

POLITECHNIKA ŚLĄSKA W GLIWICACH
WYDZIAŁ AUTOMATYKI, ELEKTRONIKI I INFORMATYKI



INSTYTUT INFORMATYKI

PRACA DYPLOMOWA MAGISTERSKA

System mobilnej sprzedaży

Promotor:

dr inż. Katarzyna Haręźlak

Autor:

Tomasz Subik

Gliwice 2009

Spis treści

1. Wstęp	6
2. Analiza	8
2.1 Specyfika pracy działu handlowego	8
2.2 Rozwiązanie problemu	9
2.3 Wybrane rozwiązania konkurencyjne	9
2.4 Cel pracy	10
3. Opis funkcjonalności systemu	11
3.1 Aplikacja mobilna	11
3.2 Aplikacja webowa	11
3.2.1 Przedstawiciel handlowy	11
3.2.2 Przedstawiciel regionalny	12
3.2.3 Dyrektor sprzedaży	12
3.3 Diagram przypadków użycia	12
4. Projekt systemu	14
4.1 Architektura rozwiązania	14
4.1.1 Architektura aplikacji	15
4.2 Projekt bazy danych	16
4.2.1 Autoryzacja oraz uwierzytelnianie użytkowników systemu	16
4.2.2 Klienci	17
4.2.3 Dane adresowe	17
4.2.4 Pracownik i jego zadania	18
4.2.5 Przechowywanie produktów	19
4.2.6 Zarządzanie zamówieniami	19
4.2.7 Przechowywanie danych słownikowych	20
5. Narzędzia	22
5.1 Wybór platformy	22
5.2 Język programowania	24
5.3 Środowisko programistyczne	25
5.4 Serwer bazy danych	26
5.5 Mobilny serwer bazy danych	27
5.6 ORM	27
6. Specyfikacja zewnętrzna	29
6.1 Aplikacja mobilna – Pocket PC	29

6.1.1 Logowanie – pierwsza synchronizacja	29
6.1.2 Ekran główny	30
6.1.3 Klienci	31
6.1.4 Produkty	33
6.1.5 Zadania	34
6.1.6 Tworzenie zamówienia	34
6.1.7 Konto osobiste	36
6.2 Aplikacja WWW	36
6.2.1 Logowanie	37
6.2.2 Ekran główny	37
6.2.3 Menu główne	39
6.2.4 Pulpity	40
6.2.5 Dodawanie, edycja oraz przeglądanie danych	42
6.2.6 Tryb tworzenia zamówienia	44
7. Specyfikacja wewnętrzna	51
7.1 Technologia Microsoft ASP .NET AJAX	51
7.2 Komponenty użytkownika	53
7.3 Obsługa zakładek w widoku szczegółów – aplikacja mobilna	56
7.4 Logowanie przy użyciu usługi sieciowej	58
7.5 Implementacja warstwy dostępu do danych	58
7.5 Generowanie raportów – Reporting Services	65
7.6 Obsługa błędów – log zdarzeń	68
7.7 Walidacja danych	69
7.8 Wysyłanie wiadomości email	70
7.9 Synchronizacja danych	71
8. Bezpieczeństwo	76
8.1 Bezpieczeństwo aplikacji mobilnych	77
8.1.1 Aspekty związane w wyświetlaniem oraz wprowadzaniem danych	77
8.1.2 Local Authentication SubSystem (LASS)	78
8.1.3 Zastosowanie metod kryptograficznych	78
8.1.4 Podpisywanie kodu	79
8.1.5 Bezpieczeństwo danych w aspekcie aplikacji mobilnych – bezpieczna replikacja	79
8.2 Bezpieczeństwo aplikacji internetowych ASP .NET	80
8.2.1 Mechanizmy MembershipProvider oraz RoleProvider	80

8.3 Bezpieczeństwo głównej bazy danych	83
8.4 Zabezpieczenie przed awariami	83
8.5 Podsumowanie	84
9. Testowanie	85
9.1 WebAii – testowanie aplikacji ASP .NET	85
9.2 Testowanie replikacji	88
9.3 Pozostałe testy	89
10. Podsumowanie	91
Bibliografia	92
Spis ilustracji	93

1. Wstęp

Wybierając temat pracy dyplomowej autor chciał stworzyć praktyczne rozwiązanie – aplikację, która mogłaby znaleźć zastosowanie w przedsiębiorstwach niezależnie od branży. Dlatego też wybrano dziedzinę handlową, która w ostatnim czasie na polskim rynku przeżywa prawdziwy boom. Praktycznie, każda korporacja, większe przedsiębiorstwo a nawet małe firmy posiadają własny dział handlowy, a jeżeli nawet nie, to powierzają obowiązki sprzedażowe firmom zewnętrznym, specjalizującym się w tej dziedzinie. Coraz więcej firm docenia możliwość wykorzystania nowoczesnych technologii mobilnych dla wsparcia swoich sieci sprzedaży, co stwarza duże możliwości przed firmami produkującymi tego typu rozwiązania.

Celem niniejszej pracy jest zaprojektowanie oraz wykonanie rozproszonego systemu wsparcia sprzedaży. System ten powinien dawać możliwość tworzenia zamówień, z dowolnego miejsca na świecie, zarówno poprzez wykorzystanie aplikacji mobilnej jak i internetowej. Kolejnymi celami jakie są stawiane przed tworzoną rozwiązaniem są m.in. możliwość zarządzania ofertą produktową, klientami, pracownikami oraz generowanie wybranych raportów opisujących działalność sprzedażową. Tworzone rozwiązanie, musi posiadać strukturę, która w stosunkowo prosty sposób pozwoli dostosować je do specyfiki działania konkretnej firmy. Dla zobrazowania funkcji systemu w niniejszej pracy zostanie opisane wykorzystanie go w firmie sprzedającej części elektroniczne.

Niniejsza praca składa się z dziesięciu rozdziałów.

Po krótkim wstępie, w rozdziale „Analiza” przedstawiona została szerzej specyfika pracy działów handlowych, a w szczególności pracowników pracujących w terenie oraz problemy, które system będzie starał się rozwiązać. Wymienione także są podobne do tworzonej, aplikacje dostępne już na rynku.

Kolejny rozdział, trzeci, opisuje funkcjonalności oferowane przez system przy uwzględnieniu podziału na aplikację mobilną oraz internetową.

W rozdziale czwartym przedstawiony jest opis systemu. Zaprezentowana została architektura rozwiązania oraz projekt bazy danych przy podziale na części odpowiedzialne za wyszczególnione dziedziny problemowe.

Następny rozdział piąty, zawiera opis narzędzi, technologii użytych do stworzenia systemu oraz argumenty przemawiające za nimi przy porównaniu z konkurencyjnymi rozwiązaniami.

Rozdział szósty stanowi po części instrukcję obsługi systemu. Pokazuje możliwości, jakie mają zarówno aplikacja mobilna jak i internetowa. Zaznaja ją ze sposobem obsługi systemu, generowanymi komunikatami oraz raportami.

Kolejny rozdział, siódmy przedstawia specyfikację wewnętrzną. Opisane w nim zostały wybrane funkcje, oraz sposoby użycia poszczególnych technologii.

Rozdział ósmy przedstawia, zagadnienie jakim jest bezpieczeństwo aplikacji mobilnych jak i internetowych. Zostały w nim ukazane podstawowe zagrożenia oraz sposoby przeciwdziałania im.

Rozdział dziewiąty opisuje tematykę testowania aplikacji. Przedstawione w nim zostało także ciekawe zagadnienie, jakim jest zastosowanie automatyzacji testów dla projektowanego systemu.

Ostatni rozdział stanowi podsumowanie pracy, przedstawienie możliwości rozwoju systemu oraz uwag autora.

2. Analiza

Celem pracy jest stworzenie kompletnego rozwiązania, które w znacznym stopniu usprawni codzienną pracę reprezentantów handlowych przedsiębiorstw. Tworzony system wsparcia sprzedaży należy do grupy oprogramowania klasy SFA (Sales Force Automation). System wspomaga zarządzanie sieciami przedstawicieli handlowych w zakresie działalności sprzedażowej oraz marketingowej małych oraz średnich przedsiębiorstw, umożliwiając między innymi elektroniczne zbieranie zamówień, kontrolowanie harmonogramów wizyt handlowych czy przypominanie o płatnościach. Istotnym celem jest zapewnienie użytkownikom mobilnym cennych informacji, które dotychczas były dostępne tylko z systemów typu CRM (ang. Customer Relationship Management) oraz ERP (ang. Enterprise Resource Planning), jedynie dla lokalnych pracowników w siedzibie firmy.

2.1 Specyfika pracy działu handlowego

W większości firm możemy zaobserwować strukturę działu handlowego składającą się z przedstawicieli handlowych, przedstawicieli regionalnych oraz dyrektora sprzedaży. Przedstawiciele handlowi pracujący w terenie odpowiedzialni są za ścisłe kontakty z klientami. Do zadań mobilnego pracownika podczas wizyty u klienta może należeć prezentacja oferty handlowej firmy, zebranie zamówienia czy też udzielenie informacji o zaległych płatnościach. Typowy handlowiec wykonuje około 10 wizyt dziennie trwających około 5-10 minut. W tym czasie przedstawiciel musi przedstawić ofertę firmy, zebrać zamówienie, poinformować o ewentualnych zalegających płatnościach oraz rozprzecznić próbki czy materiały promocyjne. Informacje zbierane są w formie papierowej poprzez wypełnianie specjalnych formularzy zamówień. Następnie reprezentant pod koniec dnia tworzy raport dzienny lub zamówienia w formie, w jakiej zebrał do przedstawiciela regionalnego bądź bezpośrednio do centrali firmy. Tego typu proces przepływu informacji jest nieoptymalny oraz niesie ze sobą wiele potencjalnych zagrożeń. Kluczowe w tym kontekście są:

- możliwe błędy podczas przepisywania informacji z formularzy w celu stworzenia raportu,
- nieefektywne wykorzystanie czasu wizyty (zbieranie informacji na papierze jest bardziej, czasochłonne),
- opóźnienie w realizacji zamówienia wynikające z wprowadzania informacji najczęściej pod koniec dnia,
- brak wiedzy o bieżącej ofercie firmy oraz aktualnej dostępności produktów,
- najczęściej brak historii wizyt odbytych z danym klientem,
- zagubienie dokumentu zamówienia w natłoku dokumentów.

Z drugiej strony mamy ludzi na wyższych stanowiskach kierowniczych, takich jak np. przedstawiciele regionalni, dyrektorzy sprzedaży czy sam zarząd firmy. Osoby te często zainteresowane są ułatwieniami w zakresie generowania różnego rodzaju raportów, zestawień czy podsumowań w celu przedstawienia usystematyzowanych wyników pracy szefostwu lub wspomoczenia procesu tworzenia strategii firmy.

2.2 Rozwiązanie problemu

W dobie coraz większej popularności urządzeń mobilnych oraz taniejących usług firm telekomunikacyjnych dobrym rozwiązaniem jest stworzenie aplikacji działającej na przenośnym urządzeniu typu Pocket PC. Rozwiązanie oparte na tego typu urządzeniu umożliwi przedstawicielowi firmy bardziej efektywny dostęp jej zasobów informacyjnych podczas wizyty u klienta. Poprzez oprogramowanie mobilne handlowiec w łatwy sposób sprawdzi takie informacje, jak dane klienta, produktów dostępnych w ofercie, plany spotkań czy też zaległe płatności. Poprzez aplikację działającą na serwerze centralnym, do systemu dostęp będą mieć także przedstawiciele kadry kierowniczej średniego oraz wysokiego szczebla, tj. przedstawiciele regionalni czy dyrektor sprzedaży. W ich przypadku system ułatwi pozyskanie danych niezbędnych do analizy, oceny efektywności sprzedaży, ustalania pracy przedstawicieli oraz definiowania planów sprzedażowych na kolejne miesiące. Podsumowując zastosowanie technologii mobilnych w sprzedaży w znacznym stopniu przyspieszy sam proces realizacji zamówienia, wyeliminuje zbędne prace biurowe a poprzez część stacjonarną umożliwi efektywne tworzenie raportów oraz kontrolę pracy pionu handlowego przedsiębiorstwa.

2.3 Wybrane rozwiązania konkurencyjne

Na rynku usług informatycznych możemy znaleźć wiele firm polskich oraz zagranicznych oferujących rozwiązania wsparcia sprzedaży. Do grona większych systemów SFA można zaliczyć następujące rozwiązania:

- HamiltonSFA¹ – produkt słowackiej firmy VISICOM. System zbudowany modułowo pozwalający klientowi dopasować rozwiązanie do swojej potrzeby. Wsparcie dla większości rozbudowanych systemów CRM i ERP. Ze względu na cenę wdrożenia rozwiązanie dostępne tylko dla wielkich korporacji.
- ECOD Agent ² firmy Comarch. Część szerokiej oferty produktów z rodziny ECOD rozszerzająca, o zastosowanie urządzeń mobilnych, jedną z największych w Polsce platform do wymiany

¹ HamiltonSFA przegląd funkcjonalności <http://www.hamiltonsfa.eu/buxus/docs/HamOvr200410PLa.pdf>

² Comarch ECOD Agent http://portal.ecod.pl/res/nowe/ECOD_Agent_PL.pdf

elektronicznych faktur, zamówień, stanów magazynowych. System przeznaczony przede wszystkim dla dużych sieci handlowych.

Duże zapotrzebowanie na rozwiązania wspierające sprzedaż mobilną spowodowały powstanie wielu produktów dedykowanych także dla małych przedsiębiorstw. Do grona takich systemów należą między innymi:

- BAH – Bystry Agent Handlowy³. Prosty system posiadający podstawową funkcjonalność Systemu SFA, taką jak zbieranie zamówień, przegląd klientów oraz materiałów. Program wzbogacony o możliwość drukowania faktur i dowodów wpłat. Posiada niezbyt rozbudowany moduł administracyjny.
- eSale⁴ – popularne rozwiązanie wykorzystywane przez wiele polskich firm z sektora MSP. Zamówienia z terenu są automatycznie wprowadzane do programu ERP funkcjonującego w firmie. Minusem rozwiązania jest brak dedykowanego modułu przeznaczonego dla kadry kierowniczej działu handlowego.
- Emigo⁵ – system SFA średniej wielkości posiadający dużą funkcjonalność obejmującą między innymi system ankietowania, badanie konkurencji, wymianę komunikatów między użytkownikami, bazę wiedzy, dostęp dla użytkownika mobilnego do rozbudowanych raportów. Rozwiązanie posiada także aplikację internetową dla kadry zarządzającej, pozwalającą na zarządzanie siecią przedstawicieli oraz przeglądanie raportów.[5]

2.4 Cel pracy

Powodów, dla którego został wybrany temat takiego rodzaju jest kilka. Do jednych z najważniejszych można zaliczyć fakt stosunkowo niewielkiej liczby tego typu rozwiązań dedykowanych dla mikro oraz małych przedsiębiorstw. Większość dostępnych systemów SFA istnieje albo jako duże, skomplikowane oraz drogie systemy albo też jako dodatkowy moduł do oprogramowania ERP, przez co cena wdrożenia wzrasta do niebagatelnych sum, na które wiele małych firm nie jest w stanie sobie pozwolić. Kolejną niewątpliwą zaletą tworzonego systemu będzie wykorzystanie najnowszych dostępnych technologii oraz narzędzi informatycznych, dzięki którym jakość produktu w porównaniu do starszych rozwiązań wzrośnie natomiast cena, z racji szybszego wykonania będzie bardzo konkurencyjna.

³ BAH – system sprzedaży mobilnej <http://www.bah.pl/index.php?page=opis>

⁴ Opis funkcjonalności systemu eSale firmy Mayer S.C. <http://www.mayer.e-tools.pl/main/index.html>

⁵ Opis funkcjonalności systemu Emigo <http://www.sagra.pl/system/Funkcjonalnosci/tabid/493/Default.aspx>

3. Opis funkcjonalności systemu

Po zapoznaniu się z wymaganiami stawianymi przez rynek oraz przykładami aplikacji konkurencyjnych można przedstawić zakres funkcjonalności, jaką będzie oferował system. Poniższy podział stworzony został uwzględniając części programowe oraz role użytkowników systemu.

3.1 Aplikacja mobilna

Aplikacja mobilna stworzona zostanie tylko i wyłącznie dla reprezentantów firmy działających w terenie, czyli przedstawicieli handlowych. Pozwalać ona będzie na:

- przeglądanie informacji o dostępnych produktach,
- dostęp do danych o klientach, danych adresowych placówek klienta oraz do bazy informacji kontaktowych osób, z którymi można odbyć spotkanie,
- przeglądanie aktualnie zleconych zadań, podzlecanie oraz realizację zadań,
- edycję danych osobowych pracownika wykorzystującego urządzenie,
- składanie zamówień bezpośrednio podczas wizyty u klienta,
- synchronizację danych z główną bazą danych systemu.

3.2 Aplikacja webowa

Aplikacja webowa działająca na zdalnym serwerze dostępna będzie poprzez połączenie internetowe z poziomu dowolnej nowoczesnej przeglądarki. Część ta dedykowana jest każdemu autoryzowanemu użytkownikowi systemu, tj. przedstawicielowi handlowemu, przedstawicielowi regionalnemu, dyrektorowi sprzedaży, administratorowi. Zgodnie z założeniem, iż do tej części systemu dostęp będą miały osoby posiadające różne role, dalej opisana jest funkcjonalność dla każdej z nich.

3.2.1 Przedstawiciel handlowy

Przedstawiciel handlowy oprócz dostępu do systemu poprzez aplikację mobilną, może także skorzystać z części stacjonarnej, która zaoferuje mu:

- wygodniejszy sposób przeglądania danych klientów, placówek oraz kontaktów,
- dostęp do informacji o produktach oraz producentach,
- możliwość składania zamówienia, np. podczas telefonicznej rozmowy z klientem,
- przegląd oraz drukowanie faktur wystawionych do swoich zamówień,
- śledzenie swoich wyników sprzedażowych za pomocą przejrzystych wykresów,
- przeglądanie aktualnych zadań, tworzenie oraz zlecanie zadań współpracownikom,

- wgląd oraz edycję swoich danych osobowych oraz hasła dostępu.

3.2.2 Przedstawiciel regionalny

Dla przedstawiciela regionalnego funkcjonalność aplikacji webowej zostanie poszerzona o:

- przeglądanie zamówień swoich oraz pracowników podległych,
- ewidencję klientów firmy oraz kontaktów,
- dostęp do raportów sprzedażowych z przydzielonego regionu,
- przeglądanie informacji o przydzielonych przedstawicielach handlowych.

3.2.3 Dyrektor sprzedaży

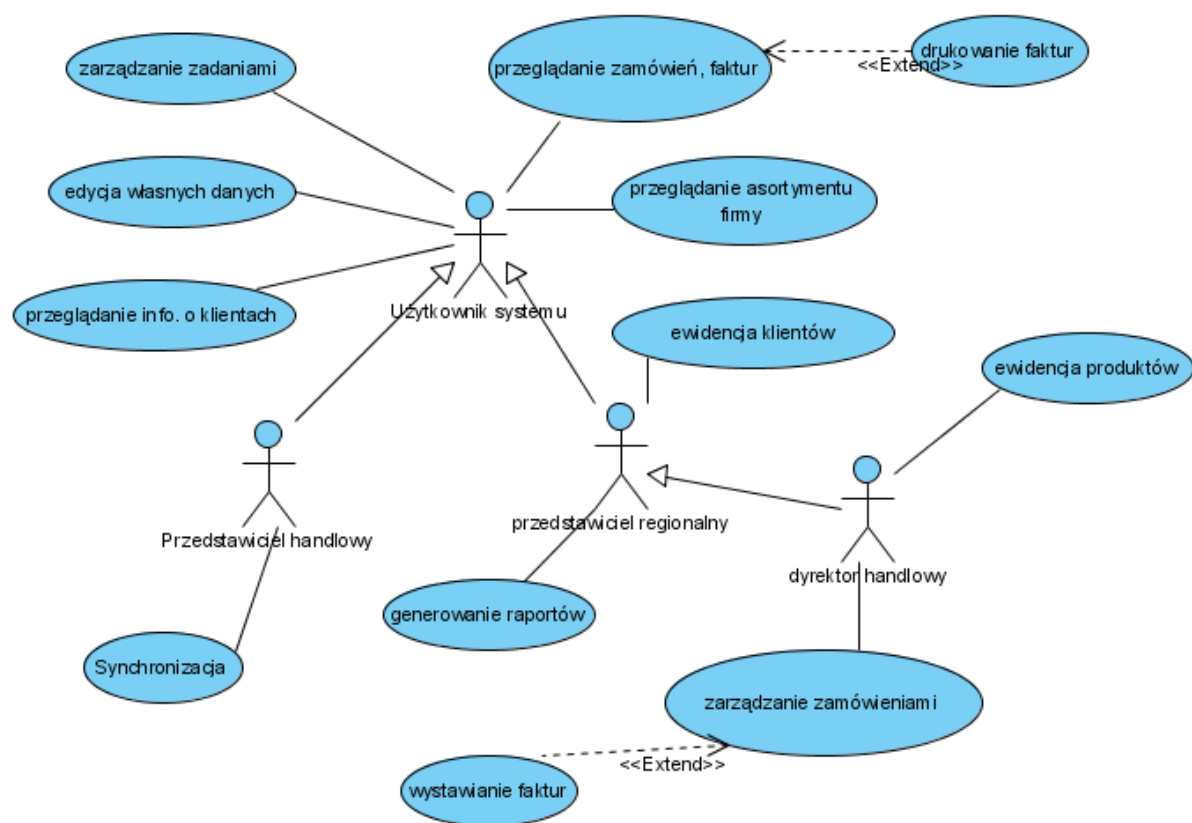
Pomijając wymienione wcześniej funkcje, dyrektor sprzedaży będzie w stanie:

- przeglądać dane wszystkich pracowników, definiować strukturę personalną firmy, czyli przydzielać pracownikom funkcje oraz podwładnych,
- przeglądać wszystkie zamówienia,
- zatwierdzać zamówienia oraz wystawiać faktury,
- generować raporty sprzedażowe według licznych kryteriów.

Podsumowując należy przedłożyć, iż użytkownicy mający dostęp do internetowej części systemu będą mieli możliwość posiadania wielu ról.

3.3 Diagram przypadków użycia

Diagramy przypadków użycia buduje się w celu zobrazowania funkcjonalności systemu oraz wskazania występujących w nim aktorów. Zdecydowanie ułatwiają one zarządzanie zakresem tworzonego rozwiązania informatycznego. Można budować system na zasadzie realizacji kolejnych przypadków użycia. Jest to możliwe dzięki temu, że każdy przypadek użycia stanowi pewną spójną całość. Realizacja systemu w oparciu o przypadki użycia ułatwia uzyskanie szybkiej informacji zwrotnej od użytkownika, dotyczącej poziomu jego akceptacji dla projektu. Na rysunku pierwszym, przedstawiony został diagram przypadków użycia dla tworzonego systemu sprzedaży.



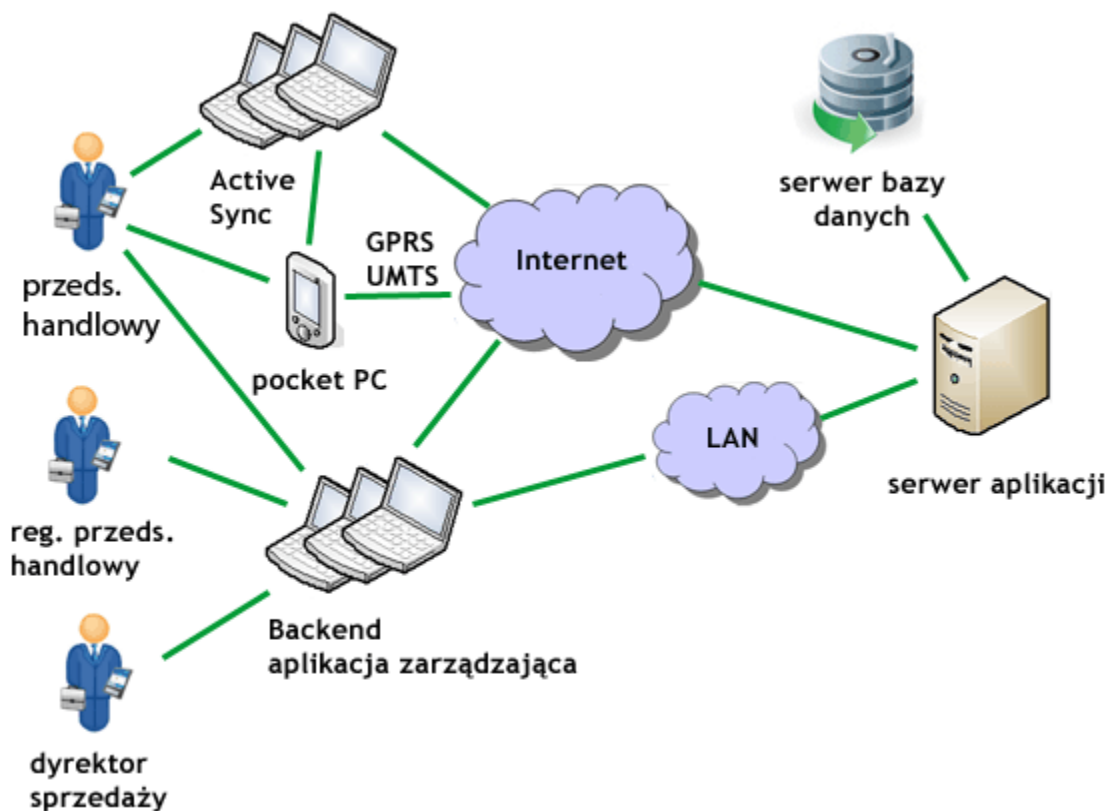
Rysunek 1 - Diagram przypadków użycia

4. Projekt systemu

Niniejszy rozdział przedstawia opis architektury oraz projekt struktury bazy danych budowanego rozwiązania wsparcia struktur sprzedaży.

