System zarządzania firmą – specyfikacja techniczna.

1. Zakres funkcjonalności

Funkcjonalność aplikacji została podzielona na 3 grupy:

- Zbiór podstawowych danych – dane kontrahentów, typy dokumentów, magazynów, cenniki, informacje o produktach, surowcach, oraz informacje technologiczne.
- Prowadzenie dokumentów: magazynowych, faktur, zamówień.
- Raporty automatyczne – czasy potrzebne na wyprodukowanie danego produktu, plany produkcyjne i plany zamówień, oraz stany magazynowe.

Istotą programu jest jak najlepsze wykorzystanie informacji zawartych w podstawowych danych, by ułatwiać pracę w jego bardziej rozbudowanych elementach. Wprowadzone dane podstawowe wyświetlane jako listy wspomagają wybór właściwych pozycji, lub podpowiadają prawidłowe wartości. Przykładem mogą być ceny produktów uzupełniane podczas tworzenia dokumentu, na podstawie informacji zawartych w cennikach, a także automatycznie doliczana stawka VAT przypisana do typu dokumentu. Produkty, surowce i usługi zebrane są tym samym miejscu ponieważ w założonej strukturze dostarczają podobnych informacji. Rozróżniane są dzięki parametrowi kategorii, oraz zawierają dane o czasie potrzebnym na zamówienie surowca, lub wykonanie usługi. Technologiczny skład produktów określa ilość i rodzaj surowców i usług potrzebnych do ich przygotowania. Grupa raportów generowanych automatycznie, pozwala na obsługę zagadnień w nich poruszanych, bez konieczności dodatkowej pracy ludzi do ich obsługi. Skład produktów pozwala na obliczanie czasów potrzebnych do przygotowania gotowego produktu. Dzięki znajomości czasów i terminów realizacji, wprowadzanych w zamówieniach możliwe jest automatyczne uzyskanie list surowców koniecznych do zamówienia w danym tygodniu, oraz planów produkcyjnych na wybrany tydzień. Faktury i dokumenty magazynowe zawierają informację o tym skąd został pobrany i gdzie trafił konkretny produkt. Dzięki takiej organizacji możliwe jest przeglądanie stanów magazynowych w wybranym dniu.

Aplikacja została zbudowana w dwóch etapach:

- Zaprojektowanie w pełni funkcjonalnej, relacyjnej bazy danych, odwzorowującej założony model przedsiębiorstwa.
- Opracowanie i napisanie interfejsu dla użytkowników pozwalającego na wygodne, szybkie i bezproblemowe wykorzystanie stworzonej bazy. Pozwala na graficzne zarządzanie bazą, oraz budowanie poleceń SQL za pomocą kliknięć myszy Jest to aplikacja napisana bezpośrednio pod funkcjonalność stworzonej bazy.

2. Budowa bazy danych

Baza danych zbudowana została w wydajnym i zgodnym ze standardami systemie zarządzania bazami danych Firebird. W bazie zaprojektowane zostały tabele połączone kluczami, oraz definiujące rodzaj wprowadzanych danych. Wszystkie tabele posiadają sztuczny auto-inkrementowany klucz główny. Upraszcza to aktualizację bazy danych zmianami wprowadzonymi w aplikacji. Baza udostępnia aplikacji dostęp do danych przez obsługę odpowiednich zapytań SQL, przez nią wysyłanych. Tabele w bazie zostały stworzone do gromadzenia danych:

- Kontrahentów – zbiór dostępnych i istotnych informacji, łącznie z terminami płatności,
- Produktów, surowców, oraz usług – podstawowe informacje, wraz z czasami,
- Skład Produktów – informacje technologiczne,
- Cennik produktów – zbiór cen wraz z datami ich obowiązywania, dzięki czemu podpowiedzi cen w dokumentach powinny być właściwe,
- Lista Typów magazynów – symbol i nazwa dostępnych magazynów,
- Lista Typów dokumentów – faktury, oraz dokumenty magazynowe, wraz z domyślnymi kierunkami przesunięć magazynowych i stawką VAT,
- Nagłówki dokumentów – typ dokumentu, symbol, dane kontrahentów, daty. Nie zawiera pozycji, ponieważ przy większej ilości niż jedna wprowadzałoby to nadmiarowość danych,
- Pozycje faktur – wartość klucza nagłówka dokumentu, cena sztuki i VAT są podpowiadane, wartość netto i brutto wyliczane są ze stawki VAT, ilości, oraz rabatu,

- Zamówienia klientów– dzięki żądanemu terminowi realizacji, umożliwiają automatyczne planowanie produkcji i listę koniecznych do zamówienia surowców.

