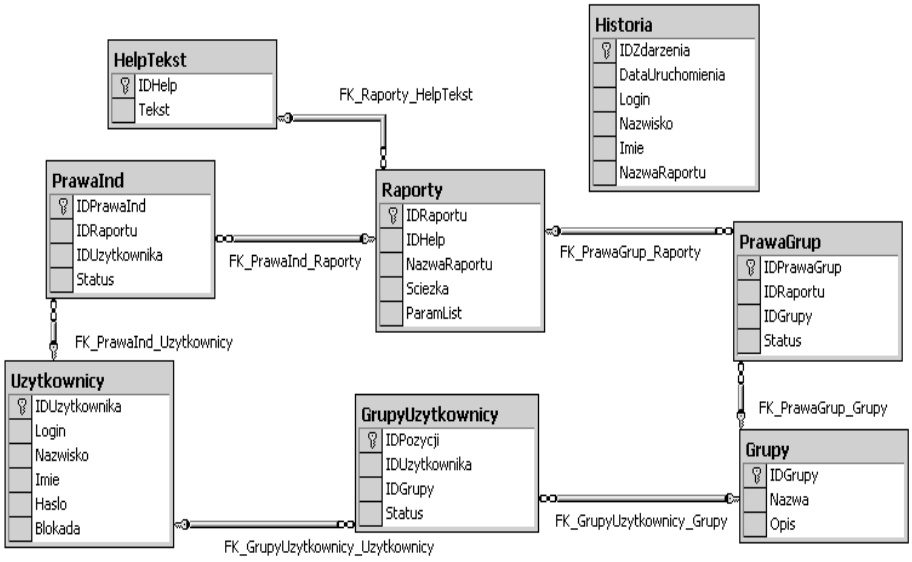
Diagram z rys.1 ilustruje związki jakie zachodzą pomiędzy tabelami oraz więzy integralności utworzone na poziomie tabel. Tabela Historia nie jest związana z żadną z tabel z powodu informacyjnego charakteru przechowywanych w niej danych. Oczywiście można byłoby utworzyć ją w oparciu o klucze obce pozostałych tabel ale założono dodatkowo, że tabela ta może być eksportowana do innej bazy. Aby zapewnić możliwość takiego działania, bez konieczności migracji innych tabel schematu pola tej tabeli zawierają informacje przepisywane z pozostałych tabel.

W tabeli Uzytkownicy należy zwrócić uwagę na pola:

- Hasło przechowujące hasło do celów logowania się w aplikacji jest zawartość jest kodowana algorytmem md5 przy pomocy procedury składowanej serwera bazy danych (KodujHaslo()).
- Blokada zawiera flagę informującą o tym czy nastąpiły próby zalogowania się bez podania prawidłowego hasła. Po trzech zakończonych niepowodzeniem próbach flaga zmienia wartość z 0 na 1 i system uniemożliwia kolejną próbę połączenia się z

aplikacją, o czym system informuje użytkownika. Tylko administrator jest w stanie przywrócić domyślną wartość flagi.

Tabela GrupyUzytkownicy zawiera dane z informace o przynależności użytkownika do grup użytkowników (grupowe przypisywanie praw). Użytkownik może należeć do jednej bądź wielu grup uprawnień (role w aplikacji). Przynależność do grup ma związek z późniejszym wyświetlaniem zawartości menu głównego dla użytkownika. Jedynie administrator ma możliwość przypisania użytkownika do grupy bądź odebrania dostępu do danej grupy (roli).



Rys. 1. Diagram tabel w bazie Reports

Kolejna tabelą jest tabela o nazwie Raporty przechowująca szczegółowe informacje o raportach. Tutaj należy zwrócić uwagę na pole ParamList, które zawiera listę parametrów umożliwiających podłączenie raportu do źródła danych wykorzystywanych w raporcie. Format zawartości tego pola zapisany jest wg następującego klucza:

- Nazwa serwera bazy danych
- Baza danych
- Nazwa użytkownika
- Hasło

Nazwa raportu nie jest unikalna i może w bazie występować wielokrotnie, w zależności od przypisania do grupy bądź użytkownika. Raport może być dostępny dla grupy, ale ponieważ prawo użytkownika jest silniejsze, w związku z tym nie każdy użytkownik może dany raport uru-

chomić. Tabela jest wykorzystywana przez procedurę składowaną **UserReport()** generującą odpowiednie dla użytkownika, a wynikające z jego uprawnień listy raportów.

Tabela HelpTekst zawiera jedynie dwa pola dotyczące opisu raportu. Zawartość pola tekst wyświetlana jest na pasku (StatusBar) okna w którym nastąpiło uruchomienie raportu.