4.1 Architektura rozwiązania

Tworzony system, jest systemem rozproszonym posiadającym centralny serwer działający w lokalnej sieci firmowej oraz aplikacje klienckie zainstalowane na urządzeniach mobilnych, komunikujące się z nim w drodze replikacji bazodanowej. Projekt zakłada także istnienie aplikacji dostępnej z poziomu dowolnej przeglądarki internetowej prezentującej dane z głównego serwera bazy danych. Przewiduje się możliwość integracji z dowolnymi systemami do zarządzania przedsiębiorstwami dostępnymi na rynku, przy wykorzystaniu dodatkowego oprogramowania. Uogólniony schemat architektury systemu przedstawiony jest na rysunku 2.



Rysunek 2 - Architektura systemu

4.1.1 Architektura aplikacji

Zarówno główna aplikacja internetowa, działająca na serwerze jak i aplikacja mobilna zaprojektowana została z uwzględnieniem podziału na warstwy. Architektura wielowarstwowa zapewnia oddzielenie od siebie warstw prezentacji, logiki biznesowej oraz danych. Podział na niezależne warstwy pozwala na tworzenia systemów zbudowanych z komponentów, co umożliwia ponowne użycie, jak i ułatwia integrację aplikacji z innymi systemami. Pozwala także na oddzielne rozwijanie i aktualizowanie poszczególnych warstw, ułatwia ich konserwację, oraz nie zakłóca działania pozostałych komponentów. W projektowanym systemie wyróżniamy warstwy, takie jak:

- baza danych
- warstwa dostępu do danych (ORM – Nhibernate, ADO .NET – dla aplikacji mobilnej)
- warstwa biznesowa (kod odpowiedzialny za logikę aplikacji, kontrola reguł, przekazywanie obiektów do warstwy prezentacji)
- warstwa prezentacji (strony aspx, kontrolki, okna)
- dodatkowo warstwa narzędziowa Utilities (klasy pomocnicze np. wysyłanie maili, bezpieczeństwo, itp.)

Ogólna wizualizacja omawianej wielowarstwowej architektury przedstawiona została na rysunku 3.



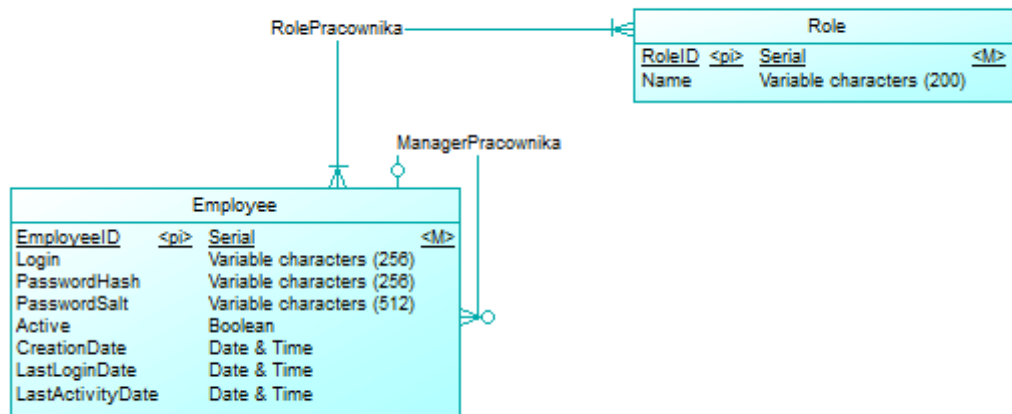
Rysunek 3 - Architektura wielowarstwowa

4.2 Projekt bazy danych

W celu zaprojektowania struktury bazy danych powstał diagram związków encji ERD przedstawiający konceptualny model bazy danych. Do utworzenia schematu posłużono się programem PowerDesigner⁶ w wersji 15.0 firmy Sybase. W celu łatwiejszego zrozumienia problemu prezentacja schematu została zrealizowana z podziałem na jego logiczne fragmenty.

4.2.1 Autoryzacja oraz uwierzytelnianie użytkowników systemu

Na rysunku 4 została przedstawiona część schematu bazy danych odpowiedzialna za logowanie oraz zarządzanie dostępem do określonych części systemu.



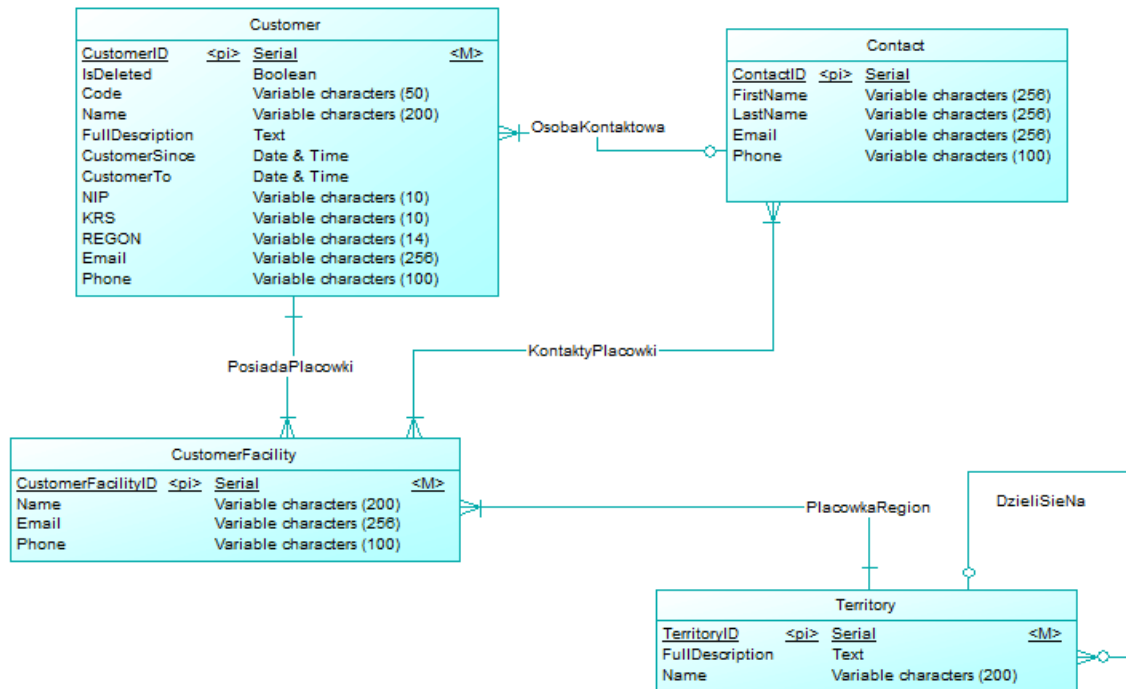
Rysunek 4 Logowanie - użytkownicy

Jak widać schemat jest bardzo prosty, składa się tylko z dwóch tabel. Dostęp do systemu będą mieli tylko pracownicy przedsiębiorstwa, więc tabela *Employee* zawiera dane niezbędne do zalogowania się danego pracownika. Pracownik może posiadać wiele ról w danej firmie (wszystko zależy od rozmiaru firmy) a dana rola będzie określać poziom uprawnień, dostęp do danych funkcjonalności aplikacji.

⁶ PowerDesigner <http://www.sybase.com/products/modelingdevelopment/powerdesigner>

4.2.2 Klienci

Na rysunku 5 przedstawiona została grupa encji odpowiedzialna za przechowanie danych kontaktowych klientów firmy.

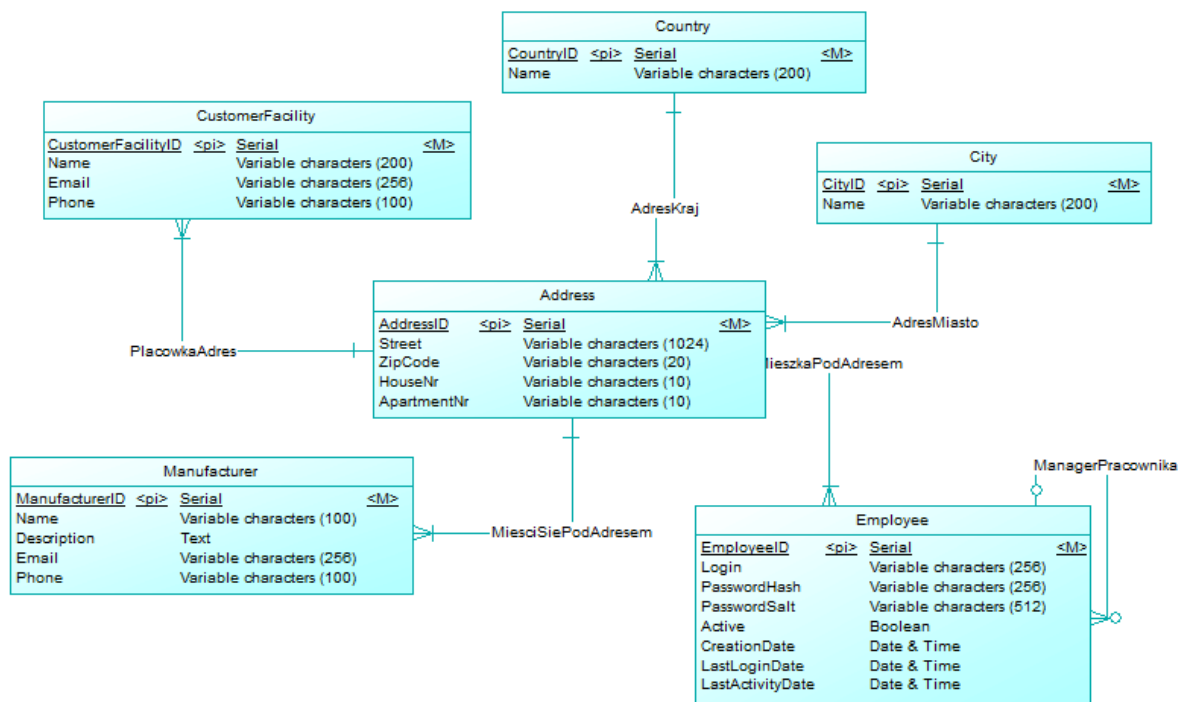


Rysunek 5 Klienci, placówki, kontakty

Klientami mogą być zarówno małe jedno-oddziałowe firmy jak i duże grupy handlowe posiadające wiele siedzib ulokowanych w całym kraju. Projekt bazy danych zakłada istnienie wielu placówek firmy danego klienta oraz różnych osób kontaktowych odpowiedzialnych za utrzymywanie relacji handlowych z daną placówką. Przewiduje się także główną osobę kontaktową dla całej firmy. System umożliwia podział terytorium objętego działalnością handlową firmy na dowolne regiony i przyporządkowanie placówek klienta do tych regionów.

4.2.3 Dane adresowe.

Na rysunku 6 zaprezentowano rozwiązanie służące gromadzeniu danych adresowych pracowników oraz placówek klientów.

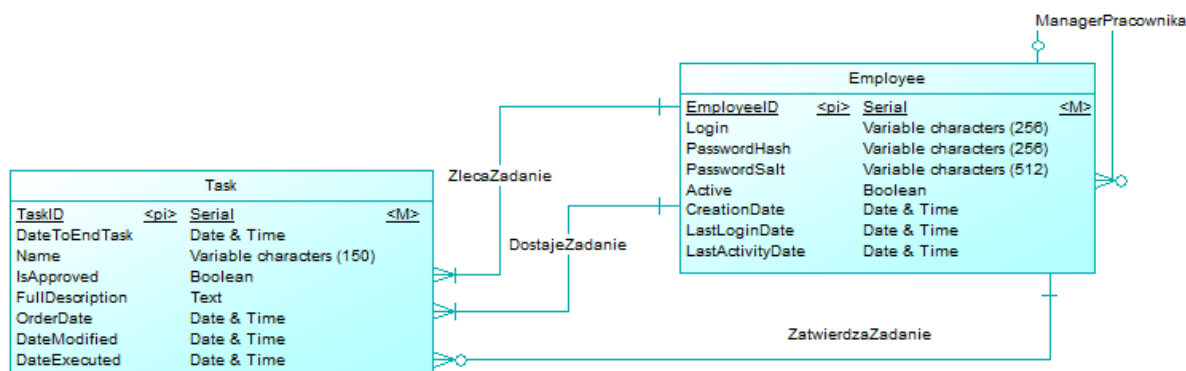


Rysunek 6 Dane adresowe

W bazie danych systemu niepotrzebne jest przechowywanie adresów poszczególnych osób kontaktowych ze strony klientów firmy, więc zrezygnowano z połączenia tabel *Address* oraz *Contact*. Jak widać na powyższym schemacie zdefiniowanie adresu jest niezbędne dla pracowników, producentów oraz placówek klientów.

4.2.4 Pracownik i jego zadania

Kolejna grupa obiektów zaprezentowanych na rysunku 7 jest odpowiedzialna za przechowywanie danych o zadaniach.

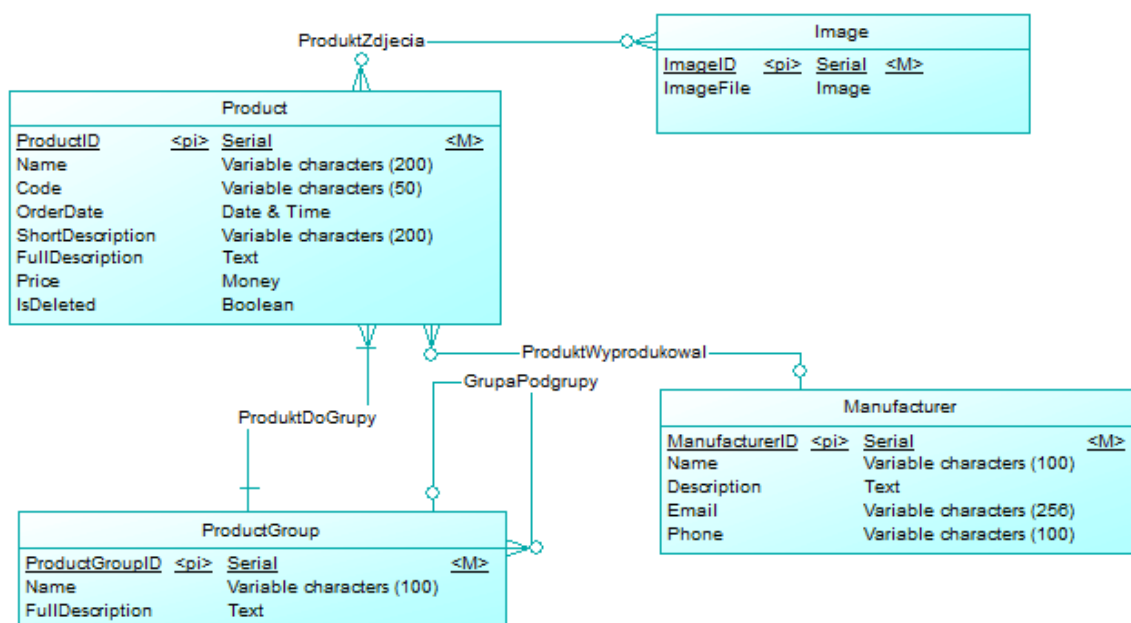


Rysunek 7 Zadania pracowników

Encja *Task* ma za zadanie gromadzenie danych o zadaniach, które są przydzielane pracownikom przez siebie lub też innych pracowników. Pracownik ma możliwość oddelegowania zadania do swojego współpracownika, lub samodzielne zrealizowanie go. Po realizacji, zadanie musi zostać potwierdzone. Czynności tej może dokonać jedynie twórca danego zadania.

4.2.5 Przechowywanie produktów

Na rysunku 8 został przedstawiony schemat bazy danych odpowiedzialny za przechowywanie danych o produktach.

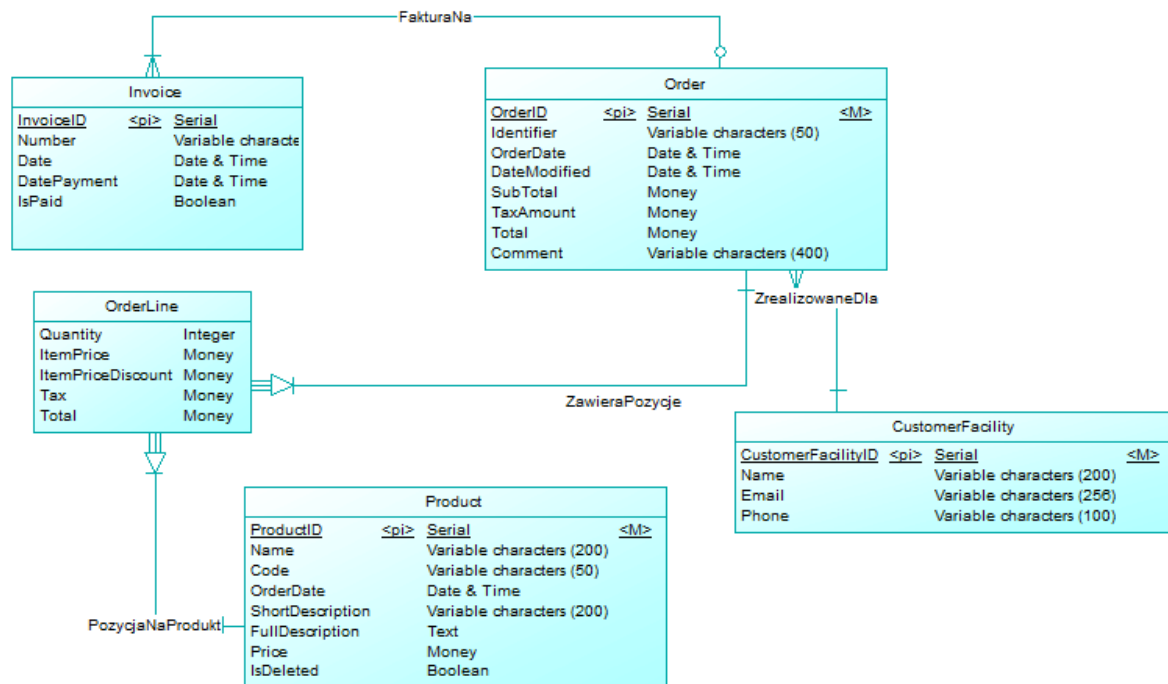


Rysunek 8 Przechowywanie informacji o produktach

Dany produkt może należeć do grupy produktów, która może dzielić się na podgrupy. Oprócz podstawowych danych takich jak nazwa, opis czy cena brutto do produktu możemy przyporządkować jego zdjęcia. Zdjęcia przechowywane są w tabeli *Image*. Tabela *Manufacturer* łącząca się z tabelą produktów w związku jeden do wielu zawiera informacje o producencie, która niewątpliwie jest ważna z uwagi na ergonomię wyszukiwania produktów.

4.2.6 Zarządzanie zamówieniami

Kolejny fragment diagramu przedstawiony na rysunku 9 prezentuje encje oraz relacje niezbędne do przechowywania i zarządzania danymi dotyczącymi zamówień klientów na poszczególne produkty.

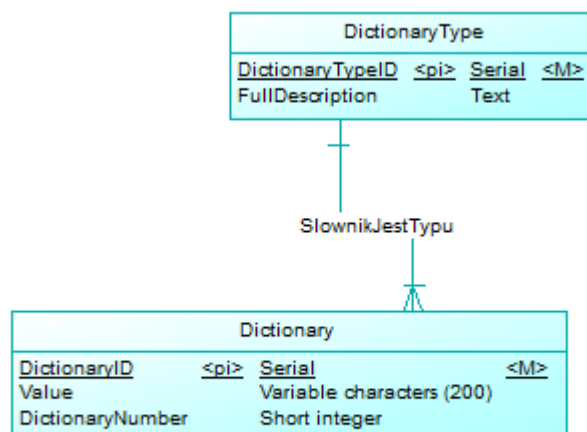


Rysunek 9 Zamówienia, faktury

Każde z zamówień (Tabela *Order* na rysunku 7) opisane jest odpowiednim nagłówkiem zawierającym między innymi datę utworzenia, modyfikacji, wartość zamówienia netto, wartość podatku oraz wartość zamówienia brutto. Zamówienie posiada wiele pozycji, każda z nich jest sporządzona na konkretny produkt z uwzględnieniem ilości, ceny za pojedynczy produkt oraz za całą pozycję. Zamówienia realizowane są dla konkretnej placówki klienta. Produkty w bazie danych przechowywane są nawet po skasowaniu dzięki fladze *IsDeleted*. Rozwiązanie to zastosowano w celu zachowania historycznych zamówień na ten dany produkt.

4.2.7 Przechowywanie danych słownikowych

Ostatni fragment schematu bazy danych przedstawiony na rysunku 10 odpowiedzialny jest za przechowywanie danych słownikowych.



Rysunek 10 Tabele zawierające informacji słownikowe

Zaproponowano w tym celu dwie tabele. Pierwsza z nich *DictionaryType* zawiera typy słowników np. „Stan zamówienia”, „Stan płatności”, „Stawki VAT”, „Status zadania” i inne. Druga tabela *Dictionary* zawiera wszystkie wartości, wszystkich słowników przypisując je danemu słownikowi poprzez relację

5. Narzędzia

Wstępnym etapem tworzenia każdego systemu informatycznego jest podjęcie decyzji o wykorzystaniu konkretnych rozwiązań technologicznych dostępnych na rynku w celu stworzenia końcowego produktu. Decyzję warto podjąć dopiero po dokładnym przeanalizowaniu wielu czynników, takich jak np. dostępności danej technologii, architektury projektowanego systemu, wiedzy zaangażowanych osób na temat wybieranej technologii, ewentualnie szacunkowych kosztów wdrożenia w nową technologię czy kompatybilności wybieranych rozwiązań. Dokonanie nieprzemyślanego wyboru może prowadzić chociażby do drastycznego zwiększenia kosztów przedsięwzięcia a w skrajnym przypadku nawet do całkowitego zaniechania projektu. Dlatego też tak ważna jest wiedza na temat integracji różnych systemów informatycznych, jak i znanie panujących trendów rynkowych.

5.1 Wybór platformy

Projekt zakłada implementację zarówno oprogramowania na urządzenia przenośne typu Pocket PC jak i aplikacji webowej działającej na serwerze WWW firmy. Tego typu okoliczności skłaniają do poszukiwań rozwiązań, które sprostaby obu przypadkom z zachowaniem maksymalnej kompatybilności między nimi. W takim przypadku najkorzystniej jest wybrać kompleksowe technologie dostępne jako całe platformy programistyczne. Na rynku dostępne są w obecnej chwili dwa takie rozwiązania platforma .NET firmy Microsoft oraz platforma J2EE⁷ firmy Sun Microsystems. Platforma stworzona przez firmę Sun Microsystems jest zestawem specyfikacji, dla których utworzonych zostało wiele implementacji (zarówno komercyjnych jak i niekomercyjnych). Microsoft .NET jest natomiast zestawem technologii i produktów stworzonych przez jedną korporację. W dolnej części, w zwięzłej tabelarycznej formie przedstawione jest krótkie porównanie obu technologii uwzględniające potrzeby projektowanego systemu.