3. Aplikacja okienkowa

Interfejs użytkownika napisany został z wykorzystaniem bibliotek Windows Forms. Jest on graficznym odwzorowaniem zdefiniowanej bazy danych, rozszerzającym jej funkcjonalność i ułatwiającym pracę na niej. Do łączenia środowiska .NET z bazą danych, Firebird udostępnia własną bibliotekę: FirebirdSql.Data.FirebirdClient.dll, która musi być dołączona do aplikacji przez referencje. Biblioteka ta została napisana w języku C#, udostępnia zestaw klas, z których w programie wykorzystane zostały:

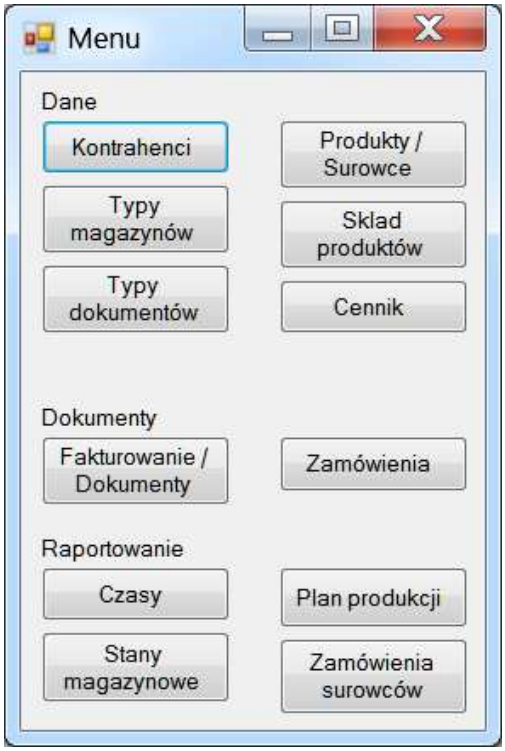
- FbConnection – reprezentuje połączenie z bazą danych,
- FbCommand – służy do przechowywania poleceń SQL do wykonania w bazie,
- FbDataAdapter – służy do pobrania danych z bazy danych do obiektu DataSet, oraz do zapisania w bazie zmienionych danych w DataSet,
- FbCommandBuilder – służy do automatycznego wygenerowania poleceń SQL do aktualizacji bazy danych po zmianach wprowadzonych podczas działania aplikacji.

Aplikacja działa w modelu bezpołączeniowym, więc do wymiany danych z bazą wykorzystywane są metody z klasy FbDataAdapter otwierające połączenie tylko na czas wymiany danych:

- FillSchema – służy do załadowania tabeli wraz z wszystkimi dostępnymi metadanymi,
- Fill – importuje tylko dane, a dzięki właściwości MissingSchemaAction ustawionej na wartość AddWithKey, importuje tabelę wraz z kluczami podstawowymi,
- Update – zapisuje w bazie danych zmiany, jakie zostały wprowadzone w obiekcie DataSet.

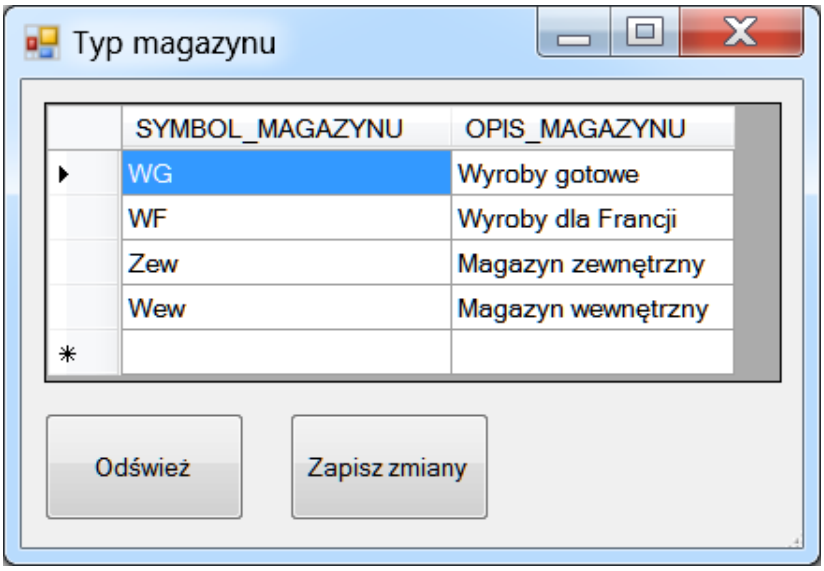
4. Rodzaje formularzy

Wszystkie funkcjonalności aplikacji zostały pogrupowane w okienku menu w taki sposób by korzystanie z nich było wygodne i łatwo dostępne.