Ostatnią z opisywanych tabel jest tabela Historia. Przeznaczeniem tabeli jest przechowywanie informacji o raportach uruchamianych przez użytkownika. Jest ona wykorzystywana do monitorowania aktywności operatorów systemu.

4 Procedury składowane

Oprócz tabel aplikacja korzysta także z procedur składowanych. Jest to znacznie wygodniejszy sposób na dostęp do danych określonego typu niż tworzenie teksu procedury w kodzie programu.

Oparcie aplikacji o procedury składowane SQL miało również na celu łatwiejszą, poźniejszą rozbudowę aplikacji o nowe komponenty, bądź zmianę funkcjonalności jej fragmentów.

Procedury składowane służą do: kodowania hasła z zastosowaniem algorytmu MD5, zwracania danych logowania z aplikacji do bazy, sprawdzania zakresu uprawnień i wyświetlania odpowiedniej listy dostępnych raportów etc.

Warto tu wspomnieć o samym szyfrowaniu hasła. Została użyta tutaj gotowa metoda ComputeHash() zawarta w przestrzeni nazw System.Security.Cryptography. Metoda ta wylicza znacznik hash dla łańcucha znakowego będącego parametrem metody KodujHaslo(). Ponieważ znacznik ten jest typu binary, podczas zwracania wyniku jest konwertowany do typu string i zapisywany w odpowiednim polu bazy danych. Wynikiem jest oczywiście zakodowane hasło. Rozmiar znacznika wynosi 160 bity i raczej jest dość trudny do odszyfrowania.

5 Usługa Windows

Dodatkowym elementem wspomagającym pracę administratora systemu zarządzania raportami jest usługa pracująca na serwerze, gdzie umieszczony jest katalog z raportami. Odpowiedzialna jest za nadzorowanie katalogu (wraz z podkatalogami) zawierającego szablony raportów. W momencie pojawienia się w którymkolwiek podkatalogu nowego raportu, serwis ten dopisze do tabeli Raporty nowy rekord zawierający informację o nazwie nowego obiektu oraz o ścieżce dostępu. Administratorowi pozostaje jedynie przypisanie go do odpowiedniej grupy lub użytkowników. Kod usługi przedstawiono na listingu 1.

Listing 1. Fragment kodu usługi nadzorującej katalog zawierający raporty.

```
1:     Private Sub DirectoryChanged(ByVal sender As Object, _
2:             ByVal e As System.IO.FileSystemEventArgs) _
3:             Handles DirectoryWatcher.Changed, _
4:     DirectoryWatcher.Created, DirectoryWatcher.Deleted
5:     'Pełna ścieżka do modyfikowanego pliku
6:             ReportPath = e.FullPath
7:           'Nazwa raportu
8:         ReportName = Right(e.FullPath, Len(e.FullPath) –
9:                               InStrRev(e.FullPath, "\"))
10:    ReportName = Left(ReportName, InStr(ReportName, ".") - 1)
11:    ConReports.Open()
12:    'Dopisanie do bazy danych informacji na temat raportu
13:    Try
14:    AppReport = "INSERT INTO
    Raporty(NazwaRaportu,Sciezka,IDHelp) VALUES(
15:          "'" & ReportName & "','" & ReportPath & "'," & 1 &
    ")"
16:    Catch
17:            MsgBox(Err.Number, MsgBoxStyle.Information)
18:    End Try
19:    Dim objCmd As New SqlCommand(AppReport, ConReports)
20:            objCmd.ExecuteReader()
21:            ConReports.Close()
22:    End Sub
23:    Sub CheckIt()
24:        DirectoryWatcher = New  _
25:        System.IO.FileSystemWatcher("D:\Raporty", "*.rpt")
26:        DirectoryWatcher.IncludeSubdirectories() = True
27:        DirectoryWatcher.EnableRaisingEvents = True
28:    End Sub
```

Usługa zapisuje również informacje o starcie i zatrzymaniu do oddzielnego logu stworzonego specjalnie dla niej – listing 2.

Listing 2. Fragment kodu – zapis do logu startu i stopu usług

```
1: Protected Overrides Sub OnStart(ByVal args() As String)
2: Dim m_watch As New FileWatch
3: m_watch.CheckIt()
4: EventLog1.WriteEntry(Err.Number)
5: EventLog1.WriteEntry("Reports started")
6: End Sub
7: Protected Overrides Sub OnStop()
8: EventLog1.WriteEntry("Reports stopped")
9: End Sub
```

Katalogi zawierające raporty umieszczone są na serwerze aplikacji i odpowiadają strukturze wydziałów. W każdym z nich znajdują się odpowiednio umieszczone raporty w formacie Crystal Reports. Jakakolwiek

zmiany istniejących raportów bądź dodanie nowych dokonywane jest w tej lokalizacji, a na skutek działania usług Windows (opisanych poprzednio) są natychmiast zauważane przez wszystkich uprawnionych użytkowników systemu.