⁷ Java Platform, Enterprise Edition - http://pl.wikipedia.org/wiki/Java_Platform,_Enterprise_Edition



Microsoft .NET 	Java 2 Enterprise Edition 
Język programowania, środowisko uruchomieniowe	
<p>Języki C++, C#, Visual Basic .NET, Java (J#⁸), Ruby(IronRuby), Python (IronPython) kompilowane do języka pośredniego Microsoft Intermediate Language (MSIL), system wykonawczy Common Language Runtime (CLR).</p>	<p>Język programowania Java kompilowany do Java bytecode i maszyna wirtualna dla tego języka.</p>
Protokoły komunikacyjne	
COM+, SOAP	IIOP, RMI
Standardowe interfejsy programistyczne	
<p>Microsoft Message Queue (MSMQ) do przesyłania sygnałów i komunikacji asynchronicznej</p> <p>Microsoft Transaction Server (MTA) do realizacji rozproszonych transakcji</p> <p>Active Data Objects .NET (ADO .NET) od komunikacji z bazami danych</p>	<p>Java Messaging System (JMS) do przesyłania sygnałów, komunikacji asynchronicznej</p> <p>Java Transaction API (JTA) do realizowania rozproszonych transakcji</p> <p>Java Database Connectivity (JDBC) do komunikowania się z bazami danych</p> <p>Java Connector Architecture do komunikacji z istniejącymi systemami</p> <p>Java Naming and Directory Interface (JNDI) do komunikacji z usługami katalogowymi</p>
Technologia budowy dynamicznych stron WWW	
ASP .NET, ASP .NET AJAX, ASP .NET MVC (Model View Controller), serwer aplikacji IIS	Java Server Pages (JSP), serwlety, serwer aplikacji JBoss
Technologia budowy aplikacji mobilnych	
Microsoft .NET Compact Framework	Java 2 Platform, Micro Edition (J2ME), Java FX Mobile

Tabela 1 – porównanie platform .NET i J2EE

Obydwie platformy mają zbliżone możliwości, jednak na korzyść rozwiązania korporacji Microsoft przeważa zdecydowanie większa liczba czynników. Platforma .NET dzięki finansowaniu ciągle szybciej się rozwija od platformy firmy Sun Microsystems. Z roku na rok powstają nowe wersje rozwiązań oraz

⁸ Język J# - http://en.wikipedia.org/wiki/J_Sharp

narzędzi do tworzenia aplikacji. Wsparcie marketingowe giganta komputerowego, jakim niewątpliwie jest korporacja Microsoft ma znaczący wpływ na rozwój społeczności programistów, co przekłada się na większe możliwości znalezienia rozwiązań niestandardowych problemów poprzez sieć Internet. Ostatnią ważną rzeczą przemawiającą za platformą .NET jest doświadczenie autora w projektowaniu oraz implementacji aplikacji w tej technologii. Czynniki te także znacząco zmniejsza ryzyko niepowodzenia projektu oraz przyspiesza jego realizację. Podsumowując, technologią wybraną do realizacji systemu wsparcia sprzedaży jest platforma .NET.

5.2 Język programowania

Platforma .NET oferuje spory wachlarz możliwych języków programowania do wyboru. Największą popularnością cieszą się szczególnie dwa z nich tj. Visual Basic .NET⁹ oraz język C#. Dla obu języków poprzez sieć Internet znajdziemy wiele przykładów aplikacji (wraz z kodami źródłowymi) oraz rozwiązań popularnych problemów, co w niewątpliwym sposób ułatwi implementację systemu i pozwoli skupić się na dopracowaniu funkcjonalności oraz architektury. Język VB .NET (Visual Basic .NET) powstał jako rozwinięcie języka Visual Basic specjalnie dla platformy .NET. Jego rozwój idzie w parze razem z rozwojem języka C#, który został w całości stworzony dla platformy .NET jako Microsoftowa odpowiedź na Javę firmy SUN. Najnowsze wersje najpopularniejszych języków używanych w technologii .NET to odpowiednio dla Visual Basic „VBx” (VB 10.0) oraz C# w wersji 4.0. Zgodnie z założeniem architektury system sprzedaży mobilnej zostanie w całości zaimplementowany w jednym języku, więc po uwzględnieniu pewnych czynników wybór padł na język C#. Największe znaczenie w wyborze pomiędzy tymi dwoma językami było większe doświadczenie autora w implementacji rozwiązań w języku C#. Kolejnym czynnikiem jest uznanie języka C# za bardziej nowoczesny oraz stworzony jako język dedykowany dla platformy .NET. Należy także zwrócić uwagę na fakt większej popularności języka C#, jeśli chodzi o tworzenie aplikacji komercyjnych.

Język C#¹⁰ jest to obiektowy język programowania zaprojektowany przez zespół pod kierunkiem Andersa Hejlsberga dla firmy Microsoft. Zgodnie z działaniem platformy .NET program zaimplementowany w tym języku kompilowany jest do języka Common Intermediate Language (CIL), specjalnego kodu pośredniego wykonywanego w środowisku uruchomieniowym platformy .NET. Główne cechy języka C# to między innymi:

- obiektość z hierarchią dziedziczenia o jednym elemencie nadrzędnym (podobnie jak w Javie),

⁹ Visual Basic .NET - http://pl.wikipedia.org/wiki/Visual_Basic_.NET

¹⁰ Język C# - [http://en.wikipedia.org/wiki/C_Sharp_\(programming_language\)](http://en.wikipedia.org/wiki/C_Sharp_(programming_language))

- możliwość implementacji wielu interfejsów przez daną klasę,
- zarządzanie pamięcią przez środowisko platformy; mechanizm Garbage Collectora (GC) odpowiedzialny jest za czyszczenie pamięci po obiektach już nie używanych,
- właściwości, indeksery, delegaty oraz zdarzenia jako nowe elementy składowe klas,
- mechanizm refleksji oraz atrybutów klas,
- kontrola typów.

Język C# w pierwotnej wersji pojawił się w roku 2001, lecz wraz z rozwojem samej platformy oraz narzędzi programistycznych pojawiły się nowe wersje czyli C# 2.0, C# 3.0 oraz obecnie w fazie beta testów C# 4.0. Warto zwrócić uwagę na kilka nowoczesnych funkcjonalności jakie zostały dodane do tego języka od chwili jego powstania tj:

- typy generyczne – rozszerzają język C# o polimorfizm parametryczny (podobne do szablonów w C++),
- typy Nullable – bardzo przydatne w pracy z bazami danych, ponieważ uwzględniają możliwość nie istnienia danej wartości w bazie,
- Language Integrated Query (LINQ) – zestaw rozszerzeń języka ułatwiający pracę ze zbiorem danych. Umożliwia wykonywanie zapytań na obiektach,
- typy domniemane: „var foo = ”hello”;” oznacza to samo, co „string foo= ”hello”;”,
- wyrażenia lambda - kod „listOfFoo.Where(delegate(Foo x) { return x.Size > 10; });” możemy zastąpić „listOfFoo.Where(x => x.Size > 10);”.

System wsparcia sprzedaży zostanie wykonany w technologii .NET Framework 3.5 przy użyciu języka C# w wersji 3.0, gdyż jest to najnowsza, dostępna jako stabilna wersja tego języka. W celu stworzenia warstwy prezentacji aplikacji webowej systemu użyta zostanie technologia o nazwie ASP .NET¹¹ (Active Server Pages .NET). Jest to platforma służąca do tworzenia dynamicznych stron www, aplikacji internetowych oraz usług sieciowych. Technologia ASP .NET może być użyta razem z dowolnym obsługiwany przez platformę .NET językiem programowania.

5.3 Środowisko programistyczne

Kolejnym krokiem po wyborze technologii oraz języka programowania jest wybór odpowiedniego, spełniającego potrzeby zintegrowanego środowiska programistycznego. Najpopularniejszym rozwiązaniem jest pakiet Visual Studio¹² produkowany przez firmę Microsoft w kilku wersjach w zależności od potrzeb. Z racji dużej popularności oraz dostępności wymienionego środowiska

¹¹ The Official Microsoft ASP .NET Site - <http://www.asp.net/>

¹² Witryna dla programistów - <http://msdn.microsoft.com/pl-pl/bb736140.aspx>

zostanie ono wybrane jako główne narzędzie realizacji systemu. Jednakże, warto wspomnieć o innych rozwiązaniach jakie mamy dostępne na rynku, zarówno komercyjnych jak i open source. Pierwszym z nich jest pakiet Borland Developer Studio 2006¹³ (BDS 2006) w skład którego wchodzi Borland C#Builder 2006. Środowisko to dostarcza wszystkie niezbędne funkcjonalności, aby tworzyć zarówno aplikacje webowe jak i desktopowe, jednak brak jest wsparcia dla tworzenia aplikacji mobilnych przy użyciu Compact Framework¹⁴(CF). Innym rozwiązaniem jest SharpDevelop¹⁵, który jest środowiskiem programistycznym dla platformy .NET rozwijanym jako projekt open source. Aplikacja w najnowszej wersji (3.0) obsługuje .NET Framework w wersji 3.5 i jest bardzo dobrą alternatywą do tworzenia oprogramowania dla osób nie przepadających za Visual Studio oraz chcących rozwijać swoje rozwiązania na działającej pod Linuksem platformę Mono¹⁶ (implementacja platformy .NET dla systemu Linux, wspierana przez firmę Novell). SharpDevelop proponuje kilka ciekawych rozwiązań takich jak np. tłumaczenie kodu Visual Basic .NET na C# i odwrotnie czy też zintegrowanie pakietu NUnit¹⁷ w celu tworzenia testów jednostkowych. Chociaż wspomniane dwa alternatywne dla Visual Studio rozwiązania do tworzenia aplikacji nie stanowią mocnej konkurencji, to oferują bardzo ciekawe funkcjonalności, a w przypadku programu SharpDevelop rozszerzają możliwość implementacji rozwiązań w technologii .NET na inne platformy systemowe, takie jak np. Linux.

5.4 Serwer bazy danych

Wybór języka, platformy programistycznej oraz narzędzi miał istotny wpływ przy podejmowaniu decyzji odnośnie serwera bazy danych. W wypadku obrania dla projektu technologii firmy Microsoft idealnym rozwiązaniem oraz najbardziej kompatybilnym z uprzednio wybranymi narzędziami jest oparcie bazy danych na serwerze SQL Server firmy Microsoft. Od niedawna w sprzedaży dostępna jest wersja SQL Server 2008 aczkolwiek w projekcie zostanie użyta wersja poprzednia czyli SQL Server 2005. Jednakże, serwer firmy Microsoft nie jest jedynym dobrym produktem występującym na rynku. Pod uwagę brane były także inne rozwiązania takie jak: Oracle 10g czy MySQL. Poniżej został przedstawiony krótki opis każdego z produktów.

Microsoft SQL Server 2005¹⁸ – Rozwiązanie SQL Server zawiera zaawansowane funkcje, między innymi usługi raportowania SQL Server 2005 Reporting Services, a także graficzne narzędzie do łatwego zarządzania bazą danych SQL Server 2005 Management Studio. Serwer ten jest stosunkowo

¹³ Borland Developer Studio 2006 - <http://www.codegear.com/products/bds2006>

¹⁴ Codeguru.pl – Środowisko .NET Compact Framework - <http://www.codeguru.pl/article-507.aspx>

¹⁵ SharpDevelop - <http://www.icsharpcode.net/OpenSource/SD/>

¹⁶ Mono project - http://www.mono-project.com/Main_Page

¹⁷ NUnit Home - <http://www.nunit.org/index.php>

¹⁸ SQL Server 2005 - <http://www.microsoft.com/poland/sql/2005/default.mspx>

łatwy w zarządzaniu oraz dostępny w wielu wersjach m.in. Express, Workgroup, Standard czy Enterprise co umożliwi dostosowanie do konkretnego klienta.

Oracle 10g¹⁹ Podobnie do oprogramowania firmy Microsoft jest to produkt komercyjny posiadający kilka wersji, w tym i darmową Express. Zarządzanie bazą Oracle jest trudniejsze w porównaniu do jej konkurenta SQL Server 2005, natomiast jako plus dla Oracle można liczyć multiplatformowość (nie ograniczenie się do jednego systemu operacyjnego). Baza danych Oracle posiada wbudowany język do tworzenia procedur składowanych PL/SQL.

MySQL²⁰ – jest open sourceowym system zarządzania bazą danych utworzonym przez szwedzką firmę MySQL AB, która w wyniku rynkowych zakupów oraz konsolidacji przeszła w „ręce” firmy Sun Microsystems, a następnie korporacji Oracle, która przejęła wcześniej wymienioną firmę. MySQL nie posiada zbyt dużej funkcjonalności, lecz uchodzi za dość szybki system bazy danych. Ciekawostką jest fakt, że dopiero w najnowszej wersji (5.0) dodano takie podstawowe rozwiązania jak procedury składowane, wyzwalacze, perspektywy, kursory. Zaletą tego systemu jest wsparcie dla wielu platform systemowych takich jak Linux, Windows, Solaris, FreeBSD, MacOS X.

5.5 Mobilny serwer bazy danych

Niezbędnym krokiem jest jeszcze określenie systemu zarządzającego danymi na urządzeniach Pocket PC. Przy wyborze SQL Servera jako głównego systemu bazy danych najlepszym wyborem dla urządzeń mobilnych będzie SQL Server Compact 3.5²¹. Po instalacji tego środowiska pod systemem Windows Mobile mamy dostęp do mobilnej wersji narzędzia Query Analyzer, z poziomu którego możemy wykonywać operacje na naszej bazie. Baza SQL Server Compact 3.5 wspiera także bardziej zaawansowane mechanizmy np. merge replication.

5.6 ORM

W celu ułatwienia pracy z bazą danych dobrą praktyką jest zastosowanie narzędzia służącego do mapowania obiektowego-relacyjnego. Wybór w tym przypadku padł na znany i sprawdzony NHibernate²², rozwijany na zasadach open source. Niestety w chwili obecnej projekt ten nie posiada wersji działającej w środowisku Windows Mobile, więc zostanie użyty tylko przy implementacji aplikacji internetowej. NHibernate jest produktem wywodzącym się z projektu Hibernate dedykowanego dla aplikacji tworzonych w języku Java, który już po kilku latach od pierwszego

¹⁹ Oracle Database 10g Express Edition - <http://www.oracle.com/technology/products/database/xe/index.html>

²⁰ MySQL wikipedia - <http://pl.wikipedia.org/wiki/MySQL>

²¹ SQL Server Compact 3.5 - <http://www.microsoft.com/Sqlserver/2005/en/us/compact.aspx>

²² NHibernate Home Page - <http://nhforge.org/Default.aspx>

wydania stał się wiodącą z technologii ORM kierowanych do twórców aplikacji biznesowych. Narzędzie to posiada bardzo dobrą dokumentację oraz szeroki wachlarz zakończonych sukcesem projektów. Zastosowanie NHibernate nie tylko znacznie ogranicza czas potrzebny na stworzenie aplikacji, lecz także niejednokrotnie okazuje się, że systemy zbudowane na tej technologii są wydajniejsze i łatwiej podatne na zmiany.

Kilka zalet NHibernate:

- wsparcie dla transakcji,
- wbudowane cache pierwszego oraz drugiego poziomu,
- przetwarzanie wsadowe (przesyłanie serii zapytań do serwera za jednym razem),
- wsparcie przy tworzeniu zapytań („Criteria API”, „Linq to NHibernate”),
- „lazy loading” – podczas pobierania obiektów, kolekcję należące do tych obiektów, pobierane są w czasie, gdy niezbędne jest ich użycie,
- „eager fetching” – przeciwieństwo „lazy loading”; NHibernate posiada mechanizm, umożliwiający w łatwy sposób wybieranie trybu pobierania danych (można pobrać dane obiektów w kolekcji należącej do innego obiektu, co eliminuje problem zwany „Select N+1”²³),
- pełne wsparcie dla typów generycznych,
- wiele projektów komercyjnych jak i darmowych wspomagających pracę z NHibernate, np. NHProf²⁴ (testowanie wydajności zapytań) czy MyGeneration²⁵ służący do generowania klas oraz plików xml opisujących mapowanie.

²³ “Select N+1” problem - <http://nhprof.com/Learn/Alert?name=SelectNPlusOne>

²⁴ NHprof - <http://nhprof.com/>

²⁵ MyGeneration - <http://www.mygenerationsoftware.com/portal/default.aspx>

6. Specyfikacja zewnętrzna

Specyfikacja zewnętrzna opisuje sposób interakcji tworzonego systemu z użytkownikiem. Rozdział ten jest pewnego rodzaju instrukcją obsługi, pokazuje w jaki sposób realizować procesy biznesowe zaimplementowane w aplikacji. System wsparcia sprzedaży składa się z dwóch odrębnych aplikacji, komunikujących się ze sobą w drodze replikacji, tak więc obie zaimplementowane części zostaną opisane oddzielnie.

6.1 Aplikacja mobilna – Pocket PC

Aplikacja mobilna jest dedykowana na urządzenie mobilne typu Pocket PC posiadające system operacyjny Windows Mobile 5.0 i wzwyż. Ta część systemu przeznaczona jest tylko i wyłącznie dla przedstawicieli handlowych firmy, służy bowiem uproszczeniu oraz przyspieszeniu ich pracy w terenie.

Dzięki zastosowaniu mobilnej bazy danych, reprezentanci firmy, mogą gromadzić dane o zamówieniach czy przeglądać dane klientów oraz produktów w trybie bezpołączeniowym (offline). Podczas synchronizacji należy zadbać o połączenie urządzenia Pocket PC z Internetem lub lokalną siecią firmy, w której znajduje się główny serwer bazy danych. Połączenie to można zrealizować w różny sposób, np. łącząc urządzenie przenośne za pomocą stacji dokującej z komputerem stacjonarnym mającym dostęp do serwera lub bezpośrednio, wykorzystując sieć WiFi lub korzystając z możliwości sieci komórkowej (GPRS).

6.1.1 Logowanie – pierwsza synchronizacja

Po uruchomieniu aplikacji na urządzeniu przenośnym pojawia się ekran logowania (Rysunek 11). Aby rozpocząć pracę z aplikacją, należy w pierwszej kolejności przeprowadzić synchronizację. Podczas pierwszego logowania aplikacja łączy się z usługą sieciową dostępną na głównym serwerze firmy, która przeprowadza uwierzytelnianie i autoryzację użytkownika. Dlatego też należy zadbać o prawidłową konfigurację połączenia internetowego. Po poprawnym wykonaniu tej procedury można przejść do procesu synchronizacji. Służy do tego przycisk, który pojawił się, jeżeli pierwsze logowanie zakończyło się sukcesem (Rysunek 12). Synchronizacja przebiega w sposób asynchroniczny w jej trakcie widoczny jest pasek postępu oraz może być w każdej chwili anulowana (zmiany zostaną wtedy wycofane). Jeżeli synchronizacja się zakończy, użytkownik może zalogować się do aplikacji (nie wymaga to już połączenie z serwerem) i rozpocząć pracę.



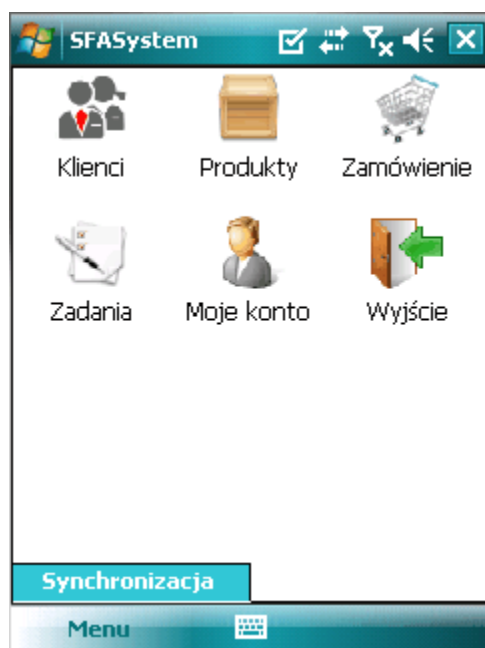
Rysunek 11 - Ekran logowania



Rysunek 12 – Proces synchronizacji

6.1.2 Ekran główny

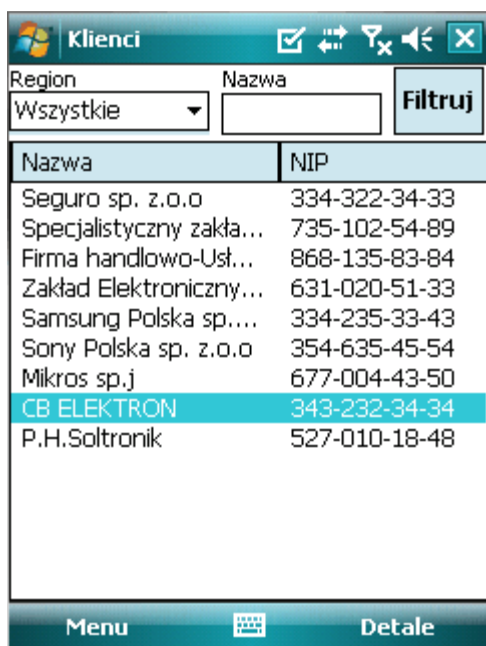
Rysunek 13 zawierający menu główne aplikacji, ukazuje się po pomyślnym procesie autoryzacji opisanym w poprzednim podrozdziale. Z tego punktu dla użytkownika dostępne są odnośniki do wszystkich podstawowych funkcjonalności programu mobilnego, czyli przeglądania listy klientów, produktów, zadań, tworzenia zamówienia oraz edycji swojego osobistego konta. W lewym dolnym menu dostępna jest także opcja synchronizacji.



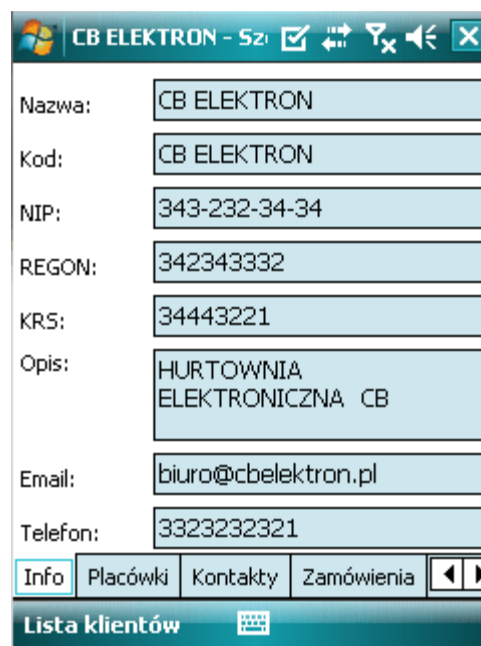
Rysunek 13 - Menu główne

6.1.3 Klienci

Pierwsza opcja menu głównego aplikacji dotyczy przeglądu listy klientów firmy (Rysunek 14). Korzystając z dostępnych na górze ekranu filtrów, możemy zawęzić listę klientów do występujących w danym regionie oraz zawierających dany ciąg liter w nazwie. Aby wyświetlić detale klienta, należy zaznaczyć go na liście oraz kliknąć opcję przycisk „Detale” widoczny w prawym dolnym rogu.



Rysunek 14 - Lista klientów



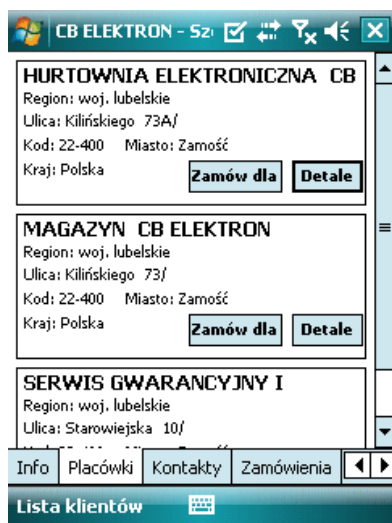
Rysunek 15 - Informacje o kliencie

Na rysunku 15 widzimy szczegółowe informacje o danym kliencie, są to tylko podstawowe dane, reszta dostępna poprzez wybranie interesującej nas zakładki. Nie wszystkie zakładki są widoczne na ekranie, dlatego do przesuwania ich widoku służą dwie strzałki widoczne w dolnym rogu.

Zakładka placówki i kontakty

Lista placówek uprzednio wybranego klienta pokazana jest na rysunku 16. Dla każdej z siedzib kontrahenta widoczne są dane adresowe oraz dwa przyciski „Zamów dla” i „Detale”. Kliknięcie na pierwszy z nich tworzy nowe zamówienie dla wybranej placówki. Przed dokonaniem tej czynności wyświetla się okno dialogowe z potwierdzeniem wyboru, które po zatwierdzeniu przenosi użytkownika do ekranu zamówienia (Rysunek 26). Kliknięcie na drugi przycisk („Detale”), prowadzi do pojawienia się informacji o placówce (Rysunek 17). Zakładki dostępne po wyświetleniu szczegółów placówki są podobne do dotyczących klienta, dlatego też omawianie ich jest zbędne. Warto tylko nadmienić, że różnica polega na ograniczeniu listy kontaktów, zamówień oraz płatności do wybranej placówki. Reszta funkcjonalności pozostaje taka sama.