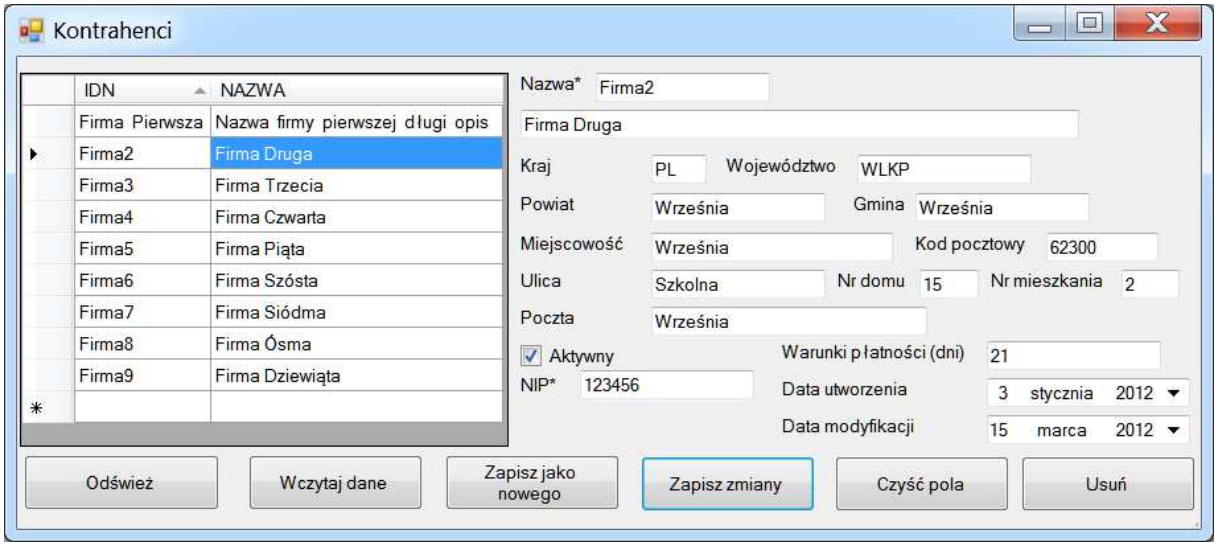


Rys. 8.1. Okienko formularza „Menu”

Formularze wybierane z menu głównego mają za zadanie komunikację z odpowiednimi fragmentami bazy danych. Wszystkie korzystają z funkcjonalności kontrolki DataGridView służącej do wyświetlania danych w formie tabelarycznej. Może ona służyć jako bezpośrednie narzędzie do zmiany danych w obiekcie DataSet, jak np. w okienku zawierającym typy magazynów, lub jedynie jako lista pozycji wyświetlanych i edytowanych w osobnych polach formularza np. okienko kontrahenci.

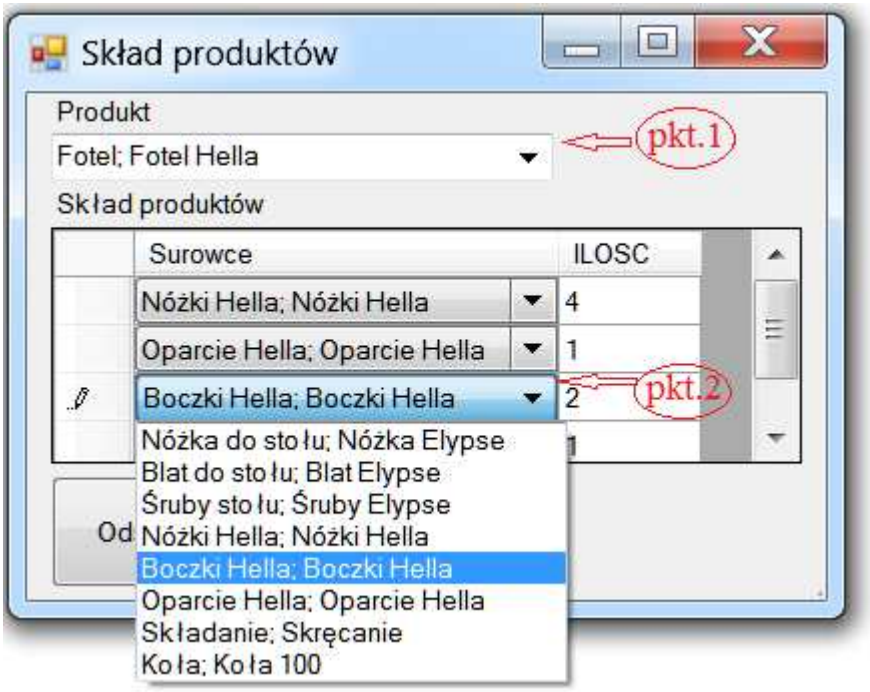


Rys. 8.2. Okienko formularza „Typ magazynu”