6 Struktura aplikacji

Opracowana aplikacja jest aplikacją typu MDI, czyli zawiera jeden dokument główny, który stanowi kontener dla każdego kolejnego otwieranego dokumentu.

Po uruchomieniu aplikacji pierwszym oknem jakie widzi użytkownik, jest okno logowania. W widocznych polach należy odpowiednio wpisać login Użytkownika oraz hasło logowania. Długość loginu przyjęto jako cztery znaki, zaś długość pola hasła nie ma ograniczeń. W chwili wpisywania identyfikatora użytkownika, litery zamieniane są na wielkie, zaś hasło zasłaniane jest symbolem *. Użytkownik ma jedynie trzy próby logowania. Po trzeciej nieudanej konto jest blokowane. Blokowanie polega na ustawieniu właściwej flagi w bazie Uzytkownicy. Flaga badana jest podczas logowania i po stwierdzeniu blokady, niemożliwe jest zalogowanie na tak oznaczone konto. Konto administratora nie podlega tego typu ograniczeniom. Dane użytkownika logującego się zwracane są poprzez procedurę składowaną umieszczoną w bazie Reports.

Pierwsze logowanie nowego użytkownika wymusza wprowadzenie hasła. Hasła są szyfrowane algorytmem MD5 i w postaci zaszyfrowanej wpisane do bazy. Podczas pracy aplikacji użytkownik ma możliwość zmiany hasła.

Nazwy pozycji menu pobierane są z bazy. Ich nazwy odzwierciedlają przynależność użytkownika do wydziału, np. Księgowość, Produkcja itp. Użytkownik może należeć do różnych wydziałów (w sensie raportów). W każdej grupie dostępne są raporty, przypisane do odpowiednich użytkowników, co w praktyce oznacza, że w danej grupie ten sam raport nie musi być dostępny dla każdego członka tej grupy.

Ze względu na liczbę licencji dostępowych (CAL) do serwera SQL, każde otwarcie nowego raportu spowoduje zamknięcie uprzednio otwartego. Uruchomienie raportu jest odnotowywane w bazie historycznej, zawierającej datę uruchomienia, nazwę użytkownika oraz nazwę raportu.

Dodatkowo administrator posiada własne okna i menu do zarządzania zarówno użytkownikami, jak i grupami uprawnień (rolami) oraz raportami.

Wygląd aplikacji może wydawać się dość surowy, jednakże jest dość użyteczny i nie rozprasza użytkowników. Poza tym najwięcej miejsca przeznaczono na sam raport a nie na elementy nawigacyjne (niezbędne dostępne są na ToolBarze) czy zbędne ozdobniki.

7 Dystrybucja aplikacji

Aby stworzyć wersję instalacyjną, został do istniejącej aplikacji dobudowany projekt instalatora o nazwie Reports. Aby jednak mogły być prawidłowo uruchamiane raporty, należało w oknie Solution Explorer wybrać opcję Add->Merge Module

i wybrać CrystalReports10_NET_EmbeddedReporting.msm. VB.NET w momencie kompilacji poprosi o klucz rejestracyjny Crystal Reports, który jest dostarczany wraz z licencją. Brak klucza uniemożliwia stworzenie wersji instalacyjnej. Następni dodano skrótu do programu, który pojawi się na pulpicie użytkownika oraz menu Programs.

Można tworzyć w instalatorze skróty do programu, które pojawią się po zainstalowaniu na pulpicie komputera oraz nazwę katalogu intalacyjnego. VB.NET domyślnie podpowiada jako nazwy katalogów, nazwę firmy oraz nazwę programu umieszczane katalogu C:\Program Files. Taką też konwencję przyjęto również podczas tworzenia oprogramowania.

Zmienione zostały również właściwości projektu instalatora. Wszystkie pliki potrzebne aplikacji do pracy zostały umieszczone w pliku Raporty.msi – pliku instalatora po wybraniu opcji Package Fles -> In setup file. Możliwy jest też podział na pliki *.cab.

Opcja Bootstrapper oznacza, że można stworzyć instalatora aplikacji Windows lub Web. Wartość pola Compression została ustawiona na Optimized For Speed. Można wybrać także opcję Optimized for size, ale ze względu na dostępność pojemnych nosników danych, można ją pominąć.