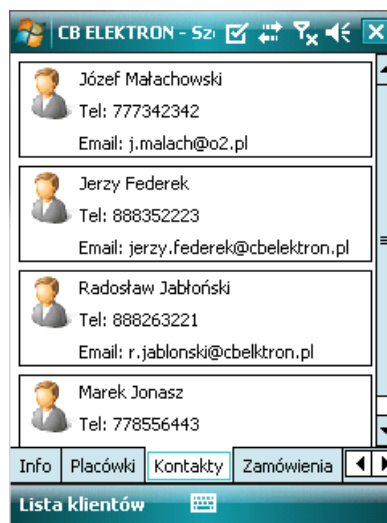
Na rysunku 18 przedstawiono widok zakładki „Kontakty” dostępnej w detalach klienta.



Rysunek 16 - Lista placówek



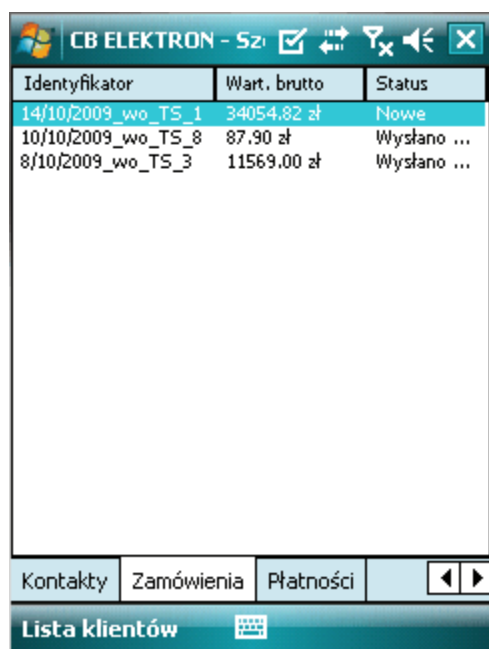
Rysunek 17 - Informacje o placówce



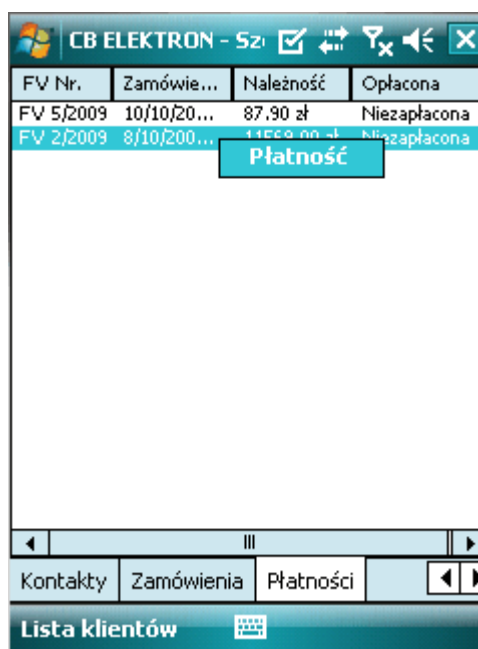
Rysunek 18 - Lista kontaktów klienta

Zakładka zamówienia i płatności

Ostatnie dwie zakładki detali klienta, pokazują odpowiednio: listę zamówień złożonych przez klienta (Rysunek 19) oraz listę płatności dla zamówień, dla których została wystawiona faktura (Rysunek 20).



Rysunek 19 - Lista zamówień

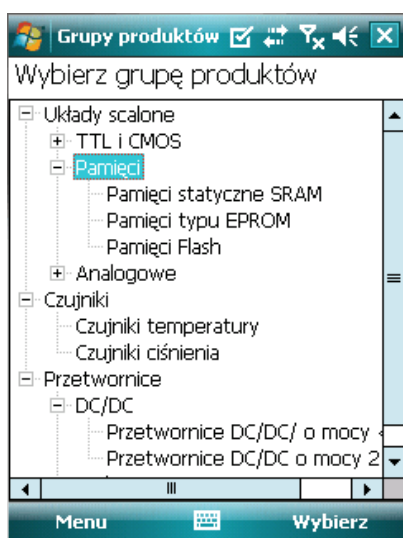


Rysunek 20 - Lista płatności

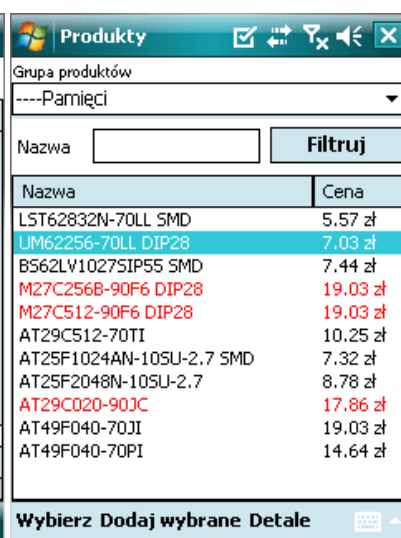
Przedstawiciel handlowy za pomocą urządzenia ma możliwość zarejestrowania faktu realizacji płatności przez klienta. Pozwala mu to na zbieranie zalegających należności od klientów firmy. Aby wykonać tą operację, należy przytrzymać rysik na wybranej płatności i z menu rozwijanego wybrać opcję „płatność” (sytuacja pokazana na rysunku 20).

6.1.4 Produkty

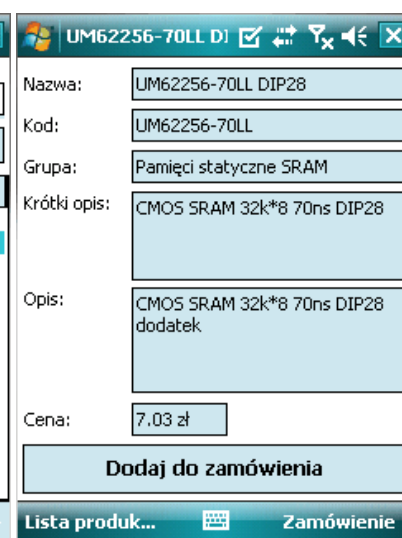
Kolejna z ważnych funkcjonalności dostępnych poprzez menu główne aplikacji to przeglądanie produktów dostępnych jako asortyment firmy. Klikając na ikonę podpisaną „Produkty” wyświetlone zostaje drzewo grup produktów (rysunek 21). Aby wyświetlona została lista produktów (rysunek 22) danej grupy, należy zaznaczyć ją w drzewie, a następnie kliknąć na przycisk „Wybierz” widoczny w prawym dolnym rogu. Analogicznie do wyświetlenia wszystkich produktów możemy wybrać z drzewa korzeń główny nazwany „Wszystkie”. Po wykonaniu jednej z opisanych operacji pojawia się okno z produktami firmy (rysunek 22), które można filtrować według nazwy oraz grupy produktów.



Rysunek 21 - Grupy produktów



Rysunek 22 - Lista produktów

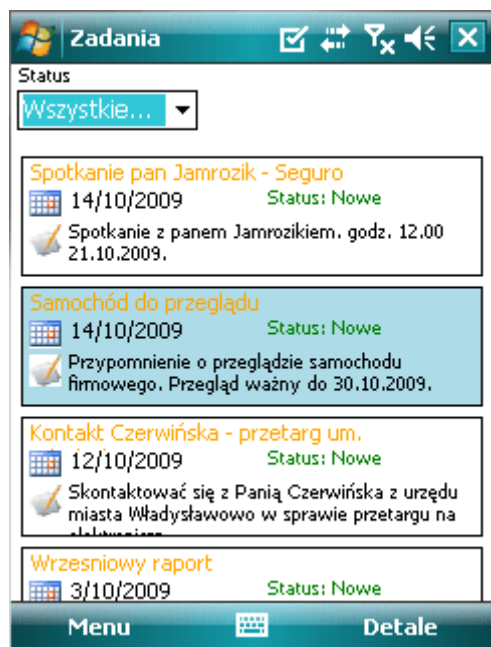


Rysunek 23 - Detale produktu

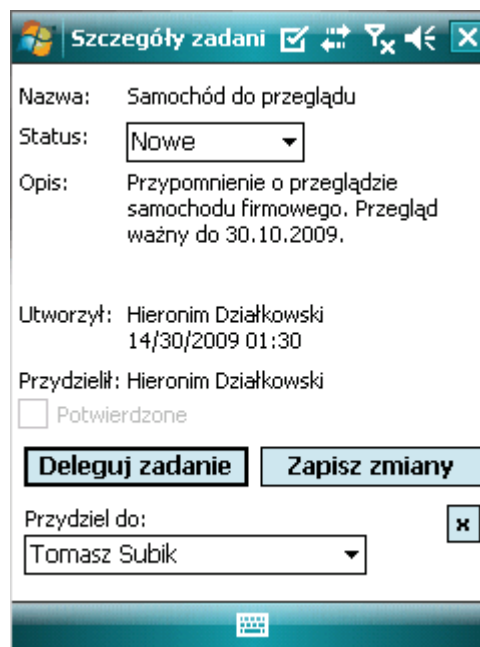
W menu dolnym ekranu *Lista produktów*, za pomocą dwóch opcji „Wybierz” oraz „Dodaj wybrane”, użytkownik aplikacji mobilnej może w prosty i szybki sposób dodawać produkty do zamówienia (więcej informacji o tworzeniu zamówienia można znaleźć w rozdziale 6.1.6). Po zaznaczeniu produktu na liście i kliknięciu opcji „Wybierz”, kolor czcionki wybranego elementu jest zmieniany na czerwony. Można wybrać dowolną liczbę produktów z aktualnie wyświetlanych, a następnie dodać je do zamówienia klikając przycisk „Dodaj wybrane”. Po zatwierdzeniu dokonanego wyboru system uaktualni zamówienie (zostanie to obwieszczone odpowiednim komunikatem ze strony aplikacji). Ostatnia z opcji menu dolnego ekranu *Lista produktów*, czyli „Detale”, służy do wyświetlenia okna zawierającego szczegółowe informacje na temat produktu (rysunek 23). Z poziomu tego okna istnieje również możliwość dodania produktu do zamówienia klikając przycisk „Dodaj do zamówienia”. Natomiast po wybraniu opcji „Zamówienie” z dolnego menu, aplikacja przechodzi do ekranu tworzonego zamówienia.

6.1.5 Zadania

Na rysunku 24 przedstawiono okno *Lista zadań*. Aby przejść do tego ekranu, należy w menu głównym programu wybrać ikonę „Zadania”.



Rysunek 24 - Lista zadań



Rysunek 25 - Szczegóły zadania

Wyświetlone zadania posortowane są według daty utworzenia począwszy od ostatnio utworzonego. Aplikacja mobilna ogranicza się do pokazania tylko zadań, które zostały przydzielone aktualnemu zalogowanemu użytkownikowi urzędnika. Listę tę można ograniczyć do zadań o wybranym statusie według filtra dostępnego w lewym górnym rogu ekranu. Jak widać na rysunku 24, opis zadania jest ograniczony do wielkości obszaru przeznaczonego na wyświetlenie pojedynczego elementu listy. Dlatego też, chcąc zobaczyć pełny opis, zmienić status, czy też przydział zadania, należy przejść do ekranu *Szczegóły zadania*. Wykonanie tej operacji odbywa się poprzez zaznaczenie zadania na liście (zostanie ono wtedy podświetlone na niebiesko), a następnie kliknięcie przycisku „Detale”. Wyświetlone w ten sposób okno (Rysunek 25), zawiera szczegółowo wszystkie informacje o danym zadaniu oraz umożliwia zmianę jego statusu lub oddelegowanie go do innego pracownika.

6.1.6 Tworzenie zamówienia

Najważniejszą funkcją z racji przeznaczenia systemu, jest niewątpliwie składanie zamówień bezpośrednio podczas wizyty u klienta. Aplikacja mobilna pozwala na tego typu operację w prosty i szybki sposób. Należy wpięrow nadmienić, iż w jednym czasie może być kompletowane tylko i wyłącznie jedno zamówienie. Dzieje się to dlatego, że działanie procesu zbierania produktów do zamówienia jest analogiczne do wstawiania przedmiotów do koszyka w sklepie internetowym.

Aktualnie kompletowane zamówienie jest globalnie dostępne w pamięci aplikacji i wszystkie operacje, takie jak np. dodawanie produktów (opisane w rozdziale 6.1.4) czy definiowanie placówki, dla której tworzone jest zamówienie (opisane w rozdziale 6.1.3) są wykonywane na tym jednym konkretnym egzemplarzu. Aby wyświetlić detale obecnie tworzonego zamówienia, można kliknąć ikonę „Zamówienie” znajdującą się w menu głównym aplikacji. Po wykonaniu tej, albo innej operacji przechodzącej do zamówienia pokazuje się okno widziane na rysunku 26.

L...	Produkt	Ilość	Cena
1	NE5532P	100	115.00 zł
1	OP07CP	50	47.50 zł
1	TL062CN	50	32.50 zł
1	TL064CD SMD	40	28.80 zł
1	DS18B20	60	426.60 zł
1	KT100	70	133.70 zł
1	LM335Z	300	654.00 zł
1	NE5532P	30	345.00 zł

Rysunek 26 - Kompletacja zamówienia

Rysunek 27 - Składanie zamówienia

Okno to jest oknem kompletacji zamówienia. Znajduję się w nim lista aktualnie dodanych produktów, uwzględniająca nazwę, ilość oraz cenę dla każdego z linii zamówienia. W celu zaktualizowania liczby przedmiotów w danej linii, należy wybrać ją, a następnie zmienić wartość pola tekstowego używając przycisków „+”, „-”, lub wpisując żądaną ilość. Po zmianie wymagane jest zatwierdzenie przyciskiem „Aktualizuj”. Usuwanie produktów jest realizowane poprzez wybranie linii zamówienia oraz wybranie opcji „Usuń wyb.” z menu dolnego ekranu. Kliknięcie na kolejną opcję menu dolnego, a konkretnie „Dodaj prod.” przenosi użytkownika do ekranu z wyborem grupy produktów. Wtedy, aby dodać pozycje do zamówienia, należy kierować się instrukcją zamieszczoną w rozdziale 6.1.4 „Produkty”. Wracając do okna *Kompletacja zamówienia*, należy nadmienić, iż nie jest możliwe złożenie pustego zamówienia, lub bez uprzednio wybranej placówki, dla której zamówienie jest tworzone (błędy wyświetlane są przy użyciu stosownych komunikatów). Ostatnią czynnością, która jeszcze nie została omówiona jest wybór placówki handlowej. Przycisk „Wybierz” znajdujący się w prawej, górnej części ekranu, przenosi użytkownika do okna wyboru klienta, gdzie kierując się instrukcjami zawartymi w rozdziale 6.1.3 „Klienci”, można dodać placówkę do zamówienia. Po skompletowaniu wszystkich

niezbędnych informacji, składamy zamówienie przyciskając „Zatwierdź zamówienie”. Jeżeli zamówienie zostanie pomyślnie dodane, wyświetlony zostanie odpowiedni komunikat (Rysunek 27). Kończącą czynnością, o której należy pamiętać jest synchronizacja z centralną bazą danych.

6.1.7 Konto osobiste

Ostatni ekran aplikacji mobilnej służy do zmian ustawień konta osobistego (Rysunek 28). W oknie tym brak jest funkcjonalności dotyczącej zmiany, np. hasła użytkownika. Tą czynność można zrobić tylko z poziomu aplikacji webowej.



Rysunek 28 - Ustawienia konta osobistego

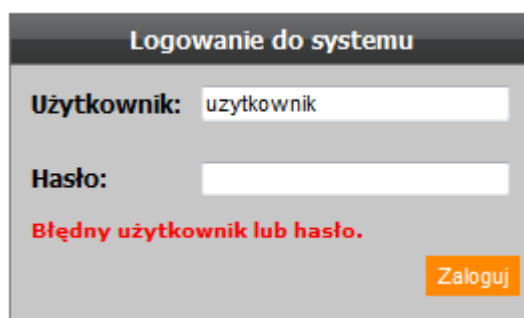
6.2 Aplikacja WWW

Aplikacja WWW została stworzona głównie w celu ułatwienia pracy kadrze menadżerskiej działu handlowego. Jednakże, dostęp do niej mają także przedstawiciele handlowi oraz osoba o uprawnieniach administratora, którą równie dobrze może być dyrektor sprzedaży. Aplikacja ta uruchomiona jest na serwerze firmy, do którego z przyczyn bezpieczeństwa nie powinno być łatwego dostępu z poziomu publicznej sieci jaką jest Internet. Jednakże, pracowników działów handlowych szczególnie cechuje duża mobilność, często pracują w domach lub hotelach. Dlatego też w takich sytuacjach, do tej części systemu zalecany jest dostęp przez skonfigurowaną wirtualną sieć prywatną

(VPN²⁶ ang. Virtual Private Network), która ze względu na kompresję oraz szyfrowanie danych charakteryzuje się dużą efektywnością oraz poziomem bezpieczeństwa.

6.2.1 Logowanie

Dostęp do funkcji aplikacji jest całkowicie zablokowany dla użytkowników niezalogowanych. Konta pracowników, może tworzyć tylko osoba posiadająca do tego uprawnienia, może być to dyrektor lub administrator. Po uruchomieniu aplikacji w przeglądarce WWW, użytkownik jest proszony o wypełnienie formularza przedstawionego na rysunku 29, podając swój login i hasło.



The image shows a web-based login form. At the top, there is a dark grey header with the text 'Logowanie do systemu' in white. Below the header, the form has a light grey background. It contains two input fields: the first is labeled 'Użytkownik:' and contains the text 'uzytkownik'; the second is labeled 'Hasło:'. Below these fields, there is a red error message that reads 'Błędny użytkownik lub hasło.'. At the bottom right of the form, there is an orange button with the text 'Zaloguj'.

Rysunek 29 - Logowanie, aplikacja WWW

W razie wpisania błędnych danych wyświetlany jest komunikat widoczny na tym samym rysunku.

6.2.2 Ekran główny

Po pomyślnym przejściu procesu uwierzytelniania oraz autoryzacji użytkownika, zostaje on przeniesiony do widoku pulpitu (pulpity są opisane w rozdziale 6.2.4), odpowiadającego jego roli. W systemie wyróżniamy cztery role, do jakich mogą być przypisani użytkownicy. W kolejności od posiadającej najmniejsze uprawnienia są to: przedstawiciel handlowy, przedstawiciel regionalny, dyrektor handlowy oraz administrator. Opcje aplikacji dostępne są w zależności od posiadanej przez użytkownika roli.

²⁶ VPN - <http://pl.wikipedia.org/wiki/VPN>

SFAsystem

system wsparcia sprzedaży

admin [wyloguj]

Produkty

Sprzedaż

Klienci

Pracownicy

Raporty

Konfiguracja

System

Moje konto

Home > Produkty > Produkty

15 października 2009 15:19

Lista produktów

Kod produktu:

Nazwa produktu:

Grupa produktów: Wszystkie

Producent: Wszyscy

Filtruj

Usuń filtr

Dodaj nowy produkt

Kod Produktu	Nazwa	Grupa	Opis	Cena netto	VAT	Cena brutto	Edycja
LST62832N-70LL	LST62832N-70LL SMD	Pamięci statyczne SRAM	CMOS SRAM 32k*8 70ns SO28	4,57 zł	22%	5,57 zł	Edycja
UM62256-70LL	UM62256-70LL DIP28	Pamięci statyczne SRAM	CMOS SRAM 32k*8 70ns DIP28	5,76 zł	22%	7,03 zł	Edycja
BS62LV1027SIP55	BS62LV1027SIP55 SMD	Pamięci statyczne SRAM	Układ scalony CMOS stat. RAM 128k*8 55ns LL power SO32 BSI	6,10 zł	22%	7,44 zł	Edycja
M27C256B-90F6	M27C256B-90F6 DIP28	Pamięci typu EPROM	CMOS EPROM 32k*8 90ns DIP28	15,60 zł	22%	19,03 zł	Edycja
M27C512-90F6	M27C512-90F6 DIP28	Pamięci typu EPROM	CMOS EPROM 64k*8 90ns DIP28	15,60 zł	22%	19,03 zł	Edycja
AM2S-2415SZ	AM2S-2415SZ SIP4	Przetwornice DC/DC o mocy <=2W	Przetwornica DC/DC 2W 24V->15V, Uisol=1kV, SIP4, -40 +85st.C RoHS	24,59 zł	22%	30,00 zł	Edycja
AM1D-0505SH30Z	AM1D-0505SH30Z SIP7	Przetwornice DC/DC o mocy <=2W	Przetwornica DC/DC 1W 5V-> 5V, Uisol=3kV, SIP7, -40 +85st.C RoHS	14,75 zł	22%	18,00 zł	Edycja
AM3T-1205SZ	AM3T-1205SZ DIP24	Przetwornice DC/DC o mocy 2>P>=5W	Przetwornica DC/DC 3W Wide Input 9-18V->5V, Uisol=1,5kV, DIP24, -40 +85st.C RoHS	44,26 zł	22%	54,00 zł	Edycja
AM3TW-2405SZ	AM3TW-2405SZ DIP24	Przetwornice DC/DC o mocy 2>P>=5W	Przetwornica DC/DC 3W 4:1 Extra Wide Input 9-36V->5V, Uisol=1,5kV, DIP24, -40 +85st.C RoHS	65,76 zł	22%	80,23 zł	Edycja
LM324D	LM324D SMD	Wzmacniacze operacyjne	4* Wzmacniacz operacyjny Uin=3-30V 0 +70st.C SO14	0,41 zł	22%	0,50 zł	Edycja

1 2 3 4

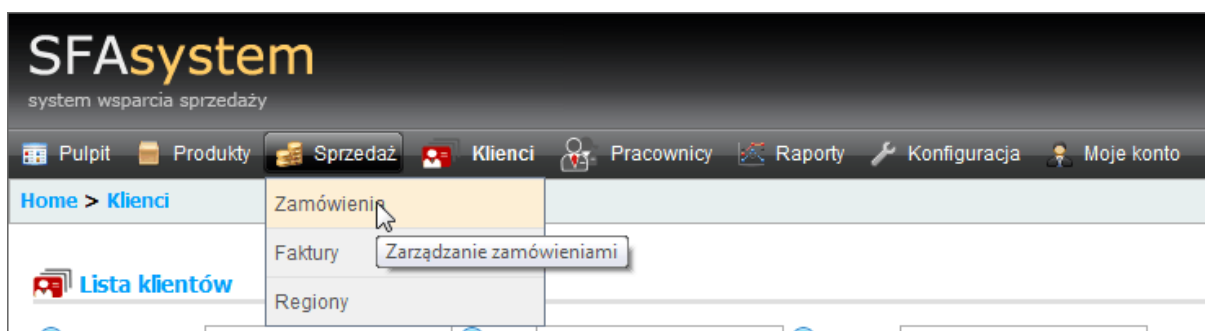
Rysunek 30 - Wygląd aplikacji WWW

Na rysunku numer 30 przedstawiono zrzut ekranu wyglądu aplikacji, w tym przypadku widoczna jest akurat lista produktów. Pomarańczowym prostokątem zaznaczono główna część treści strony.

Opis widoku aplikacji:

- Nazwa użytkownika oraz przycisk służący do wylogowania.
- Rozwijane menu główne systemu. Po prawej stronie znajduje się aktualna data i godzina.
- Dostępne przyciski funkcyjne dla strony. W tym przypadku wyświetlana jest lista, dlatego też mamy opcję filtrowania, usunięcia filtrów i dodania produktu.
- Obszar zawierający filtry.
- Obszar wyświetlający wyniki filtrowania.
- Stronicowanie wyników.

6.2.3 Menu główne



Rysunek 31 - Menu główne

Menu główne aplikacji umożliwia wybór poszczególnych stron pozwalających przeglądać, wprowadzać czy też modyfikować dane. Poniżej zostanie zaprezentowana całościowa struktura menu z podziałem na grupy.