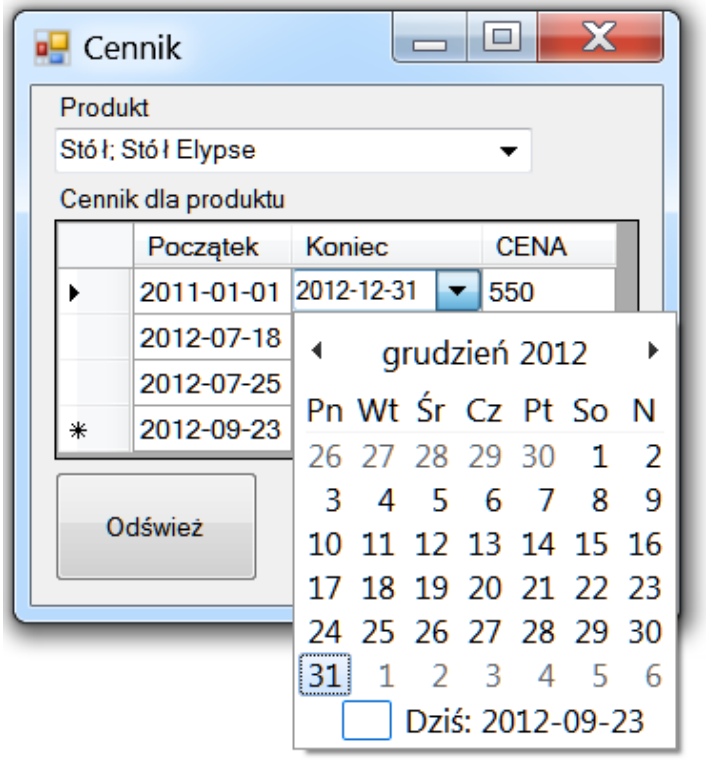
Rys. 8.3. Okienko formularza „Kontrahenci”

Kontrolki ComboBox pozwalają na wybór parametru z listy dostępnych możliwości. Taka kontrolka jest najlepszym rozwiązaniem podczas uzupełniania pola klucza obcego, zapobiegająca wpisaniu wartości spoza zakresu (rys pkt.2), oraz jako element będący filtrem wyświetlanych danych (rys pkt.1).



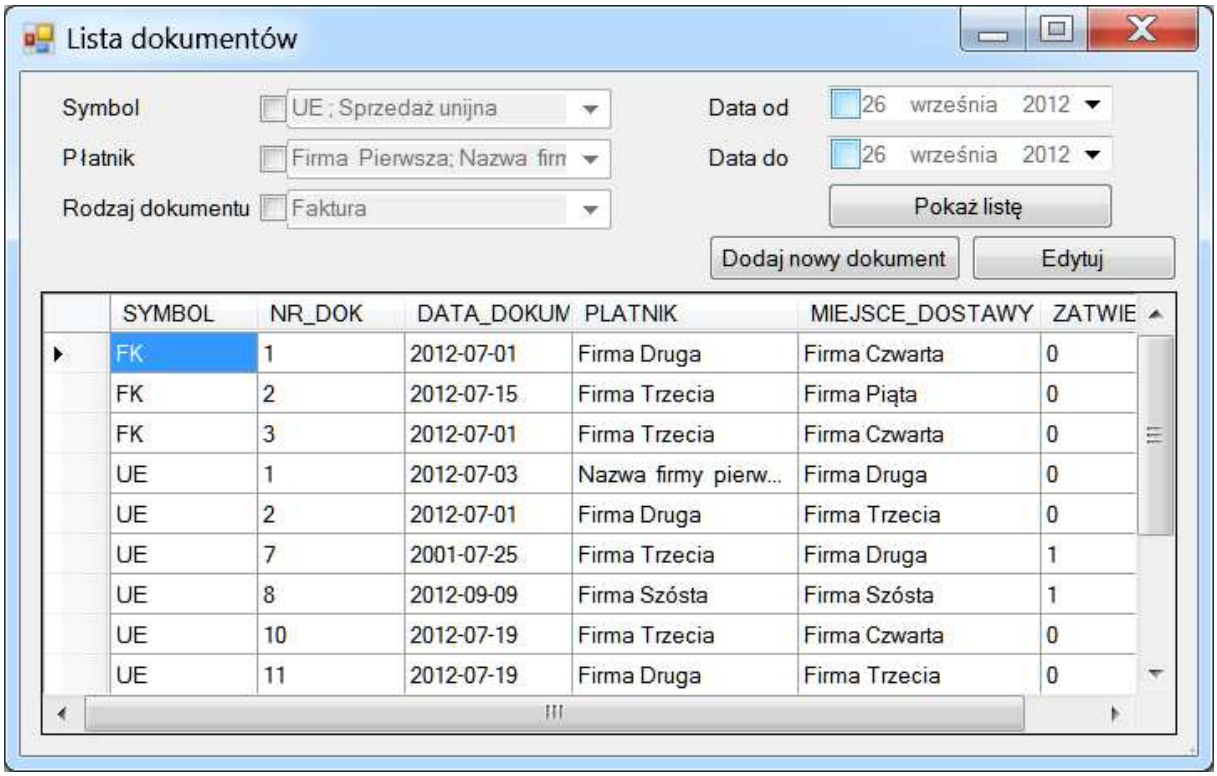
Rys. 8.4. Okienko formularza „Skład produktów”