Ostatnim krokiem jest kompilacja i budowa wersji instalacyjnej. Końcowym efektem działania tej ocji jest stworzenie wszystkich plików instalacyjnych aplikacji Raports na dysku. Powstają też pliki do dystrybucji .NET Framework 1.1[2.] Pliki, te można przegrać na CD i instalować na docelowaej maszynie. Nie ma obawy (jak to było w poprzednich wersjach Visual Basic) niezarejstrowania bibliotek DLL, nadpisania poprzednich itp. kłopotów. Zjawisko to nosiło potoczną bazwę „Piekła DLLi”. W obecnej postaci FrameWork rozwiązuje wszytkie te bolączki automatycznie.

Sam proces instalacji przebiega na ogół bez zarzutu. Instalację na stacji roboczej z systemem W2K lub Win XP może przeprowadzić jedynie użytkownik z prawami administratora. W momencie wystąpienia kłopotów z zainstalowaniem programu, program instalatora potrafi wycofać cały proces do stanu z przed nieudanej próby instalacji.

[2] Firma Microsoft nie zaleca używania Framework w wersji 1.0.

8 Podsumowanie

W chwili obecnej aplikacja pracuje w firmie Ruukki Polska i jest nadal rozwijana. Została wzbogacona o nowe elementy wspomagające pracę dodatkowej grupy użytkowników. W chwili obecnej w systemie zarejestrowanych jest ponad 30 operatorów, wykorzystujących po kilkadziesiąt różnych raportów.

Rozważana jest możliwość przeniesienia aplikacji całkowicie na jeden z serwerów i rezygnacja z dystrybucji na stacjach roboczych.

Kolejnym etapem będzie rozbudowa modułu kontrolującego użytkowników oraz przeniesienie kodu na platformę C#.

Do dziś większość raportów została zapisana, po uprzednich testach, w wersji CR 10 z SP4. Zapisanie w uaktualnionej wersji wynikało z pewnych błędów Crystala, z którym jednym dość istotnym były problemy z pobieraniem danych z baz w oparciu o procedury składowane oraz przekazywanie do nich parametrów zewnętrznych bądź dość nieczytelne eksporty do Excela.

Ciekawym modułem, zaiplementowanym w Raportach jest obsługa bazy danych zużytego surowca w oparciu o numer parti. Moduł wspiera pracę osoby zajmującej się kontrolą rozliczania produkcji. Osoba ta wprowadza do bazy informacje o zużytym surowcu na podstawie raportów z Działu Produkcji. W momencie wprowadzania numeru partii, moduł łączy się z bazami danych systemu FourthShift i sprawdza, czy rzeczywiście w bazach surowiec został całkowicie zużyty. Jeśli tak jest, indeks zapisywany jest do tabeli zużytego surowca. Tabela ta z kolei jest wykorzystywana przez raport produkcyjny. Jeśli jednak w FourthShifcie dana partia figuruje jako nie zużyta do końca, moduł informuje użytkownika o tym fakcie i nie pozwala na dopisanie takich danych do tabeli. Informacja wyświetlana informuje również ile danego materiału pozostało w systemie.

Oprócz powyższej weryfikacji następuje również sprawdzenie, czy w tabeli nie istnieje już rekord z identycznym numerem partii (numery są unikalne). W tym przypadku operator także jest informowany o próbie dopisania istniejąceych danych.

Aplikacja ta wzbudziła duże zainteresowanie firmy audytorskiej Ernst & Young. Audytorzy przyznali, że spotkali się pierwszy raz z tak scentralizowanym i ściśle kontrolowanym środowiskiem raportującym, bazującym na zróżnicowanych systemach i bazach danych.

Literatura

[1] Celko J.: SQL – Zaawansowane Techniki Programowania,MIKOM, Warszawa 1999 r.

[2] Codde E. F.: The relational model for database management, MA: Addison –Wesley 1990 r.

[3] Date C. J.: Relational database: writings 1989-1991, MA: Addison –Wesley 1992 r.

[4] Esposito D.: Tworzenie aplikacji za pomocą ASP.NET I ADO.NET, RM, Warszawa 2002 r.

[5] Halvorson M.: Microsoft Visual Basic .NET, RM, Warszawa 200

REPORT MANAGEMENT SYSTEM

Summary – The paper presents method of administering large report management system realized for Polish division of Ruukki concern. This solution based on MS SQL Server and VB platforms. Dynamic reports generation is obtained due to Crystal Reports technology. Main task of this application is authorization control for reports generation and monitoring (audit) individual system operator activity. Application has been accustomed according to requirements of big company Ruukki Poland.