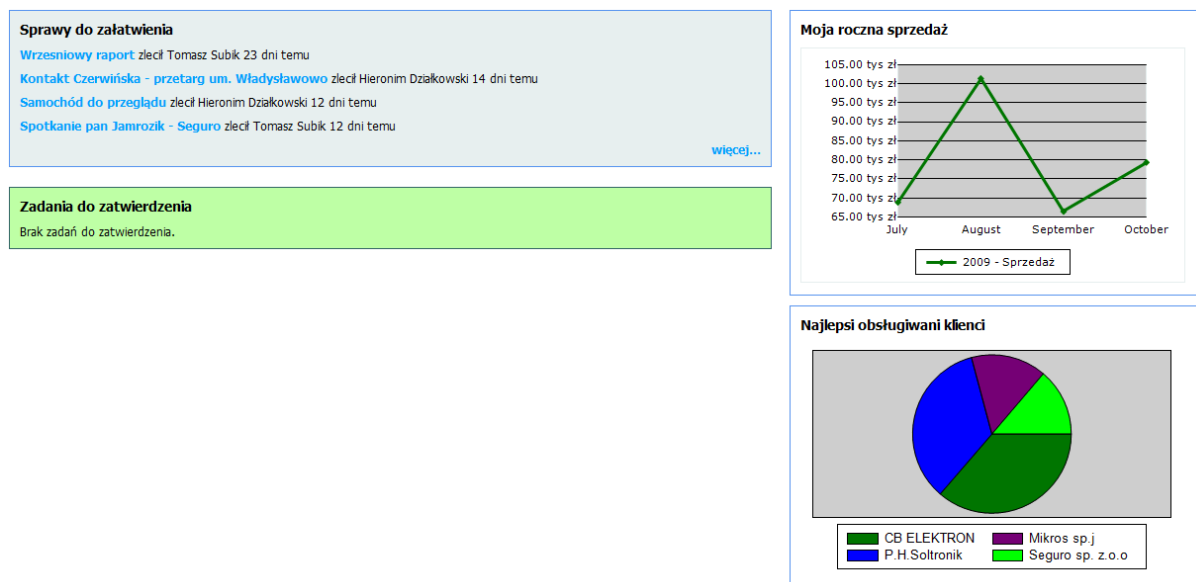
- Pulpit
- Produkty
 - Grupy produktów
 - Producenci
- Sprzedaż
 - Zamówienia
 - Faktury
 - Regiony
- Klienci
 - Placówki
 - Kontakty
- Pracownicy
 - Zadania pracowników
- Raporty
 - Zestawienie klientów
 - Zestawienie produktów
 - Najlepsze produkty
 - Sprzedaż pracownik
 - Najlepsi klienci
 - Najlepsi przedstawiciele handlowi
 - Sprzedaż grupy produktów
 - Sprzedaż regiony

- Konfiguracja
 - Ustawienia globalne
 - Słowniki

Nie każda z powyższych opcji jest dostępna dla każdego użytkownika. Widoczność danego węzła menu głównego jest zależna od posiadanej roli.

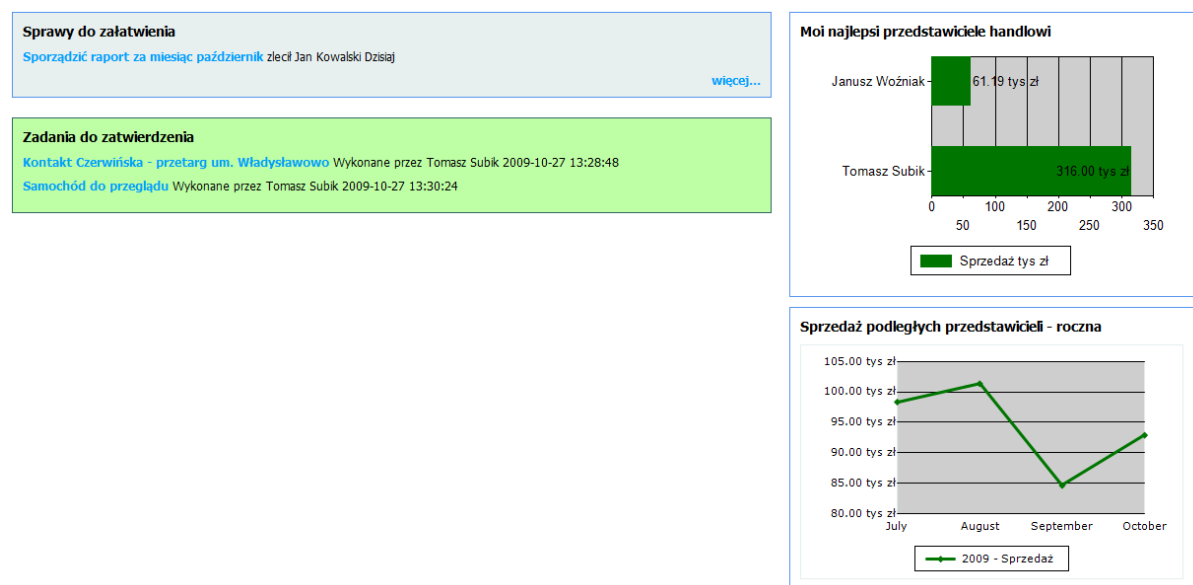
6.2.4 Pulpity

W celu ulepszenia interakcji oraz użyteczności aplikacji dla każdej z przewidzianych w systemie ról, został stworzony pulpit (ang. Dashboard). Jest to ekran powitalny, zawierający najważniejsze dla użytkownika informacje i szybkie odnośniki.



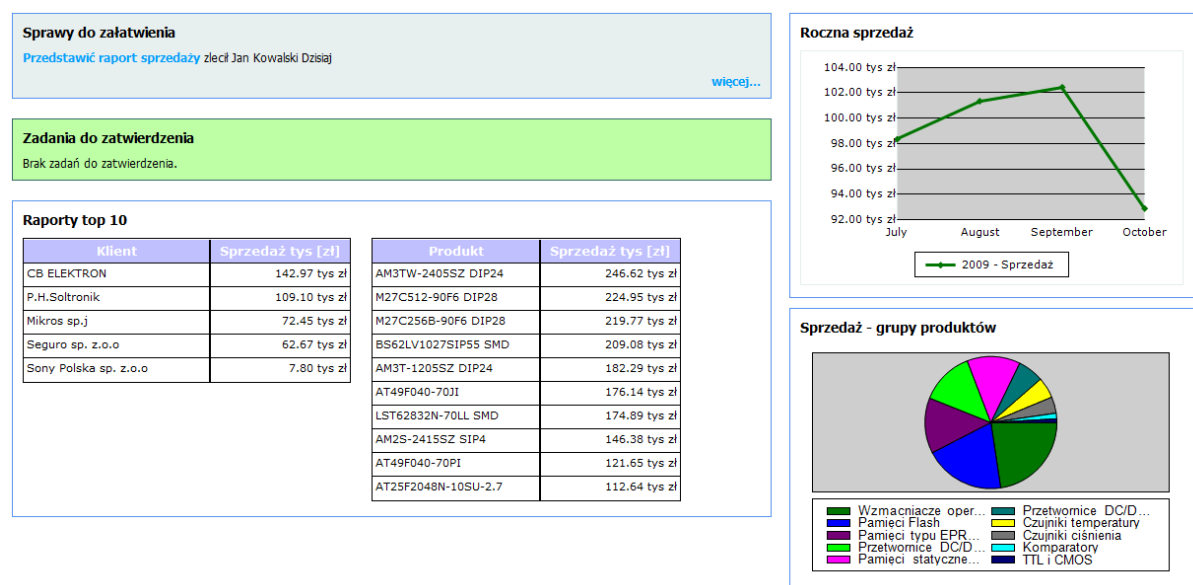
Rysunek 32 - Pulpit przedstawiciela handlowego

Na rysunku 32 zamieszczony został pulpit przedstawiciela handlowego. Informuje on reprezentanta firmy o ważnych sprawach do załatwienia, potrzebie zatwierdzenia zleconych przez niego zadań, czy też o własnych wynikach sprzedaży. Wykres liniowy znajdujący się po prawej stronie pokazuje generowanego przez pracownika obroty w danych miesiącach oraz roku.



Rysunek 33 - Pulpit przedstawiciela regionalnego

Na kolejnym rysunku, numer 33, pokazano natomiast pulpity przedstawicieli regionalnego. Prócz informacji wspólnych dla wszystkich prezentowanych pulpity, zawiera on informacje o obrotach, jakie generują poszczególni przedstawiciele handlowi podlegający pod przedstawiciela regionalnego.



Rysunek 34 - Pulpit dyrektora handlowego

Ostatni z pokazanych pulpity został stworzony dla kierownika najwyższego szczebla, którym jest dyrektor handlowy. Zawiera on raporty najlepiej sprzedających się produktów, najlepszych klientów firmy, generowanej rocznej sprzedaży rozbitej na poszczególne miesiące oraz sprzedaży według grup produktów. Przedstawiony jest na rysunku 34.

6.2.5 Dodawanie, edycja oraz przeglądanie danych

Podstawowe czynności, takie jak przeglądanie, edycja czy też dodawanie danych, dla zachowania prostoty działania aplikacji wykonywane są w większości w podobny sposób.

Dodawanie danych na przykładzie produktu

Aby dodać produkt, należy w pierwszej kolejności z menu głównego (Rysunek 35) wybrać widoczną pozycję „Produkty”. Po wyświetleniu listy produktów w obszarze z opcjami (przyp. obszary opisane zostały na rysunku 30) naciskamy na przycisk „Dodaj nowy produkt”.

Rysunek 35 - Dodawanie danych przykład

Na rysunku 35 pokazano przykładową formę dodawania danych w tym przypadku produktów. Po uzupełnieniu pól należy przycisnąć przycisk „Zapisz” i jeżeli użytkownik zapomniiał o wypełnieniu obowiązkowego pola, zostanie on powiadomiony o tym fakcie stosownym komunikatem (Rysunek 36).

Rysunek 36 - Walidacja danych

Poprawne wprowadzenie produktu prowadzi do przeniesienia do strony edycji uprzednio dodanego elementu oraz pojawieniu się informacji o zakończonej sukcesem operacji (Rysunek 37).

The screenshot shows a web application interface. At the top, a green notification bar with a checkmark icon contains the text 'Dodano nowy produkt'. Below this, the main header area includes a breadcrumb 'Edycja produktu(Powrót do listy produktów)' and a blue 'Zapisz' button. The main content area has two tabs: 'Produkt - informacje' and 'Zdjęcia', with the latter being active. Under the 'Zdjęcia' tab, there is a section titled 'Dodaj nowe zdjęcie' containing a 'Wybierz zdjęcie:' label, a text input field, a 'Browse...' button, and an 'Upload' button.

Rysunek 37 - Okno edycji produktu

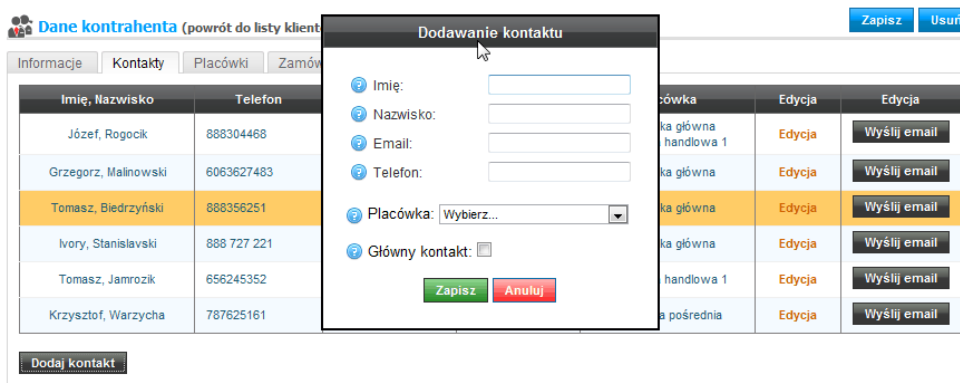
W powyższym przykładzie, oprócz wypełnianych wcześniej pól, produkt może posiadać także zdjęcia. Obrazek dotyczący elementu można dodawać z dysku lokalnego klikając przycisk „Browse...” a następnie „Upload”. Pomyślny wynik tej operacji jest przedstawiony na rysunku 38.

This screenshot shows the same 'Edycja produktu' window after a successful image upload. The notification bar now says 'Dodano zdjęcie'. The 'Zdjęcia' tab is active and displays a table with two columns: 'Zdjęcie' and 'Usuń'. The 'Zdjęcie' column contains a thumbnail of a circuit board, and the 'Usuń' column contains a 'Usuń' button. Below the table, the 'Dodaj nowe zdjęcie' section with its input fields and buttons remains visible.

Rysunek 38 - Produkt dodawanie zdjęcia

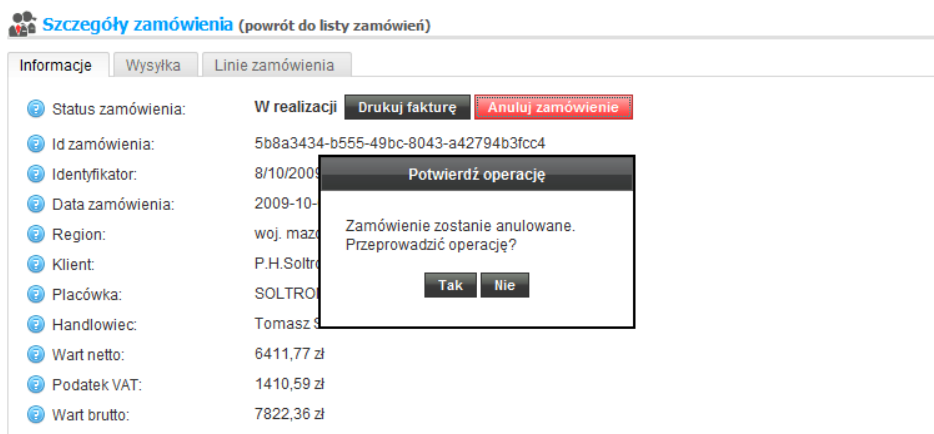
Dodawanie danych przy użyciu okna modalnego

Większość operacji dodawania danych przebiega w sposób podobny, jak w pokazanym wcześniej przykładzie. Jednakże, w aplikacji do przeprowadzenia tego typu czynności wykorzystano także okna modalne. Zastosowano je, aby w niektórych przypadkach przyspieszyć samo wprowadzanie informacji a także uczynić proces ten bardziej przyjazny użytkownikowi. Na rysunku 39 pokazano okno modalne, służące do wprowadzania informacji o nowym kontakcie firmy. Opcja ta jest dostępna w widoku szczegółów wybranego klienta w zakładce „Kontakty”.



Rysunek 39 – Okno modalne - wprowadzanie informacji o kontakcie klienta

Innym bardzo dobrym przykładem zastosowania okna modalnego w systemie są potwierdzenia. Na rysunku 40, pokazano zapytanie, czy przeprowadzić operację anulowania zamówienia.

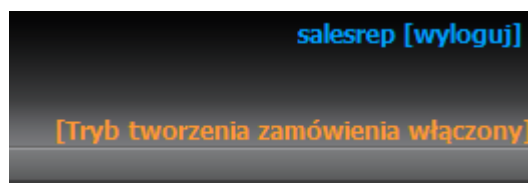


Rysunek 40 - Okno modalne - anulowanie zamówienia

6.2.6 Tryb tworzenia zamówienia

Proces tworzenia zamówienia został już opisany w rozdziale dotyczącym aplikacji mobilnej, aczkolwiek funkcjonalność ta jest także dostępna w tej części systemu. Ogólna zasada kompletowania zamówienia jest taka sama, czyli w jednym czasie, przez jednego użytkownika, może być tworzone tylko i wyłącznie jedno zamówienie. Jak wcześniej było wspomniane przypomina to proces dodawania produktów do koszyka w sklepie internetowym, lecz w tym wypadku klient nie ma dostępu do serwisu, a zamówienie musi wprowadzić pracownik.

Aby rozpocząć dodawanie zamówienia, należy przełączyć aplikację w „Tryb zamówienia” (jeżeli oczywiście nie zostało to zrobione wcześniej). Tryb ten załączamy przechodząc do strony „Zamówienia” i klikając na przycisk „Dodaj zamówienie” widoczny w obszarze opcji. Pomyślne włączenie trybu zamówienia sygnalizowane jest przez informację widoczną pod loginem użytkownika (rysunek 41), natomiast kliknięcie na nią przenosi nas do aktualnie kompletowanego zamówienia.



Rysunek 41 - Informacja "Tryb zamówienia"

Dodawanie produktów, wybór placówki handlowej dla zamówienia

Podczas kompletacji zamówienia, niezbędne jest szybkie i wygodne dodawanie produktów, dlatego też, jak w przypadku aplikacji mobilnej operacja, ta jest przeprowadzana w widoku listy produktów za pomocą dodatkowego przycisku „Dodaj do zamówienia”. Rysunek 42, przedstawia opisaną przed momentem operację. Użytkownik jest informowany o pomyślnym przebiegu procesu dodania produktu stosownym komunikatem.

✓ Produkt M27C2568-90F6 DIP28 dodany do zamówienia.

Lista produktów Filtruj Usuń filtr Dodaj nowy produkt

Kod produktu: Nazwa produktu: Grupa produktów: Producent:

Kod Produktu	Nazwa	Grupa	Opis	Cena netto	VAT	Cena brutto	Edycja	Zamówienie
LST62832N-70LL	LST62832N-70LL SMD	Pamięci statyczne SRAM	CMOS SRAM 32k*8 70ns SO28	4,57 zł	22%	5,57 zł	Edycja	Dodaj do zamówienia
UM62256-70LL	UM62256-70LL DIP28	Pamięci statyczne SRAM	CMOS SRAM 32k*8 70ns DIP28	5,76 zł	22%	7,03 zł	Edycja	Dodaj do zamówienia
BS62LV1027SIP55	BS62LV1027SIP55 SMD	Pamięci statyczne SRAM	Układ scalony CMOS stat. RAM 128k*8 55ns LL power SO32 BSI	6,10 zł	22%	7,44 zł	Edycja	Dodaj do zamówienia
M27C256B-90F6	M27C256B-90F6 DIP28	Pamięci typu EPROM	CMOS EPROM 32k*8 90ns DIP28	15,60 zł	22%	19,03 zł	Edycja	Dodaj do zamówienia
M27C512-90F6	M27C512-90F6 DIP28	Pamięci typu EPROM	CMOS EPROM 64k*8 90ns DIP28	15,60 zł	22%	19,03 zł	Edycja	Dodaj do zamówienia
AM2S-2415SZ	AM2S-2415SZ SIP4	Przetwornice DC/DC o mocy <=2W	Przetwornica DC/DC 2W 24V->15V, Uiso=1kV, SIP4, -40 +85st.C RoHS	24,59 zł	22%	30,00 zł	Edycja	Dodaj do zamówienia
AM1D-0505SH30Z	AM1D-0505SH30Z SIP7	Przetwornice DC/DC o mocy <=2W	Przetwornica DC/DC 1W 5V-> 5V, Uiso=3kV, SIP7, -40 +85st.C RoHS	14,75 zł	22%	18,00 zł	Edycja	Dodaj do zamówienia
AM3T-1205SZ	AM3T-1205SZ DIP24	Przetwornice DC/DC o mocy 2>P>5W	Przetwornica DC/DC 3W Wide Input 9-18V->5V, Uiso=1,5kV, DIP24, -40 +85st.C RoHS	44,26 zł	22%	54,00 zł	Edycja	Dodaj do zamówienia
AM3TW-2405SZ	AM3TW-2405SZ DIP24	Przetwornice DC/DC o mocy 2>P>5W	Przetwornica DC/DC 3W 4:1 Extra Wide Input 9-36V->5V, Uiso=1,5kV, DIP24, -40 +85st.C RoHS	65,76 zł	22%	80,23 zł	Edycja	Dodaj do zamówienia
LM324D	LM324D SMD	Wzmacniacze operacyjne	4* Wzmacniacz operacyjny Uin=3-30V 0 +70st.C SO14	0,41 zł	22%	0,50 zł	Edycja	Dodaj do zamówienia

Rysunek 42 - Proces dodawania produktów do zamówienia

W podobny sposób wygląda wybór placówki kontrahenta, dla którego tworzone jest zamówienie (Rysunek 43).

Lista obsługiwanych placówek

Filtruj

Usuń filtr

Dodaj nową placówkę

Nazwa:

Wybierz region sprzedaży: Wszystkie regiony

Wybierz klienta: Wszyscy klienci

Nazwa placówki	Należy do	Region	Miasto	Edycja	Zamówienie
Placówka główna	Seguro sp. z o.o	woj. śląskie	Gliwice	Edycja	Dodaj do zamówienia
Placówka handlowa 1	Seguro sp. z o.o	woj. śląskie	Gliwice	Edycja	Dodaj do zamówienia

Rysunek 43 - Wybieranie placówki dla zamówienia

Proces kompletacji zamówienia – aktualizacja, usuwanie pozycji, akceptacja

Tworzenie zamówienia

Dodaj pozycję

Złóż zamówienie

Kasuj zamówienie

Zamówienie dla: Placówka główna - firma Seguro sp. z o.o

Wybierz...

Komentarz:

Zapisz komentarz

Pozycje zamówienia

Lp.	Produkt	Ilość	Cena netto	Wartość netto	VAT [%]	Kwota VAT	Wartość brutto	Opcje
1	LST62832N-70LL SMD	166	4,57 zł	757,89 zł	22 %	166,73 zł	924,62 zł	Usuń
2	M27C512-90F6 DIP28	100	15,60 zł	1559,84 zł	22 %	343,16 zł	1903,00 zł	Usuń
3	AM2S-2415SZ SIP4	44	24,59 zł	1081,97 zł	22 %	238,03 zł	1320,00 zł	Usuń
4	AM1D-0505SH30Z SIP7	50	14,75 zł	737,70 zł	22 %	162,30 zł	900,00 zł	Usuń
5	AM3TW-2405SZ DIP24	100	65,76 zł	6576,23 zł	22 %	1446,77 zł	8023,00 zł	Usuń
6	TL074CN	100	0,58 zł	58,20 zł	22 %	12,80 zł	71,00 zł	Usuń
7	TL072CN	100	0,53 zł	53,28 zł	22 %	11,72 zł	65,00 zł	Usuń
8	TL064CD SMD	100	0,59 zł	59,02 zł	22 %	12,98 zł	72,00 zł	Usuń

Aktualizuj pozycje

Wartość netto: 10884,13 zł

Wartość podatku: 2394,49 zł

Wartość brutto: 13278,62 zł

Rysunek 44 - Ekran kompletowania zamówienia

Na Rysunku, numer 44, znajduje się główna część treści strony ekranu kompletacji zamówienia. Zaczynając od góry widoczna jest nazwa placówki, dla której tworzone jest zamówienie oraz przycisk kierujący użytkownika do okna z listą placówek. Poniżej tych elementów znajduje się pole

komentarza do zamówienia, którego treść zostaje zapamiętana po wybraniu przycisku „Zapisz komentarz”. Poniżej komentarza wylistowane są aktualne pozycje zamówienia, które można usuwać używając przycisku „Usuń”. Pole tekstowe w kolumnie „Ilość” służy do zmiany ilości danego produktu na zamówieniu, pole to nie akceptuje znaków innych niż cyfry. Aby zmiany ilości zostały zapamiętane należy wcisnąć przycisk „Aktualizuj pozycje”, wtedy zarówno w liście pozycji zamówienia jak i w podsumowaniu znajdującym się po prawej stronie poniżej zostaną uaktualnione. Po zakończonym procesie kompletacji, zamówienie potwierdzamy przyciskiem „Złóż zamówienie”, możemy je także wycofać używając „Kasuj Zamówienie”. Wybranie jakiegokolwiek z tych opcji wyłącza tryb zamówienia oraz przenosi użytkownika do strony z listą zamówień, informując o dokonanej operacji.

Realizacja zamówienia

Nowo dodane zamówienie otrzymuje w systemie status „Nowe” i czeka na potwierdzenie, które oznacza rozpoczęcie procesu realizacji. Aby zmienić status zamówienia na „W realizacji”, należy przejść do szczegółów zamówienia wybierając je z listy, a następnie kliknąć na zielony przycisk „Zatwierdź zamówienie”. Po dokonaniu tej operacji status zostaje zmieniony, a użytkownik od tego momentu ma możliwość wystawienia faktury VAT (Rysunek 45). Wysyłane jest także powiadomienie na adres email placówki (jeżeli został wprowadzony) o aktualnym statusie zamówienia (Rysunek 46).



Status zamówienia:	W realizacji	Wystaw fakturę	Anuluj zamówienie
Id zamówienia:	b50d0007-53e9-4f9f-9f89-9ca400de47e6		
Identyfikator:	16/10/2009_wo_TS_1		
Data zamówienia:	2009-10-16 13:29:17		
Region:	woj. śląskie		
Klient:	Seguro sp. z o.o		
Placówka:	Placówka główna		
Handlowiec:	Tomasz Subik		
Wart. Handlowiec:	10884,13 zł		
Podatek VAT:	2394,49 zł		
Wart brutto:	13278,62 zł		

Rysunek 45 - Zatwierdzenie zamówienia

Ta wiadomość została wygenerowana automatycznie. Prosimy nie odpowiadać.

Państwa zamówienie z dnia 14-10-2009 złożone na adres: Zwycięstwa 11/6, 44-100 Gliwice, Polska, na kwotę całkowitą 13278,62 zł jest obecnie realizowane. O wysyłce powiadomimy Państwa osobnym e-mailem.