W formularzu z cenami pojawia się rozszerzenie funkcjonalności pola DataGridView, dające możliwość wybór daty z rozsuwanego kalendarza. Podstawowy zestaw bibliotek nie udostępnia takiej funkcji, ale została ona pobrana ze strony Microsoftu i zamieszczona w programie jako dodatkowa klasa. Pozwala to zaobserwować jak łatwo można rozbudowywać aplikacje o dowolne możliwości tworząc własne klasy.



Rys. 8.5. Okienko formularza „Cennik”

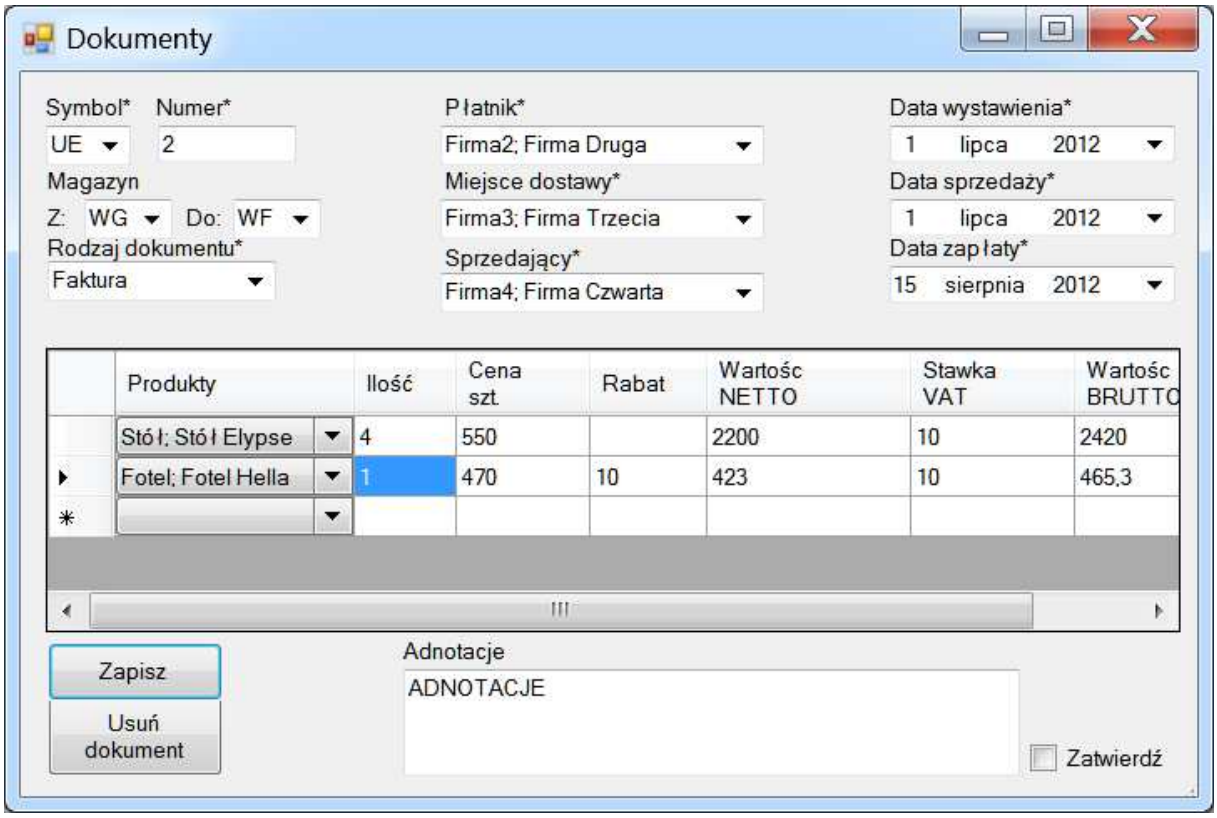
Korzystanie z kontrolek podczas wyboru filtrów ma wpływ na postać polecenia SQL, pobierającego do formularza dane w formie tabeli. Wybór odpowiednich parametrów, oraz ich wartości w środowisku graficznym, pozwala na budowanie wielu opcji składni zapytania. W aplikacji funkcjonalność ta ogranicza się jedynie do zakresu potrzeb danego formularza, ale daje to możliwość dowolnej jej rozbudowy wraz z rozbudową bazy danych. W napisanej aplikacji najwięcej kontrolek modyfikujących polecenie SQL znajduje się w formularzu z listą dokumentów.