Dziękujemy,
Firma Demonstracyjna
Z siedzibą
ul. Testowa 12A
44-100 Testowo, Polska


Rysunek 46 - Email realizacja zamówienia

Podobnym emailiem jak przy realizacji zamówienia użytkownik jest informowany podczas zmiany statusu zamówienia na „Wysłane”. Zmiany tej dokonuje się w zakładce „Wysyłka” szczegółów zamówienia za pomocą dostępnego tam przycisku.

Faktury VAT

Faktura może być wystawiona tylko do zatwierdzonego już zamówienia. Otrzymuje ona unikalny identyfikator informujący o roku wystawienia oraz numerze faktury w danym roku (np. FV 6/2009). Lista faktur została uwzględniona na rysunku 47.

 **Faktury** [Filtruj](#) [Usuń filtr](#)

 Pokaż: Wszystkie

Numer	Data wystawienia	Klient	Zamówienie	Data zapłaty	Czy opłacona	Opcje	Szczegóły
FV 6/2009	2009-10-16 13:52:09	Seguro sp. z o.o	16/10/2009_wo_TS_1		False	Zapłacono Wydruk	Szczegóły zamówienia
FV 5/2009	2009-10-14 11:32:15	CB ELEKTRON	10/10/2009_wo_TS_8		False	Zapłacono Wydruk	Szczegóły zamówienia
FV 4/2009	2009-10-08 12:56:33	P.H. Soltronik	8/10/2009_wo_TS_5		False	Zapłacono Wydruk	Szczegóły zamówienia
FV 3/2009	2009-10-08 12:34:15	Seguro sp. z o.o	8/10/2009_wo_JW_1		False	Zapłacono Wydruk	Szczegóły zamówienia
FV 2/2009	2009-10-08 10:35:37	CB ELEKTRON	8/10/2009_wo_TS_3		False	Zapłacono Wydruk	Szczegóły zamówienia
FV 1/2009	2009-10-08 10:16:22	Seguro sp. z o.o	8/10/2009_wo_TS_1		False	Zapłacono Wydruk	Szczegóły zamówienia

Rysunek 47 - Lista faktur

Każdą fakturę można wydrukować z poziomu aplikacji, do czynności tej należy użyć przycisku „Wydruk” (Wydruk faktury pokazany jest na rysunku 48).

polecenie firmy	Faktura VAT		FV 6/2009					
	2009-10-16 data wystawienia		ORYGINAL/KOPIA 2009-10-16 data sprzedaży					
Sprzedawca: Test Sp. z o.o. Adres: ul. Testowa 100, 02-222 Testowa NIP: 734-111-11-11			Nabywca: Seguro sp. z o.o. Adres: ul. Zwykła 11/6, 44-100 Gliwice, Polska NIP: 334-322-34-33					
Forma płatności: Przelew Termin płatności: 2009-10-30 Bank: Konto: 11 1111 1111 1111 1111 1111								
Lp.	Nazwa produktu	Ilość	J.M.	Cena netto	Wartość Netto	VAT [%]	Kwota VAT	Wartość Brutto
1	LST62832N-70LL SMD	166	szt	4,57 zł	757,89 zł	22%	166,73 zł	924,62 zł
2	M27C512-90F6 DIP28	100	szt	15,60 zł	1559,84 zł	22%	343,16 zł	1903,00 zł
3	AM25-2415SZ SIP4	44	szt	24,59 zł	1081,97 zł	22%	238,03 zł	1320,00 zł
4	AM1D-0505SH30Z SIP7	50	szt	14,75 zł	737,70 zł	22%	162,30 zł	900,00 zł
5	AM3TW-2405SZ DIP24	100	szt	65,76 zł	6576,23 zł	22%	1446,77 zł	8023,00 zł
6	TL064CD SMD	100	szt	0,59 zł	59,02 zł	22%	12,98 zł	72,00 zł
7	TL072CN	100	szt	0,53 zł	53,28 zł	22%	11,72 zł	65,00 zł
8	TL074CN	100	szt	0,58 zł	58,20 zł	22%	12,80 zł	71,00 zł
Razem					10884,13 zł	X	2394,49 zł	13278,62 zł
Razem do zapłaty: 13278,62 zł								
Kwota słownie: trzynaście tysięcy dwieście siedemdziesiąt osiem zł 62/100								
imie, nazwisko i podpis osoby upoważnionej do odświadczenia dokumentu					imie, nazwisko i podpis osoby upoważnionej do wystawienia dokumentu			

Rysunek 48 - Wydruk faktury VAT

Po wystawieniu oraz wysłaniu faktury, klient musi opłacić ją do ustalonego terminu, jeżeli wpłata wpłynie pracownik ma możliwość ustawienia statusu płatności dokumentu na „zapłacone”.

Raporty

Cześć raportująca jest ważnym elementem każdego systemu wspomagania sprzedaży, dlatego też nie mogło jej zabraknąć w wykonywanej aplikacji. Moduł ten dostępny z poziomu aplikacji internetowej mogą wykorzystywać tylko osoby z kierownictwa firmy (posiadające uprawnienia systemowe *SalesManager* oraz *SalesDirector*). W dalszej części rozdziału zwięźle opisane zostaną wszystkie dostępne raporty oraz przedstawiony przykład wygenerowanego raportu.

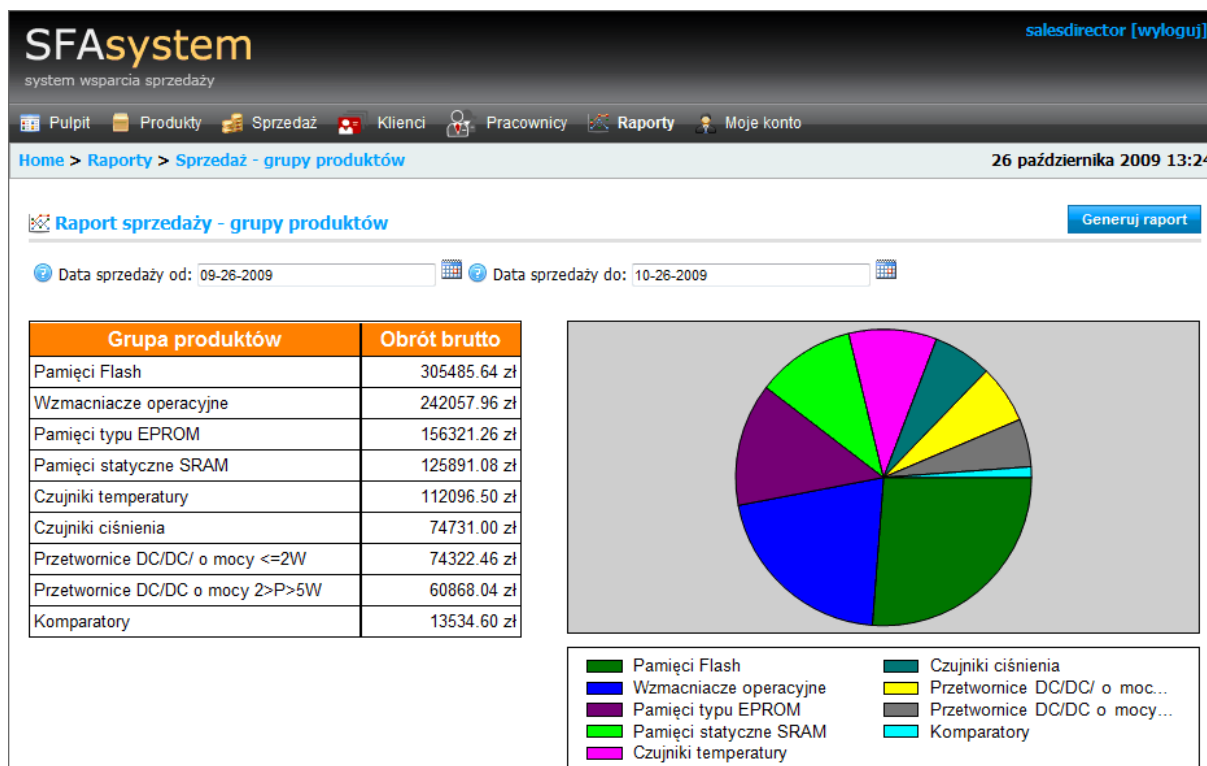
Raporty systemu wsparcia sprzedaży:

- Zestawienie klientów – lista wszystkich klientów firmy uwzględniająca najważniejsze dane (dostęp *SalesManager*, *SalesDirector*).
- Zestawienie produktów – lista wszystkich produktów oferowanych przez firmę (dostęp *SalesManager*, *SalesDirector*).

- Najlepsze produkty – Top 10 sprzedawanych produktów w wybranym regionie oraz okresie sprzedaży (dostęp *SalesDirector*).
- Sprzedaż pracownik – Wyniki sprzedaży w wybranym roku dla wybranego pracownika (dostęp *SalesManager, SalesDirector*).
- Najlepsi klienci – Zestawienie klientów generujących największy obrót brutto dla firmy w wybranym okresie (dostęp *SalesDirector*).
- Najlepsi przedstawiciele handlowi – Top 10 przedstawicieli handlowych w wybranym okresie sprzedaży (dostęp *SalesDirector*).
- Sprzedaż grupy produktów – Raport generowanych dochodów przy sprzedaży produktów z określonych grup w wybranym okresie (dostęp *SalesDirector*).
- Sprzedaż regiony – Raport generowanych dochodów w regionach w wybranym okresie sprzedaży (dostęp *SalesDirector*).

Przykładowy raport – Sprzedaż według grup produktów

Rysunek 49 przedstawia wygląd przykładowego raportu wygenerowanego przez aplikację. Wybrany raport pokazuje rozłożenie generowanego przychodu na poszczególne grupy produktów w wybranym okresie.



Rysunek 49 - Raport sprzedaży - grupy produktów

7. Specyfikacja wewnętrzna

Rozdział ten opisuje wybrane zagadnienia/technologie użyte podczas implementacji systemu oraz sposób ich zastosowania. Obie aplikacje zostały stworzone przy pomocy języka C# oraz najnowszej obecnie dostępnej wersji platformy .NET czyli 3.5. Dodatkowo podczas wykonywania programów posłużono się takimi technologiami jak Microsoft AJAX, NHibernate czy Log4Net. Wymienione projekty zostaną opisane szerzej w dalszej części rozdziału.

7.1 Technologia Microsoft ASP .NET AJAX

W celu zwiększenia interakcji oraz atrakcyjności części internetowej aplikacji została użyta technologia Microsoft ASP .NET AJAX. Termin ten określa zbiór rozszerzeń dla technologii tworzenia stron ASP .NET implementujących funkcjonalność technik zwanych Ajax. AJAX (Asynchronous Javascript and XML) jest to grupa powiązanych technik tworzenia stron WWW używanych w celu tworzenia bardziej interaktywnych aplikacji. Przy zastosowaniu AJAX'a aplikacja internetowa może pobierać dane z serwera w sposób asynchroniczny, bez uciążliwych dla klienta przeładowań całej strony. Przyspiesza to znacząco działanie aplikacji oraz zwiększa jej użyteczność. Microsoft ASP .NET AJAX jak było wspomniane kilka linijek wyżej, jest to rozszerzenie tej technologii na platformę .NET. Głównym celem implementacji tego dodatku było ułatwienie pracy programistów, którzy chcieli zastosować technologię AJAX w swoich rozwiązaniach. Poprzez wspólny projekt społeczności programistów oraz firmy Microsoft zwany „AJAX Control Toolkit”, możliwe jest korzystanie z dobrodziejstw technologii AJAX opierającej się na języku Javascript bez użycia tegoż języka. „AJAX Control Toolkit” jest to zbiór kontrolerek gotowych do użycia w technologii ASP .NET, które w znaczący sposób uatrakcyjnią użyteczność tworzonego rozwiązania. W celu lepszego zaznajomienia się z technologią Microsoft AJAX opisane zostaną najważniejsze kontrolki oraz pokazany sposób wykorzystania tej technologii w systemie.

Kontrolka *ScriptManager*

Każda z aplikacji pragnących wykorzystać możliwości techniki AJAX musi zostać wyposażona w jeden egzemplarz kontrolki *ScriptManager*. Kontrolka ta jest centralnym elementem pakietu wykonuje takie zadania jak np. koordynowanie oraz rejestrowanie skryptów javascript, zarządzanie innymi elementami z pakietu, interakcje skryptów javascript z metodami Web Service. Ważną właściwością tej kontrolki jest „EnablePartialRendering”, która domyślnie jest ustawiona na wartość „true”. Właściwość ta definiuje, czy częściowe odświeżanie strony ma zostać użyte w aplikacji.

Kolejną niezbędną kontrolką jest `UpdatePanel`, której przynajmniej jeden egzemplarz musi być użyty w aplikacji. Kontrolka ta odpowiada za asynchroniczne aktualizowanie niezależnych obszaru strony oraz zapewnia „ajaksowy” wygląd i działanie aplikacji. `UpdatePanel` jest pewnego rodzaju kontenerem elementów podlegających pod system dynamicznego odświeżania strony. Tworząc aplikację internetową korzystającą z technologii MS AJAX, jest możliwość wykorzystania kilku, a nawet kilkunastu kontroltek `UpdatePanel`. Dzięki temu można zdefiniować kilka niezależnych od siebie regionów odświeżanych w razie potrzeby.

Wykorzystanie technologii AJAX w aplikacji – Auto-uzupełnianie

Pierwszym przykładem zastosowania technologii AJAX oraz zbioru kontroltek Ajax Control Toolkit jest funkcjonalność pokazująca auto-podpowiedzi przeglądarki podczas wpisywania tekstu w polu tekstowym.

```
<asp:TextBox ID="txtCity" runat="server"></asp:TextBox>
<ajaxToolkit:AutoCompleteExtender ID="autoCompleteExt"
TargetControlID="txtCity" runat="server"
ServicePath="~/Administration/WebServices/AutoCompleteCityService.asmx"
CompletionInterval="100" ServiceMethod="GetCitiesList"
MinimumPrefixLength="2" CompletionSetCount="10" />
```

Listing 1 - Autouzupełnianie Web Forms

Listing 1 przedstawia sposób dodania kontrolki `AutoCompleteExtender` uzupełniającej kontrolkę `TextBox` o funkcjonalność auto-podpowiedzi. Krótki opis właściwości:

- `ServicePath` – ścieżka do usługi sieciowej zwracającej odpowiedź.
- `CompletionInterval` – czas, po którym auto-uzupełnianie jest aktywowane (w milisekundach).
- `ServiceMethod` – metoda usługi sieciowej zwracająca odpowiedź.
- `MinimumPrefixLength` – liczba znaków, od której uzupełnianie jest aktywne.
- `CompletionSetCount` – liczba odpowiedzi.

Do zrozumienia w całości działania tej kontrolki na listingu 2 przedstawiono kod funkcji usługi sieciowej, którą wykorzystano do zwracania odpowiedzi, podczas wprowadzania miasta (Listing 1, parametr `ServiceMethod`).

```
[System.Web.Services.WebMethod]
[System.Web.Script.Services.ScriptMethod]
public string[] GetCitiesList(string prefixText, int count)
```

```

{
    IList<City> cities = CityService.GetCitiesLikeName(prefixText, count);
    List<string> result = new List<string>();
    foreach (City city in cities)
        result.Add(city.Name);
    return result.ToArray();
}

```

Listing 2 - Funkcja zwracanie miast o określonym prefiksie

Przedstawiona funkcja, aby mogła być wykorzystana razem z kontrolką `AutoCompleteExtender` musi spełniać kilka wymogów:

- posiadać argumenty (`string`, `int`), pierwszy przekazuje tekst wpisany w kontrolce, drugi wartość właściwości `CompletionSetCount`,
- zwracać tablicę typu `string`,
- być oznaczoną jako `[System.Web.Script.Services.ScriptMethod]` – każda funkcja wywoływana z poziomu Javascript, musi posiadać takie oznaczenie.

7.2 Komponenty użytkownika

Technologia ASP.NET pozwala programiście nie tylko na używanie wbudowanych kontrolek, lecz także na implementację własnych. Kontrolki użytkownika mogą się służyć pomocą wszędzie tam, gdzie narzuca się fakt zamknięcia pewnej funkcjonalności i udostępnienie metod niezbędnych do operowania na niej. Takim przykładem, może być wyświetlanie komunikatów czy też kontrolek służących do wybierania elementów niestandardowych kolekcji. Implementacje obu z tych przykładów zostaną opisane w niniejszym podrozdziale.

Kontrolka użytkownika - MessageBox

W celu zwiększenia atrakcyjności oraz użyteczności systemu została stworzona kontrolka informująca użytkownika o przebiegu poczynionych akcji, błędach i innych zdarzeniach. Na listingu 3 zaprezentowano kod kontrolki w ASP.NET. W kodzie można wyróżnić odnośnik `HyperLink`, po kliknięciu którego zamknięty zostaje panel z wiadomością (`Panel`) oraz element `Literal`, który zawiera treść wyświetlanego komunikatu.

```

<div class="container">
    <asp:Panel ID="MessageBox" runat="server">
        <asp:HyperLink runat="server" ID="CloseButton">
            <asp:Image ID="Image1" runat="server"
                ImageUrl="~/App_Themes/administration/images/messageBox/close.png"
                AlternateText="Kliknij aby zamknąć" />
        </asp:HyperLink>
        <p>
            <asp:Literal ID="litMessage" runat="server"></asp:Literal>
        </p>
    </asp:Panel>
</div>

```

```

        </p>
    </asp:Panel>
</div>

```

Listing 3 - Kod ASP, kontrolka MessageBox

Technologia ASP .NET pozwala na tworzenie własnych kontrolek jednak muszą one dziedziczyć po klasie `System.Web.UI.UserControl`. Metoda `Page_Load` pokazana w następnym fragmencie kodu wywoływana jest przy każdorazowym odświeżeniu strony. Zaimplementowano w niej dynamiczne dodanie do elementu `HyperLink`, krótkiego kodu javascript, którego funkcją jest ukrycie panelu z wiadomością, gdy użytkownik wybierze ten odnośnik.

```

public partial class MessageBoxControl : System.Web.UI.UserControl
{
    #region Load
    protected void Page_Load(object sender, EventArgs e)
    {
        if (ShowCloseButton)
            CloseButton.Attributes.Add("onclick", "document.getElementById('" +
                MessageBox.ClientID + "').style.display = 'none'");
    }
    #endregion
}

```

Listing 4 - MessageBoxControl - zdarzenie Page_Load

Definicją typu `enum` określono rodzaje wiadomości.

```

#region Enum
public enum MessageType
{
    Error = 1,
    Info = 2,
    Success = 3,
    Warning = 4,
    Validation = 5
}
#endregion

```

Listing 5 - MessageType

W celu wyświetlenia wiadomości, możemy wywołać albo funkcję `Show`, podając typ wiadomości oraz jej treść, albo funkcję dla określonego typu np. dla błędów `ShowError`.

```

#region Show methods
public void Show(MessageType messageType, string message)
{
    CloseButton.Visible = showCloseButton;
    litMessage.Text = message;

    MessageBox.CssClass = messageType.ToString().ToLower();
    this.Visible = true;
}
public void ShowError(string message)
{
    Show(MessageType.Error, message);
}
[...]
#endregion

```

Listing 6 - MessageBoxControl - wyświetlanie

Wygląd pola z treścią określany jest poprzez styl CSS (MessageBox.CssClass), np. wiadomość typu Error zdefiniowaną ma klasę o nazwie error.

```
.error {
    color: #D8000C;
    background-color: #FFBABA;
    background-image: url('images/MessageBox/error.png');
}
```

Listing 7 - Styl CSS dla klasy error

Kontrolka użytkownika – Wybór grupy produktów – aplikacja mobilna

Zarówno aplikacja WWW jak i ta działająca na urządzeniach mobilnych, wykorzystuje kontrolki użytkownika w celu stworzenia obiektów do wybierania danego elementu z listy elementów. Klasa `SelectProductGroup` jest przykładem implementacji takiej funkcjonalności. Dodawanie grup produktów odbywa się w sposób rekurencyjny poprzez wywołanie funkcji `BindData(Guid ForParentEntityID, string prefix)`, której argumentami są: identyfikator grupy nadrzędnej oraz wcięcie, obrazujące hierarchię. Dla każdej grupy zwracana jest lista podgrup, której nazwy elementów z prefiksem dodawane są do kontrolki `ComboBox`.

```
public partial class SelectProductGroup : UserControl
{
    //Wypełnij danymi
    public void BindData()
    {
        cbxPGroups.Items.Clear();
        cbxPGroups.Items.Add(new ComboBoxItem(this.EmptyItemText, Guid.Empty));
        BindData(Guid.Empty, "--");
    }
    //Wypełnij danymi rekurencyjnie
    private void BindData(Guid ForParentEntityID, string prefix)
    {
        //pobierz liste podgrup dla danej grupy
        IList<ProductGroup> productGroupList =
            ProductManager.GetSubProductGroups(ForParentEntityID);
        if (productGroupList != null && productGroupList.Count > 0)
        {
            //kazda podgrupe oraz jej podgrupy dodaj do listy
            for(int i=0; i<productGroupList.Count; i++)
            {
                ProductGroup productGroup = productGroupList[i];
                ComboBoxItem item = new ComboBoxItem(prefix + productGroup.Name,
                    productGroup.ProductGroupID);
                this.cbxPGroups.Items.Add(item);
                if (productGroup.ProductGroupID == this.selectedProductGroupID)
                    cbxPGroups.SelectedItem = item;
                //dodaj podgrupy grupy zwiakszajac jednocześnie wciecie
                BindData(productGroup.ProductGroupID, prefix + "--");
            }
        }
    }
}
```

Listing 8 - Kontrolka SelectProductGroup - bindowanie

W celu pobrania identyfikatora aktualnie wybranego elementu używa się właściwości `SelectedProductGroupId`.

```

//ID wybranej grupy produktów
private Guid selectedProductGroupId;
public Guid SelectedProductGroupId
{
    get
    {
        Guid result = Guid.Empty;
        try
        {
            ComboBoxItem item = (ComboBoxItem)this.cbxPGroups.SelectedItem;
            result = (Guid) item.Value;
        }
        catch { }

        return result;
    }
    set
    {
        this.selectedProductGroupId = value;
    }
}

```

Listing 9 - Kontrolka SelectProductGroup - pobranie zaznaczonego id

Można także zmienić domyślny tekst dla niewybranego elementu np. na „Wybierz grupę produktów...” przy pomocy właściwości `EmptyItemText`.

```

//Tekst, jaki przyjmuje kontrolka jeśli nie wybrano, żadnej z wartości
private string emptyItemText = "[ --- ]";
public string EmptyItemText
{
    get
    {
        return emptyItemText;
    }
    set { emptyItemText = value; }
}

```

Listing 10 - Kod kontrolka - wybieranie grupy produktów

7.3 Obsługa zakładek w widoku szczegółów – aplikacja mobilna

Podczas prezentacji aplikacji mobilnej pokazano jak, detale klienta, czy jego placówki wyświetlane są przy użyciu przydatnej w środowisku urządzeń przenośnych kontrolki `TabControl` (Rysunek 50).



Rysunek 50 - Zakładki kontrolki TabControl

Kontrolka ta, pozwala w wygodny sposób, przełączać się pomiędzy zdefiniowanymi widokami. Funkcja `tabCustomer_SelectedIndexChanged` zostaje wywołana podczas zmiany aktualnej

zakładki. W celu przyspieszenia działania aplikacji, pobieranie danych dla poszczególnych zakładek jest realizowane dopiero przy ich wyświetlaniu.

```
private void tabCustomer_SelectedIndexChanged(object sender, EventArgs e)
{
    switch (tabCustomer.SelectedIndex)
    {
        case 0:
            break;
        case 1:
            BindFacilityList();
            break;
        case 2:
            BindContactList();
            break;
        case 3:
            BindOrderList();
            break;
        case 4:
            BindInvoiceList();
            break;
    }
}
```

Listing 11 - Funkcja zmiany zakładki

Dla przykładu zamieszczono funkcję `BindContactList()`, odpowiedzialną za dynamiczne tworzenie oraz rozmieszczanie w obszarze strony, kontrolek ze szczegółami osób kontaktowych klienta. Sygnalizacja procesu bindowania danych zrealizowana jest poprzez wyświetlenie kursora „proszę czekać” (`Cursor.Current = Cursors.WaitCursor`). W podobny sposób uzupełniane są inne zakładki.

```
private void BindContactList()
{
    if (needBindContactList)
    {
        Cursor.Current = Cursors.WaitCursor;
        int yPos = 5;
        tabContacts.Controls.Clear();
        foreach (Contact contact in
            CustomerManager.GetCustomerContacts(customer.CustomerID))
        {
            ctrlCustomerContact control = new ctrlCustomerContact();
            control.BindData(contact);
            control.Top = yPos;
            control.Left = 5;
            control.Width = 220;
            control.Height = 60;
            tabContacts.Controls.Add(control);
            yPos += control.Height + 5;
        }
        Cursor.Current = Cursors.Default;
        needBindContactList = false;
    }
}
```

Listing 12 - Bindowanie listy kontaktów

7.4 Logowanie przy użyciu usługi sieciowej

W celu zmniejszenia ryzyka wycieku poufnych danych zaimplementowano mechanizm niepozwalający, nieautoryzowanym użytkownikom na przeprowadzanie synchronizacji danych. Wykonane jest to za pomocą usługi sieciowej (ang. Web service), dostępnej na serwerze głównym. Metoda logowania, należąca do tej usługi została przedstawiona na listingu 13, sprawdza czy podane dane są poprawne i jeżeli tak zwraca identyfikator pracownika.

```
[WebMethod]
public Guid LoginUser(string username, string password)
{
    if (EmployeeService.Login(username, password))
    {
        Employee emp = EmployeeService.GetEmployeeByLogin(username);
        if (emp.Roles.Contains(EmployeeService.GetRoleByName("SalesRep")))
        {
            return emp.Employeeid;
        }
    }
    return Guid.Empty;
}
```

Listing 13 - Usługa sieciowa – logowanie

Pierwszym, niezbędnym krokiem do wykorzystania usługi sieciowej w projekcie tworzonym przy pomocy pakietu Visual Studio, jest dodanie do niej referencji. Tworzona jest wtedy klasa, implementująca interfejs usługi dostępnej na serwerze (w podanym przykładzie `LoginService`). Użycie jej, musi być poprzedzone podaniem adresu url (`service.Url`), po czym można korzystać, ze zdefiniowanych na serwerze metod.

```
if (!SynchronizationManager.Instance.DatabaseExists)
{
    LoginService.LoginService service = new
    SFASystem.Pocket.App.LoginService.LoginService();
    service.Url = "http://192.168.1.1:58098/Web/WebServices/LoginService.asmx";
    Guid employeeID = service.LoginUser(username, password);
    if (employeeID != Guid.Empty)
    {
        DataManager.Instance.CurrentEmployeeID = employeeID;
        UIHelper.ShowInfo("Pomyślna autoryzacja. Przeprowadź synchronizację!");
        ApplicationManager.Instance.IsWebServiceAuth = true;
        if (WebAuthorizationOK != null)
            WebAuthorizationOK(null, null);
    }
    else
        UIHelper.ShowError("Błędny login lub hasło");
}
```

Listing 14 - Fragment kodu - logowanie poprzez web service

7.5 Implementacja warstwy dostępu do danych

System sprzedaży został zaimplementowany z zastosowaniem architektury wielowarstwowej. Zarówno dla aplikacji mobilnej, jak i internetowej można wydzielić warstwy, takie jak: dostępu do

danych, biznesową oraz prezentacji. Taki podział pewnych obowiązków w aplikacji, zwiększa skalowalność oraz wprowadza porządek do struktury kodu. Rozdział ten opisuje, krótko warstwę dostępu do danych aplikacji mobilnej oraz przedstawia trochę bardziej rozbudowany opis technologii NHibernate w odniesieniu do warstwy dostępu do danych aplikacji WWW.

Warstwa dostępu do danych aplikacja mobilna

Z racji braku implementacji projektu NHibernate dla urządzeń mobilnych, aby stworzyć warstwę dostępu do danych dla części przenośnej, należało bezpośrednio użyć technologii ADO .NET. Technologia ta umożliwia sposób dostępu do danych w bazie w dwojaki sposób. Pierwszy z nich jest sposobem bezpołączeniowym, czyli z wykorzystaniem klas `DataTable`, `DataSet`, `DataAdapter` i ładowaniem danych do pamięci aplikacji. Sposób ten jest bardzo wygodny przy użyciu wbudowanych w platformę kontrolerek oraz nie wymaga wielu połączeń z bazą danych, gdyż może ładować dane tylko raz. Sposób ten ma też swoje wady, takie jak duże zapotrzebowanie na pamięć, która w przypadku urządzeń mobilnych jest cennym zasobem, więc lepszym zastosowaniem będzie cieszył się drugi model pobierania danych, czyli połączeniowy. Dane w tym wypadku pobierane są przy użyciu klasy `DataReader`, która działa szybciej, gdyż wiersze zapytania zadanego bazie danych pobierane tak szybko, jak stają się dostępne (eliminuje to oczekiwanie na zwrócenie pełnego zasobu rekordów).

Listing numer 15, prezentuje fragment klasy `ProductDB`, należącej do warstwy dostępu do danych aplikacji dedykowanej dla urządzenia Pocket PC. Pokazuje on sposób pobierania z bazy danych produktu o określonym ID (funkcja `GetProductByID(Guid)`). Należy zwrócić uwagę na wykorzystanie trybu bezpołączeniowego przy dostępie do danych (obiekt klasy implementującej interfejs `IDataReader`) oraz sposób przypisywania wartości do poszczególnych pól klasy `Product` wykorzystujący funkcje pomocnicze przy rzutowaniu typu elementu zawartego pod wskazanym indeksem w obiekcie `dataReader`.

```
public static Product GetProductByID(Guid productID)
{
    Product product = null;
    if (productID == Guid.Empty)
        return product;

    string sqlQuery = "SELECT * FROM PRODUCT WHERE ProductID='" + productID +
        "'";
    return GetProductFromQuery(sqlQuery);
}

private static Product GetProductFromQuery(string sqlQuery)
{
    Product product = null;

    DbCommand dbCommand = DBHelper.GetDBCommand(sqlQuery);
```

```

        using (IDataReader dataReader = DBHelper.ExecuteReader(dbCommand))
        {
            while (dataReader.Read())
            {
                product = GetProductFromReader(dataReader);
            }
        }
        return product;
    }

    private static Product GetProductFromReader(IDataReader dataReader)
    {
        Product product = new Product();
        product.ProductID = DBHelper.GetGuid(dataReader, "ProductID");
        product.ProductGroupID = DBHelper.GetGuid(dataReader, "ProductGroupID");
        product.ShortDescription = DBHelper.GetString(dataReader, "ShortDescription");
        product.FullDescription = DBHelper.GetString(dataReader, "FullDescription");
        product.Price = DBHelper.GetDecimal(dataReader, "Price");
        product.TaxID = DBHelper.GetGuid(dataReader, "Tax");
        product.Name = DBHelper.GetString(dataReader, "Name");
        product.Code = DBHelper.GetString(dataReader, "Code");
        product.ManufacturerID = DBHelper.GetGuid(dataReader, "ManufacturerID");
        return product;
    }

```

Listing 15 - Warstwa dostępu do danych - Pocket PC

Operacja wstawiania nowych danych do bazy wykonywana jest przy użyciu obiektu klasy `DbCommand`, który reprezentuje określone zapytanie wraz z parametrami. Dla przykładu na listingu 16 zaprezentowano dodanie nowego zamówienia. W celu zwiększenia wygody przeprowadzania operacji na bazie danych stworzono klasę `DBHelper`, implementującą najważniejsze funkcje, takie jak np. tworzenie, wywoływanie komend, dodawanie parametrów czy konwersje typów.

```

public static Order InsertOrder(Order order)
{
    string sqlQuery = "INSERT INTO [Order] (OrderID, CustomerFacilityID, TerritoryID, EmployeeID, OrderStatus, Identifier, OrderDate, SubTotal, TaxAmount, Total, Comment) " +
        "VALUES(@OrderID, @CustomerFacilityID, @TerritoryID, @EmployeeID, @OrderStatus, @Identifier, @OrderDate, @SubTotal, @TaxAmount, @Total, @Comment)";
    if (order.OrderID == Guid.Empty)
        order.OrderID = Guid.NewGuid();
    else
        return null;

    DbCommand dbCommand = DBHelper.GetDBCommand(sqlQuery);
    DBHelper.AddInParameter(dbCommand, "OrderID", DbType.Guid, order.OrderID);
    DBHelper.AddInParameter(dbCommand, "CustomerFacilityID", DbType.Guid, order.CustomerFacilityID);
    DBHelper.AddInParameter(dbCommand, "TerritoryID", DbType.Guid, order.TerritoryID);
    DBHelper.AddInParameter(dbCommand, "EmployeeID", DbType.Guid, order.EmployeeID);
    DBHelper.AddInParameter(dbCommand, "OrderStatus", DbType.Guid, order.OrderStatusID);
    DBHelper.AddInParameter(dbCommand, "Identifier", DbType.String, order.Identifier);
    DBHelper.AddInParameter(dbCommand, "OrderDate", DbType.DateTime, order.OrderDate);
    DBHelper.AddInParameter(dbCommand, "SubTotal", DbType.Decimal, order.SubTotal);
    DBHelper.AddInParameter(dbCommand, "TaxAmount", DbType.Decimal, order.TaxAmount);
}

```

```

        DBHelper.AddInParameter(dbCommand, "Total", DbType.Decimal, order.Total);
        DBHelper.AddInParameter(dbCommand, "Comment", DbType.String, order.Comment);
        DBHelper.ExecuteNonQuery(dbCommand);

        return order;
    }

```

Listing 16 - Wstawienie danych do bazy, aplikacja mobilna

Proces modyfikacji przebiega w sposób podobny, zmienia się jedynie rodzaj zapytania zadawanego bazie danych.

Warstwa dostępu do danych aplikacja WWW

Dostęp do bazy danych stacjonarnej części systemu zaimplementowany został przy użyciu znanego narzędzia służącego do mapowania obiektowo-relacyjnego rozwijanego na licencji open-source. Mowa tu o wspomnianym, w rozdziale poświęconym doborowi technologii, projekcie NHibernate. Rozwiązanie to znacznie ułatwia tworzenie aplikacji bazodanowych, gdyż pozwala programiście w większym stopniu skupić się na rozwijaniu funkcjonalności biznesowej aplikacji, pozostawiając wiele kwestii środowisku, takich jak np. zarządzanie trwałością obiektów, pobieranie, zapisywanie danych czy też cache'owanie.

Rozpoczęcie pracy z technologią NHibernate wymaga utworzenia pliku/plików xml, zawierających opis dotyczący mapowania oraz klas mapowanych obiektów. Najlepszym rozwiązaniem jest utworzenie dla każdej z mapowanych tabel osobnego pliku. Pliki opisu mapowania zawierają rozszerzenie .hbm.xml oraz mogą być tworzone samodzielnie lub przy użyciu generatorów kodu, takich jak, np. MyGeneration. Poniżej przedstawiony zostanie przykładowy plik xml oraz kod klasy do której się on odnosi.

Mapowanie tabeli odbywa się poprzez podanie klasy C#, która będzie odzwierciedlać ją w kodzie aplikacji (element `class`). Niezbędny jest także opis klucza głównego (element `id`).

```

<?xml version="1.0" encoding="utf-8"?>
<hibernate-mapping xmlns="urn:nhibernate-mapping-2.2">
<!--Build: with lujan99@usa.net Nhibernate template-->
<!--nazwa klasy,przestrzeń nazw, nazwa tabeli oraz czy włączony lazy loading-->
    <class name="SFASystem.Domain.Customer,SFASystem.Domain" table="Customer"
        lazy="true">
        <!--opis kolumny klucza głównego tabeli -->
        <id name="Customerid" column="CustomerID" type="Guid">
            <generator class="guid.comb" /><!--Rodzaj generatora identyfikatorów-->
        </id>
    </class>
[...]
```

Listing 17 - plik mapowania xml, identyfikator

Dla podanego pliku xml musi być zadeklarowana klasa `Customer`.

```
[Serializable]
public class Customer{
    #region Member Variables
    protected Guid customerid;
    public virtual Guid Customerid
    {
        get { return customerid; }
        set { customerid = value; }
    }
    [...]
}
```

Listing 18 - Klasa Customer, identyfikator

Mapowanie pozwala także odzwierciedlić relacje zawarte pomiędzy poszczególnymi tabelami. Relacja „wiele do jednego”:

```
<!--Opis relacji wiele do jednego -->
<many-to-one name="Contact" column="ContactID" cascade="save-update" />
```

Listing 19 - Plik mapowania xml, relacja wiele do jednego

Implementowana jest jako obiekt klasy do której się odnosi:

```
protected Contact contact;
public virtual Contact Contact
{
    get { return contact; }
    set { contact = value; }
}
```

Natomiast relacja „jeden do wielu”:

```
<!--Opis relacji jeden do wielu, Klient posiada wiele placówek
lazy(okreslenie, czy kolekcja wypelniana od razu, czy dopiero przy
użyciu)-->
<bag name="CustomerFacility" inverse="true" lazy="true" cascade="delete">
    <key column="CustomerID" />
    <one-to-many class="SFASystem.Domain.CustomerFacility,SFASystem.Domain" />
</bag>
```

Listing 20 - plik mapowania xml, relacja jeden do wielu

Jako lista obiektów klasy, do której się odnosi:

```
protected IList<CustomerFacility> customerfacility;
public virtual IList<CustomerFacility> CustomerFacility
{
    get { return customerfacility; }
    set { customerfacility = value; }
}
```

Listing 21 - Klasa Customer, lista placówek

Pozostałe pola tabeli mapowane są w następujący sposób, dla pól:

```
<property name="FullDescription" column="FullDescription" type="string" not-
null="true" />
<property name="CustomerSince" column="CustomerSince" type="DateTime" />
```

Listing 22 - Plik mapowania xml, atrybuty

Klasa `Customer` zawiera:

```
protected string fullDescription;
public virtual string Name
```

```

{
    get { return name; }
    set { name = value; }
}
protected DateTime? customersince;
public virtual DateTime? CustomerSince
{
    get { return customersince; }
    set { customersince = value; }
}

```

Listing 23 - Klasa Customer, atrybuty

Plik xml oraz klasa mu odpowiadająca zostały utworzone przy użyciu generatora MyGeneration²⁷. Warto wspomnieć o ważnej, lecz często zapominanej przez programistów czynności, jaką jest ustawienie sposobu kompilacji plików xml na „Zasób wbudowany” (ang. Embedded Resource).

Aby uzyskać dostęp do danych w bazie używając projektu NHibernate, potrzebujemy obiekt klasy Session. W aplikacjach ASP .NET najlepszą praktyką jest używanie jednej sesji na każde żądanie wyświetlenia strony. Można to osiągnąć implementując własny moduł aplikacji IHttpModule. Dzięki takiej funkcjonalności możemy otwierać sesję na początku żądania wyświetlania strony, a zamykać ją wraz z zakończeniem przetwarzania bieżącego żądania (Listing 24).

```

public class NHibernateSessionModule : IHttpModule
{
    public void Init(HttpApplication context)
    {
        context.EndRequest += new System.EventHandler(context_EndRequest);
    }

    void context_EndRequest(object sender, System.EventArgs e)
    {
        HBManager.Instance.CloseSession();
    }
    public void Dispose()
    {
    }
}

```

Listing 24 - Moduł Http zamykanie sesji NHibernate

Omawiany moduł odpowiedzialny jest za zamykanie sesji po każdym żądaniu, o ile ta w ogóle jest otwarta, czy istnieje. Do zarządzania sesjami NHibernate została stworzona klasa HBManager, która w oparciu o wzorzec Singleton²⁸ ma za zadanie tworzenie sesji, zwracanie już istniejącej, zamykanie jej oraz zarządzanie transakcjami.

Mechanizm działania NHibernate opiera się w dużej mierze na metodach generycznych, więc dla aplikacji internetowej powstało bazowe repozytorium implementujące najważniejsze funkcje, takie jak pobieranie, zapisywanie i usuwanie obiektów danej klasy (Listing 25).

²⁷ MyGeneration portal - <http://www.mygenerationsoftware.com/portal/default.aspx>

²⁸ Singleton - [http://pl.wikipedia.org/wiki/Singleton %28wzorzec projektowy%29](http://pl.wikipedia.org/wiki/Singleton_%28wzorzec_projektowy%29)

```

public abstract class BaseRepository<T, TId> : IRepository<T, TId>
{
    public virtual T GetByID(TId id)
    {
        ISession session = HBManager.Instance.GetSession();
        return session.Get<T>(id);
    }

    public IQueryable<T> GetQueryable()
    {
        ISession session = HBManager.Instance.GetSession();
        return session.Linq<T>();
    }

    public virtual void SaveOrUpdate(T entity)
    {
        ISession session = HBManager.Instance.GetSession();
        using (ITransaction transaction = session.BeginTransaction())
        {
            session.SaveOrUpdate(entity);
            transaction.Commit();
        }
    }

    public virtual void Remove(T entity)
    {
        ISession session = HBManager.Instance.GetSession();
        using (ITransaction transaction = session.BeginTransaction())
        {
            session.Delete(entity);
            transaction.Commit();
        }
    }
}

```

Listing 25 - NHibernate repozytorium

Warto zwrócić uwagę, że zastosowanie klasy generycznej w znacznym stopniu przyspiesza implementację systemu. Kolejną rzeczą godną zainteresowania jest użycie nowego projektu stworzonego specjalnie dla NHibernate, jakim jest „Linq for NHibernate”. O wykorzystaniu tego projektu w kodzie świadczy linia `return session.Linq<T>();`. Zwraca ona obiekt implementujący interfejs `IQueryable`, dla którego zapytania możemy wykonywać za pomocą języka Linq. Dla przykładu funkcja `GetCustomers` (Listing 26), zwracająca klientów według zadanych kryteriów.

```

public IList<Customer> GetCustomers(string name, string nip, string region)
{
    ISession session = HBManager.Instance.GetSession();
    var query = from c in session.Linq<Customer>()
                where c.Name.Contains(name) &&
                     c.NIP.Contains(nip) && c.REGION.Contains(region)
                select c;
    return query.ToList();
}

```

Listing 26 - Linq to NHibernate przykład

Po wywołaniu opisanej funkcji w bazie danych zostanie następujące zapytanie.

```
SELECT *  
FROM [CUSTOMER] customer  
WHERE customer.Name LIKE @name AND customer.NIP LIKE @nip AND  
customer.REGON LIKE @regon
```

Listing 27 - Wygenerowane zapytanie SQL

7.5 Generowanie raportów – Reporting Services

Możliwość generowania raportów, jest jedną z najważniejszych funkcjonalności, które zawiera internetowa część systemu. W celu wdrożenia tej opcji, użyta została usługa zwana Reporting Services, będąca jednym z modułów bazodanowego oprogramowania Microsoft SQL Server. Usługa ta, jest opcjonalnym dodatkiem do serwera bazodanowego, więc należy zwrócić uwagę, czy jest dostępna, jeżeli nie to przeprowadzić instalację za pomocą standardowego instalatora SQL Server.

Konfiguracja Reporting Services

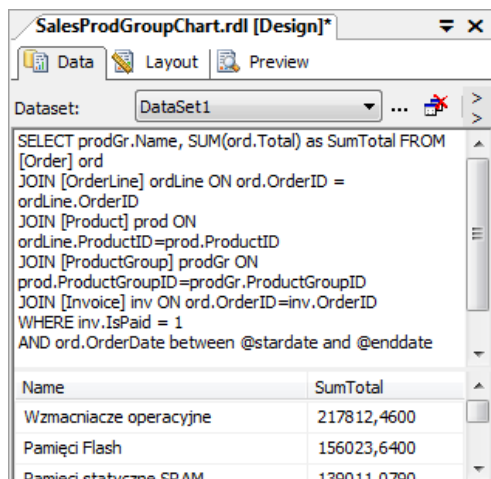
Korzystanie z serwera raportów, wymaga poprawnej jego konfiguracji. Proces ten został w przystępny sposób pokazany w materiale wideo dostępnym pod adresem <http://www.asp.net/learn/sql-videos/video-112.aspx>, jak i w wielu miejscach w Internecie, więc nie będzie on szczegółowo opisany w tej pracy. Warto jednak zwrócić uwagę na dwa elementy, pozwolą na sprawną konfigurację narzędzia. Pierwszym z nich jest konieczność posiadania serwera IIS, z zainstalowanymi wszystkimi niezbędnymi modułami (bez tego nie ma możliwości instalacji Reporting Services). Drugi dotyczy środowisk systemów Windows typu Vista lub Windows 7, w których należy zainstalować najnowsze dodatki (ang. Services Pack) do oprogramowania SQL Server. Poprawne zakończenie procesu konfiguracji, dla wybranej strony udostępnianej przez serwer IIS skutkuje utworzeniem dwóch katalogów wirtualnych. Pierwszy to punkt dostępowy aplikacji do plików z raportami na serwerze, drugi natomiast to oprogramowanie będące menadżerem raportów. Za pomocą tej aplikacji możemy przeglądać, tworzyć oraz usuwać raporty oraz zarządzać prawami dostępu do nich.

Tworzenie raportu oraz wykorzystanie w aplikacji

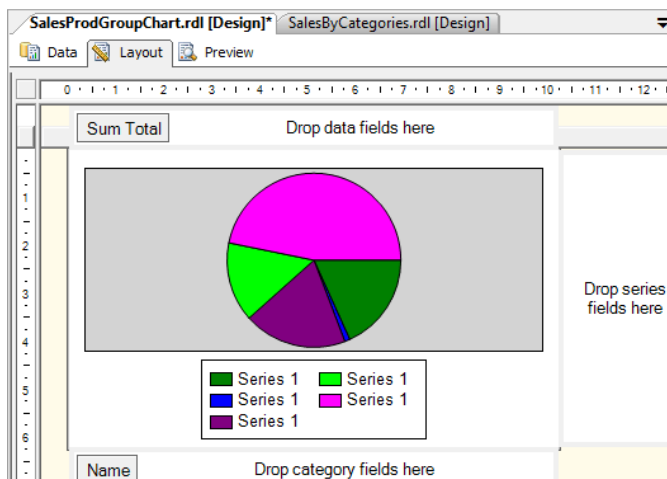
Najlepszym sposobem na dodanie raportu w standardzie Reporting Services jest utworzenie nowego projektu typu „Report Server Project” za pomocą oprogramowania SQL Server Business Intelligence Development Studio, dostępnego w pakiecie SQL Server. Następnie w opcjach projektu, należy

wypełnić adres URL, pod którym znajduje się uprzednio zainstalowany serwer raportów oraz wybrać nazwę katalogu, do którego będą kopiowane tworzone raporty.

Zamieszczony dalej opis prezentuje prosty raport będący wykresem kołowym wyników sprzedaży poszczególnych grup produktowych. Podgląd plików raportu .rdl w środowisku Business Intelligence zawiera trzy zakładki „Data”, „Layout” oraz „Preview”.



Rysunek 51 - Raport, zakładka Data



Rysunek 52 - Raport, zakładka Layout

Pierwsza z nich zawiera kod SQL zapytania zwracającego dane potrzebne do utworzenia raportu. W zakładce „Data”, która została zaprezentowana na rysunku 51, warto zwrócić uwagę na dwa parametry kodu SQL (@startdate oraz @enddate), wykorzystanie ich zostanie opisane w dalszej części tego podrozdziału. Druga z prezentowanych zakładek, czyli „Layout” (Rysunek 52) służy do graficznego rozmieszczenia wyświetlanych danych przy użyciu dostępnych kontroltek. Ostatnia zakładka „Preview” prezentuje stworzone zestawienie, wykresy czy tabelki wypełniając je danymi zwróconymi poprzez zapytanie. Gotowy raport/raporty umieszcza się na serwerze używając opcji „Deploy”, wszystkie elementy umieszczane są pod adresem:

[http://\[adres_serwera\]/\[nazwa_serwera_raportów\]/\[nazwa_folderu\]](http://[adres_serwera]/[nazwa_serwera_raportów]/[nazwa_folderu])

Dla wykonywanego systemu adres ten miał następującą postać:

<http://localhost/reportserver/SFASystemReports>

Od tej pory można ich użyć w implementowanej aplikacji internetowej.