SYMBOL	NR_DOK	DATA_DOKUM	PLATNIK	MIEJSCE_DOSTAWY	ZATWIE
FK	1	2012-07-01	Firma Druga	Firma Czwarta	0
FK	2	2012-07-15	Firma Trzecia	Firma Piąta	0
FK	3	2012-07-01	Firma Trzecia	Firma Czwarta	0
UE	1	2012-07-03	Nazwa firmy pierw...	Firma Druga	0
UE	2	2012-07-01	Firma Druga	Firma Trzecia	0
UE	7	2001-07-25	Firma Trzecia	Firma Druga	1
UE	8	2012-09-09	Firma Szósta	Firma Szósta	1
UE	10	2012-07-19	Firma Trzecia	Firma Czwarta	0
UE	11	2012-07-19	Firma Druga	Firma Trzecia	0

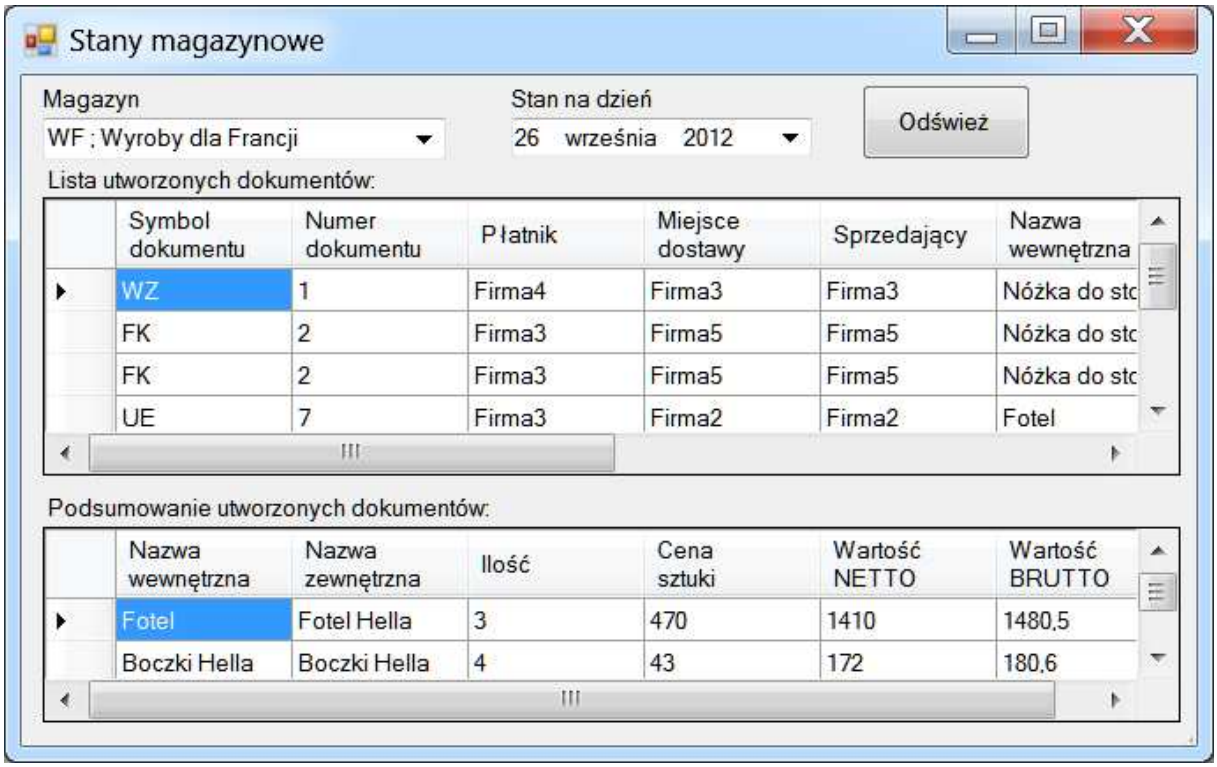
Rys. 8.6. Okienko formularza „Lista dokumentów”

Po dokonaniu wyboru dokumentu z listy, do formularza edycji dokumentu zostają załadowane dane nagłówka i pozycji, lub otwarty zostaje jako pusty w przypadku wyboru nowego dokumentu.



Rys. 8.7. Okienko formularza „Edycja dokumentów”

Grupa formularzy zebranych jako raporty, wykorzystuje kontrolki DataGridView, nie udostępniając możliwości ich edycji. Aplikacja wysyła do bazy polecenie SQL zbierające informacje z kilku tabel, wraz parametrami wybranymi w formularzach i w odpowiedzi dostaje dane tylko do odczytu w postaci tabeli. Przykładem może być formularz informujący o stanach magazynowych, wyświetlający listę zebranych danych, oraz ich podsumowanie.

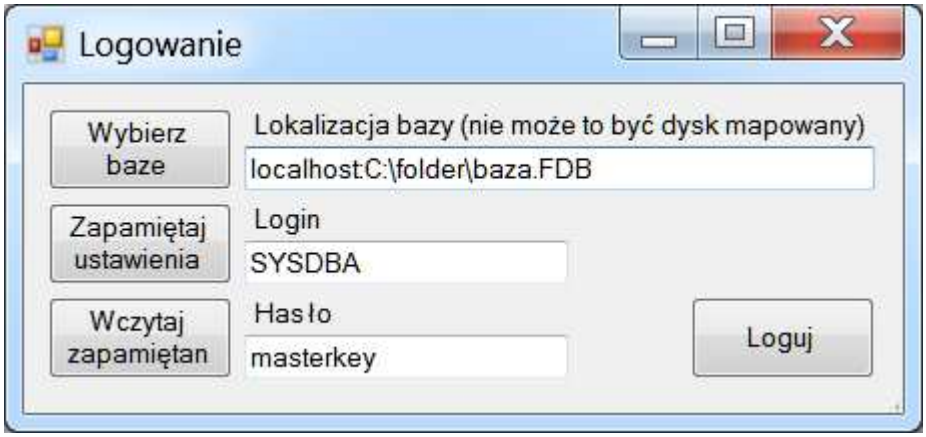


Rys. 8.8. Okienko formularza „Stany magazynowe”

5. Uwagi

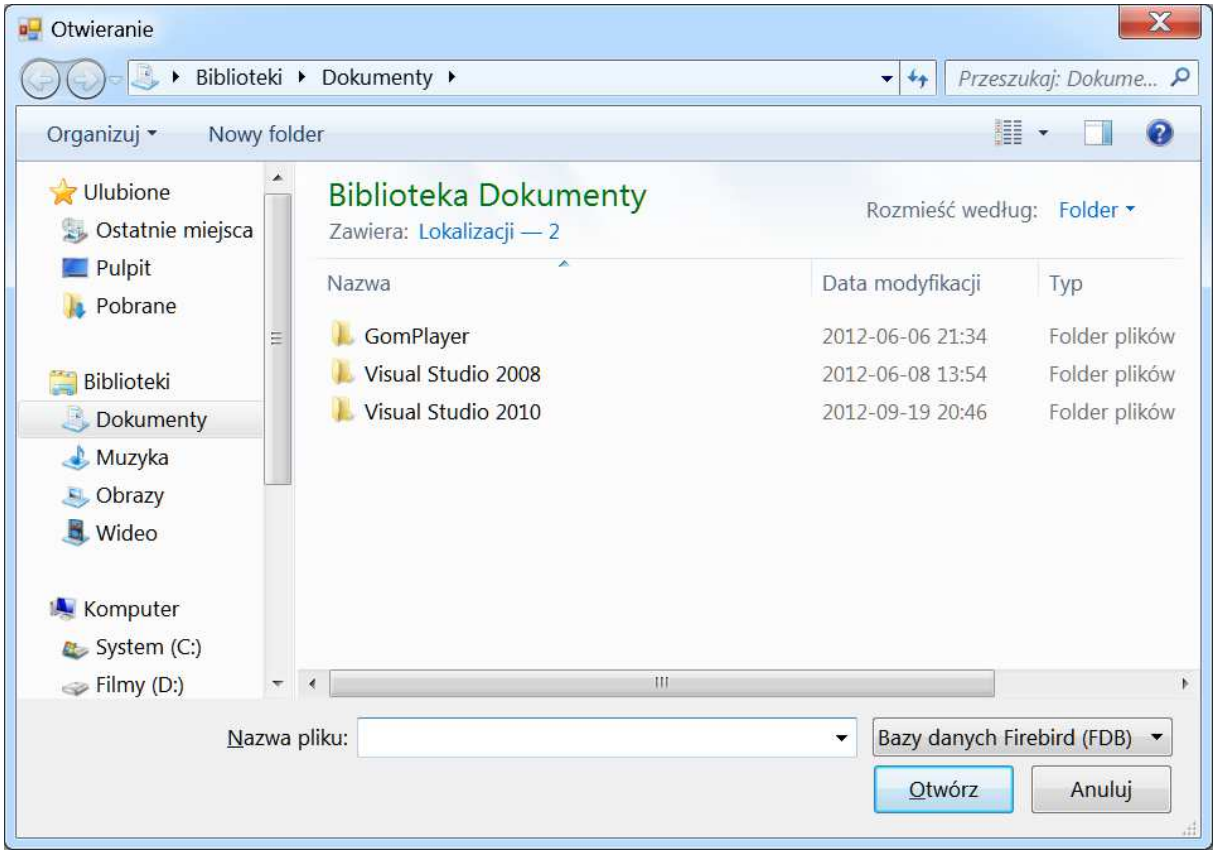
5.1. Logowanie do bazy

Aplikacja komunikuje się z bazą danych, niezależnie od tego gdzie plik z bazą fizycznie się znajduje. Wystarczy, że informacja ta wraz z loginem i hasłem podana zostanie do obiektu klasy FbConnection. Każdy z formularzy wymienia dane z bazą, więc powinien znać jej lokalizację. W tym celu powstało okienko logowania zapisujące uzyskane w nim informacje jako dostępne w całej aplikacji.



Rys. 8.9. Okienko formularza „Logowanie”

W odszukaniu bazy pomaga obiekt klasy OpenFileDialog pozwalający przeszukiwanie zasobów przez okno z filtrem na pliki z rozszerzeniem *.FDB.



Rys. 8.10. Okienko otwierania

Wartość zwracana jako lokalizacja pliku musi zostać zmodyfikowana, tak by była zgodna z wymaganą strukturą przez Firebird. [12] Okno logowania zmieni ciąg znaków informujący o lokalizacji na dysku lokalnym z np. „C:\folder\baza.FDB” na „localhost:C:\folder\baza.FDB”, oraz informujący o lokalizacji na dysku sieciowym z np. „\\192.168.1.5\folder\baza.FDB” na „192.168.1.5:\folder\baza.FDB”. Firebird nie zezwala na lokalizację bazy na dyskach mapowanych. Dla wygody użytkownika udostępniona została również możliwość zapisu do pliku ustawień logowania i wczytywanie ich automatycznie przy uruchamianiu programu.

5.2. Aktualność danych

Model bezpołączeniowy wiąże się z możliwością niezależnego wprowadzenia zmian przez użytkowników i brakiem wzajemnych informacji o tym, do czasu aktualizacji obiektu DataSet. Nowym wierszom powstałym w aplikacji, zostają przyznane nowe numery kluczy głównych, jednak nie oznacza to, że jest to numer jaki otrzyma dany wiersz w bazie danych. Wprowadzanie zmian dla niewłaściwego klucza głównego spowodowałoby błędy informacji, dlatego po zapisie nowego wiersza w aplikacji jest on zapisywany w bazie, a następnie obiekt DataSet jest uaktualniany.

W przypadku zmian wprowadzanych do tych samych wierszy danych, aplikacja zakłada, że osoba dokonująca zmian jako ostatnia ma najaktualniejsze dane. Z tego powodu została zastosowana metoda „ostatni wygrywa”, jednak praca ma na celu zaprezentowanie tylko przykładowych możliwych rozwiązań.