W celu wyświetlenia określonego raportu w aplikacji internetowej, dodajemy do wybranej strony kontrolkę `ReportViewer`. Na listingu 12 zaprezentowano użycie tej kontrolki do wyświetlenia

zaprojektowanego raportu sprzedaży według grup produktów. Należy zwrócić uwagę na właściwość `ReportPath`, która jest określeniem ścieżki do raportu umieszczonego na wybranym serwerze.

```
<rsweb:ReportViewer ID="rvSales" runat="server" Font-Names="Verdana"
    Font-Size="8pt" Height="240px" ProcessingMode="Remote" BackColor="Red"
    ShowParameterPrompts="False" ShowToolBar="False" Width="400px">
    <ServerReport ReportPath="/SFASystemReports/SalesProdGroupChart" />
</rsweb:ReportViewer>
```

Listing 28 - Kontrolka Report Viewer

Tak dodana kontrolka, bez problemu wyświetla proste raporty. Jednakże, jeżeli wśród nich będzie taki, który posiada parametry (przykładzie @startdate, @enddate) niezbędne jest wcześniejsze ich zainicjowanie. Robione to jest, w sposób podobny do prezentowanego na listingu numer 29. Można wyróżnić w nim wstępną walidację danych, poprzez sprawdzenie poprawnego formatu dat (`DateTime.TryParse`) oraz zabezpieczenie przed wprowadzeniem „daty od” młodszej od „daty do”. Najważniejsze jednak jest utworzenie nowych parametrów `ReportParameter`, opisujących daty, a następnie przekazanie ich do raportu metodą `SetParameters`.

```
private void SetReportData()
{
    DateTime outer;
    DateTime? startDate = null;
    DateTime? endDate = null;
    if (DateTime.TryParse(txtStartDate.Text, out outer))
        startDate = outer;
    if (DateTime.TryParse(txtEndDate.Text, out outer))
        endDate = outer;
    MessageBoxControl msgBox =
    (MessageBoxControl)Page.Master.FindControl("messageBox");
    if (startDate == null || endDate == null)
    {
        msgBox.ShowValidateError("Zły format daty. Prawidłowy format daty to MM-dd-
        yyyy.");
        return;
    }
    if (startDate > endDate)
    {
        msgBox.ShowValidateError("Data od nie może być późniejsza niż data do.");
        return;
    }
    ReportParameter[] reportParameters = new ReportParameter[2];

    reportParameters[0] = new ReportParameter("startdate",
    startDate.Value.ToString());
    reportParameters[1] = new ReportParameter("enddate", endDate.Value.ToString());

    rvSales.ServerReport.SetParameters(reportParameters);
    rvSales.ServerReport.Refresh();
}
```

Listing 29 - Ustawienie parametrów raportu sprzedaży według grup produktów

7.6 Obsługa błędów – log zdarzeń

Każdy system informatyczny, musi posiadać pewien mechanizm pozwalający reagować w odpowiedni sposób podczas wystąpienia błędu. W wielu przypadkach mechanizm ten można zaimplementować z wykorzystaniem bloku try-catch. Ponieważ błędy wygenerowane w metodach zagnieżdżonych są przekazywane do ich przodków, można je „wyłapywać” na różnych poziomach wywołań funkcji. Przykład zamykania połączenia przy wystąpieniu błędu w odczycie z mobilnej bazy danych zamieszczono na listingu 30. Po wykonaniu określonej akcji, wyjątek, „rzucany” jest dalej do funkcji nadrzędnych, w celu obsłużenia na wyższym poziomie.

```
public static DbDataReader ExecuteReader(DbCommand cmd)
{
    SqlConnection conn = new SqlConnection(DBHelper.GetConnectionString());
    try
    {
        PrepareCommand(cmd, conn, null);
        DbDataReader rdr = cmd.ExecuteReader(CommandBehavior.CloseConnection);
        cmd.Parameters.Clear();
        return rdr;
    }
    catch
    {
        conn.Close();
        throw;
    }
}
```

Listing 30 - Blok try-catch - aplikacja mobilna

W aplikacjach ASP.NET istnieje możliwość obsługi błędów na poziomie globalnym. Wykonywane jest to w zdarzeniu `Application_Error` pliku `Global.asax`, które zostało przedstawione na listingu 31. Błędy rejestrowane są w pliku tekstowym, przy użyciu projektu Log4NET²⁹, zapisywane są w nim dane takie jak, treść wyjątku, data, użytkownik, adres strony, na której wystąpił czy identyfikator sesji.

```
void Application_Error(object sender, EventArgs e)
{
    try
    {
        log4net.ILog log = log4net.LogManager.GetLogger("Application");
        StringBuilder errorBuilder = new StringBuilder();
        errorBuilder.AppendLine("");
        if (Context.User != null && Context.User.Identity != null)
        {
            errorBuilder.AppendLine("User: " + Context.User.Identity.Name);
        }
        if (Context.Request != null)
        {
            errorBuilder.AppendLine("Request:");
            if (Context.Request.Cookies["ASP.NET_SessionID"] != null)
                errorBuilder.AppendLine("SessionID: " +
                    Context.Request.Cookies["ASP.NET_SessionID"].Value);
            errorBuilder.AppendLine("Url: " + Context.Request.Url);
            errorBuilder.AppendLine("UserIP: " + Context.Request.UserHostAddress);
            errorBuilder.AppendLine("UserHost: " + Context.Request.UserHostName);
        }
    }
}
```

²⁹ Log4NET - <http://logging.apache.org/log4net/>

```

    }
    errorBuilder.AppendLine("Exception:");
    errorBuilder.AppendLine(Context.Error.ToString());
    if (Context.Error.InnerException != null)
    {
        errorBuilder.AppendLine("-----");
        errorBuilder.AppendLine("Inner Exception:");
        errorBuilder.AppendLine(Context.Error.InnerException.ToString());
    }
    errorBuilder.AppendLine("#####");
    log.Error(errorBuilder.ToString());
}
catch (Exception ex)
{
}
}

```

Listing 31 - Obsługa błędów, log zdarzeń - aplikacja internetowa

7.7 Walidacja danych

Zapewnienie poprawności oraz obligatoryjności niektórych danych, realizowane jest jako proces walidacji. Do zaimplementowania tego mechanizmu, użyto standardowych kontrolerek ASP.NET, takich jak np.:

- **RequiredFieldValidator** - sprawdza, czy użytkownik wpisał coś do danego pola,
- **RangeValidator** – sprawdza, czy wpisana wartość mieści się w przedziale,
- **RegularExpressionValidator** – pozwala na sprawdzenie, czy wpisane wartość odpowiada wyrażeniu regularnemu,
- **CompareValidator** – porównuje wartości, dwóch podanych kontrolerek.

Kontrolki te, posiadają niektóre metody oraz wartości wspólne. Należą do nich:

- **ControlToValidate** – nazwa kontrolki do walidacji,
- **ErrorMessage** – treść komunikatu, jeżeli wystąpi błąd walidacji,
- **Validate** – metoda sprawdza poprawność danych w kontrolce powiązanej,
- **Display** – określa sposób wyświetlania komunikatu (None – brak, Static – miejsce od razu rezerwowane na stronie, Dynamic – miejsce na wyświetlenie tworzone dynamicznie).

W celu wyświetlania bardziej użytecznych, dynamicznych podpowiedzi (patrz rysunek 34), użyto kontrolki **ValidatorCalloutExtender**, dostępnej w zestawie **AjaxControlToolkit**³⁰. W aplikacji występuje wiele obligatoryjnych pól danych, więc stworzono kontrolkę rozszerzającą standardowy element **TextBox** o walidację (Listing 32).

³⁰ AjaxControlToolkit - <http://www.ajaxcontroltoolkit.com/>

```

<asp:TextBox ID="txtValue" runat="server"></asp:TextBox>
<asp:RequiredFieldValidator ID="rfvValue" ControlToValidate="txtValue" Font-
Name="verdana" Font-Size="9pt" runat="server"
Display="None"></asp:RequiredFieldValidator>
<ajaxToolkit:ValidatorCalloutExtender runat="Server" ID="rfvValueE"
TargetControlID="rfvValue" HighlightCssClass="validatorCalloutHighlight" />

```

Listing 32 - Walidacja kontrolki TextBox

7.8 Wysłanie wiadomości email

System wsparcia sprzedaży posiada moduł pozwalający na wysyłanie wiadomości email. Funkcjonalność ta utworzona została w celu powiadamiania klientów o akceptacji oraz wysyłce zamówienia. Konfiguracja skrzynki mailowej, znajduje się w pliku web.config aplikacji internetowej w sekcji `<system.net>`. Występują tam wszystkie niezbędne ustawienia potrzebne do prawidłowego działania modułu, takie jak adres serwera SMTP, port, nazwa użytkownika oraz hasło.

```

<system.net>
  <mailSettings>
    <smtp from="Nazwa firmy < sfasystem.mail@gmail.com >"
      <network host="smtp.gmail.com" port="587"
        userName="sfasystem.mail@gmail.com" password="*****"/>
    </smtp>
  </mailSettings>
</system.net>

```

Listing 33 - Konfiguracja połączenia z serwerem poczty

Maile wysyłane są za pomocą klasy `MailHelper`, której przykładowa funkcja `SendShippedOrderEmail`, została zaprezentowana na listingu 34. Metoda ta jest wywoływana podczas zmiany statusu zamówienia na „Wysłano”. Tworzenie maila, polega na przypisaniu pewnym zmiennym fragmentom szablonu wiadomości w formacie HTML (np. `<%Address%>` - adres klienta), konkretnych wartości. W przykładzie zaimplementowano to za pomocą zdefiniowania słownika (`IDictionary replacements`), zawierającego wszystkie zmienne szablonu oraz ich wartości. Po zdefiniowaniu treści (`MailDefinition`), tworzona jest dla niej wiadomość email (`MailMessage`), którą wysyła się za pomocą klasy `SmtpClient`. Klasa ta, korzysta z ustawień zapisanych w pliku konfiguracyjnym, oczywiście na tym etapie możliwa jest ich zmiana.

```

public static void SendShippedOrderEmail(string emailAddress, Order order)
{
    MailDefinition mailDefinition = new MailDefinition();
    mailDefinition.BodyFileName = "~/MailTemplates/ZamowienieWyslano.html";
    mailDefinition.IsBodyHtml = true;
    mailDefinition.Subject = "Nazwa firmy - Zamówienie wysłano";
    IDictionary replacements = new Hashtable();
    //Dodawanie znaczników, które zostaną podmienione w szablonie maila na właściwe
    wartości
    replacements.Add("<%OrderDate%>", order.OrderDate.ToString("dd-MM-yyyy"));
    Address address = order.CustomerFacility.Address;
    String strAddress = "ul. " + address.Street +
        address.HouseNr + "/" + address.ApartmentNr + ", " +
        address.ZipCode + " " + address.City.Name + " " + address.Country.Name;
    replacements.Add("<%Address%>", strAddress);
    replacements.Add("<%Total%>", order.Total.ToString("0.00 zł"));
}

```

```

MailMessage msg = mailDefinition.CreateMailMessage(emailAddress, replacements,
new Panel());
MailAddress mailFrom = new MailAddress(msg.From.Address, "Nazwa firmy");
msg.From = mailFrom;

SmtpClient client = new SmtpClient();
client.EnableSsl = true;
client.Send(msg);
}

```

Listing 34 - Funkcja wysyłająca mail informujący o wysyłce zamówienia

7.9 Synchronizacja danych

Projektowany system jest systemem rozproszonym, w którym urządzenie przenośne pełni funkcje klienta większego systemu. Aby wszystko działało poprawnie, zamówienia spływały do siedziby firmy w jak najkrótszym czasie oraz przedstawiciele posiadali aktualne informacje niezbędne było wprowadzenie mechanizmu synchronizacji danych urządzeń mobilnych z głównym systemem. Dostępne są dwa sposoby synchronizacji mobilnej bazy danych z głównym serwerem. Pierwszy „Remote Data Access”, pozwala wykonać zapytanie SQL, pobrać dane z serwera i przechowywać lokalnie ich kopię. Jeśli dane mają być uaktualnione w urządzeniu przenośnym po czym zwrócone do serwera, można włączyć śledzenie zmian, a na serwer wysłać tylko zmienione rekordy. Jednakże, do wykonanie tej funkcjonalności w zaprojektowanym systemie użyto drugiej możliwości, jaką była replikacja scalająca (ang. Merge Replication). Mechanizm ten jest znacznie bardziej zaawansowany. Pozwala m. in. śledzić zmiany nie tylko w urządzeniu przenośnym, ale i na serwerze, dzięki czemu podczas synchronizacji zmienione informacje mogą być przesłane w obydwu kierunkach. Przy użyciu Merge Replication, programista posiada większą kontrolę nad sposobem, w jaki zostaną rozwiązane ewentualne konflikty.

Tworzenie publikacji

Pierwszym krokiem, jaki należy wykonać w procesie realizacji mechanizmu Merge Replication, jest stworzenie publikacji głównej bazy danych. Publikacja została utworzona z uwzględnieniem wszystkich tabel bazy danych w celu zachowania więzów referencyjnych. Jednakże, na tabele Order, OrderLine oraz Invoice zostały nałożone filtry, w celu ściągania danych tylko o zamówieniach utworzonych przez konkretnego przedstawiciela handlowego. Rozwiązanie to sprawia, że dane urządzenie mobilne może być przyporządkowane tylko do jednego użytkownika, co zwiększa bezpieczeństwo systemu.

```
SELECT <published_columns> FROM [dbo].[Order] WHERE Convert(nvarchar(100),[EmployeeID])=HOST_NAME()
```

Zaprezentowany filtr zamówień posiada dwa filtry rozszerzające.

```
SELECT <published_columns> FROM [dbo].[Order] INNER JOIN [dbo].[Invoice] ON [Order].[OrderID] = [Invoice].[OrderID]
```

```
SELECT <published_columns> FROM [dbo].[Order] INNER JOIN [dbo].[OrderLine] ON [Order].[OrderID] = [OrderLine].[OrderID]
```

Dzięki nim, tabelom Invoice oraz OrderLine bazy mobilnej, zostaną przekazane jedynie wiersze dotyczące przefiltrowanych zamówień. Należy zwrócić uwagę na parametr HOST_NAME, który zwraca funkcja o tej samej nazwie. Jest to, może niezbyt elegancki, ale często stosowany oraz skuteczny sposób, na przekazanie jednego parametru (w tym wypadku identyfikatora pracownika) z poziomu kodu aplikacji do filtrów replikacji. Wybrano, także sposób rozwiązywania konfliktów podczas replikacji, na uznający dane głównego serwera jako priorytetowe w stosunku z danymi mobilnej bazy danych.

Po pomyślnym wykonaniu publikacji, każda z tabel należących do niej zostaje rozszerzona o nową kolumnę ROWGUIDCOL, przechowującą unikatowe identyfikatory rekordów oraz dodane są także wyzwalacze (ang. triggers), działające przy każdej operacji insert, update czy delete.

Konfiguracja serwera IIS

Aplikacja subskrybenta (w przypadku tego systemu, aplikacja mobilna) danej publikacji komunikuje się z serwerem głównym poprzez protokół http. Aby komunikacja stała się możliwa, należy odpowiednio skonfigurować serwer IIS zainstalowany na serwerze głównym. Trzeba zwrócić uwagę, czy moduł ISAPI serwera IIS jest zainstalowany, gdyż za jego pomocą wykonywane są zdalne wywołania udostępnianych bibliotek. Następnie najlepiej użyć specjalnego narzędzia, jakim jest Configure Web Synchronization Wizard. Narzędzie to tworzy wirtualny katalog, do którego kopiuje bibliotekę Server Agent (plik sqlcesa35.dll), poprzez którą dokonywana będzie replikacja. Aplikacja konfigurująca poprosi także o podanie nazwy zasobu sieciowego z udostępnioną migawką replikowanej bazy danych. Migawkę, zaraz po utworzeniu publikacji wykonuje program zwany Snapshot Agent, folder z utworzonymi wtedy plikami należy udostępnić jako zasób sieciowy oraz nadać uprawnienia kontu systemowemu, w środowisku którego uruchomiony jest serwer IIS. Czynność tworzenia migawki należy powtórzyć po każdorazowej zmianie w schemacie bazy danych.

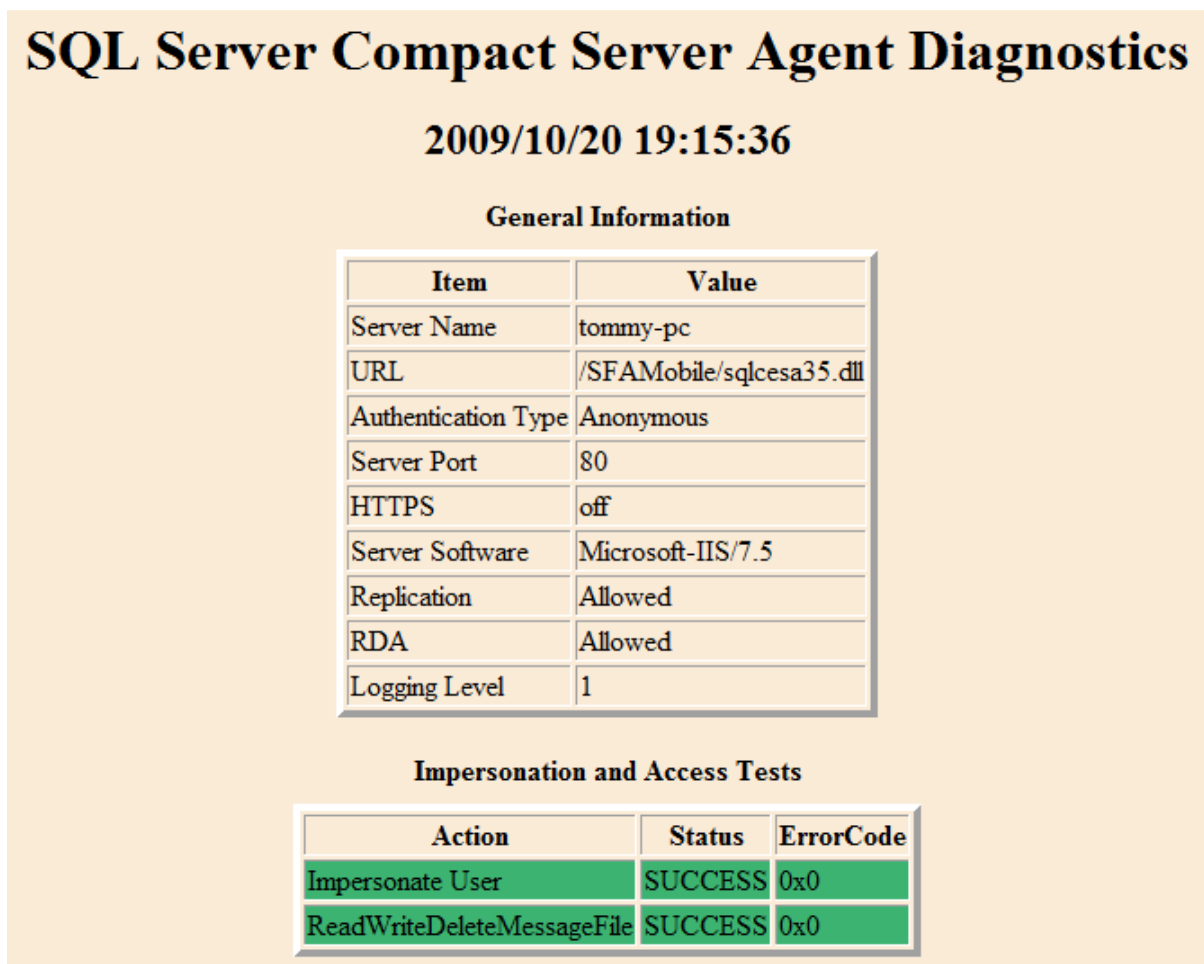
Po ukończonym procesie konfiguracji używając poniższego formatu adresu, można przeprowadzić test poprawności działania Server Agent.

[http://\[nazwa-komputera\]/\[zasób sieciowy\]/sqlcesa35.dll?diag](http://[nazwa-komputera]/[zasób sieciowy]/sqlcesa35.dll?diag)

Dla wykonywanego systemu adres ten miał następującą postać:

<http://TOMMY-PC/SFAMobile/sqlcesa35.dll?diag>

Jeżeli wszystko jest poprawnie ustawione w przeglądarce zostanie wyświetlona strona, podobna do zaprezentowanej na rysunku 53.



Rysunek 53 - SQL Server Agent Diagnoza

Obsługa replikacji – aplikacja Pocket PC

W celu obsługi synchronizacji po stronie aplikacji mobilnej zaimplementowana została klasa `SynchronizationManager`. Przed rozpoczęciem tego procesu niezbędna jest inicjalizacja klasy `SqlCeReplication` poprzez podanie niezbędnych parametrów, realizuje to funkcja `Initialize()`. W metodzie tej, przekazywany jest identyfikator pracownika, jako wspomniany wcześniej parametr filtrów replikacji o nazwie `HostName`.

```
public void Initialize()
{
    replication = new SqlCeReplication();
    replication.InternetUrl = @"http://192.168.1.1/SFAMobile/sqlcesa35.dll";
    replication.Publisher = "TOMMY-PC";
}
```

```

replication.PublisherDatabase = "SFASystem";
replication.PublisherSecurityMode = SecurityType.DBAuthentication;
replication.PublisherLogin = "sfasystemuser";
replication.PublisherPassword = "sfasystemuser";
replication.Publication = "SFASystem";
replication.HostName = DataManager.Instance.CurrentEmployeeID.ToString();
replication.Subscriber = "SFASystemSubscriber";
replication.SubscriberConnectionString = "Data Source=\" +
Settings.DatabaseFile + "\";Password=\"sfasystem\";Encrypt=true";
isInitialized = true;
}

```

Listing 35 - Metoda Initialize, menadżer replikacji

Dostępne są dwie metody przeprowadzania replikacji, asynchroniczna oraz synchroniczna. W systemie wykorzystana została pierwsza metoda z racji większych możliwości takich jak, anulowanie procesu czy sposobność użycia paska postępu. Rozpoczęcie asynchronicznej replikacji realizuje funkcja `PerformAsyncSynchronization()`. Jeżeli mobilna baza danych nie istnieje, tworzona jest jako subskrypcja serwerowej publikacji (`AddSubscription` (`AddOption.CreateDatabase`)). Synchronizacja rozpoczyna się poprzez wywołanie metody `BeginSynchronize`, której parametrami są m.in. funkcja zwrotna, wywoływana podczas zakończenia synchronizacji (`SyncCompletedCallback`), czy funkcja zwrotna, przekazująca aktualny postęp synchronizacji (`OnSynchronizationCallback`).

```

public void PerformAsyncSynchronization()
{
    if (this.isSynchronizing)
    {
        return;
    }

    this.isSynchronizing = true;
    try
    {
        //jesli baza nie istnieje, tworzy subskrypcję
        if (!File.Exists(Settings.DatabaseFile))
        {
            replication.AddSubscription(AddOption.CreateDatabase);
        }
        OnSyncStarted();
        //synchronizacja
        this.asyncResult = replication.BeginSynchronize(
            new AsyncCallback(SyncCompletedCallback),
            null,
            null,
            new OnSynchronization(OnSynchronizationCallback),
            replication);
    }
    catch (Exception ex)
    {
        OnSyncError(ex);
    }
}

```

Listing 36 - Rozpoczęcie asynchronicznej replikacji, menadżer replikacji

Wspomniana funkcja zwrotna, wykonywana po zakończeniu procesu synchronizacji sprawdza, czy synchronizacja zakończyła się sukcesem i przekazuje go funkcjom obsługi zdarzenia OnSyncCompleted.

```
/// <summary>
/// Metoda wywoływana gdy synchronizacja zostanie zakończona
/// </summary>
/// <param name="ar"></param>
private void SyncCompletedCallback(IAsyncResult ar)
{
    this.isSynchronizing = false;
    SyncComplEventArgs sychResultArg;
    try
    {
        //Pobierz rezultat synchronizacji
        SqlCeReplication replication = (SqlCeReplication)ar.AsyncState;
        replication.EndSynchronize(ar);
        replication.SaveProperties();
        replication = null;
        sychResultArg = new SyncComplEventArgs(SynchronizationResult.Success);
    }
    catch (Exception ex)
    {
        LogHelper.Error("Synchronization failed\n" + ex.Message + "\n" +
            ex.StackTrace);
        UIHelper.ShowError("Synchronization failed\n" + ex.Message + "\n");
        sychResultArg = new SyncComplEventArgs(SynchronizationResult.Failed);
    }
    finally
    {
        this.isSynchronizing = false;
    }
    OnSyncCompleted(sychResultArg);
}
```

Listing 37 – Zakończenie asynchronicznej replikacji, menadżer replikacji

Dzięki zastosowaniu asynchronicznej metody, użytkownik może anulować przeprowadzanie replikacji. Funkcjonalność ta została zaimplementowana w funkcji CancelSynchronize(). Po takiej rezygnacji (CancelSynchronize()), podobnie jak w poprzednim przykładzie odpowiedni status jest przekazywany dalej poprzez zdarzenie OnSyncCompleted.

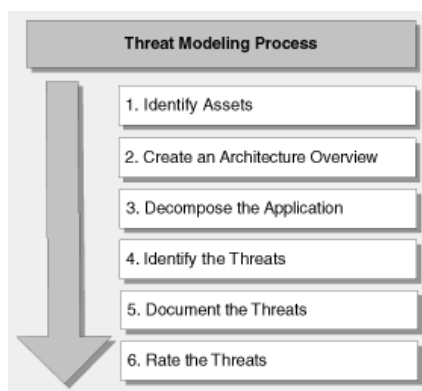
```
/// <summary>
/// Metoda anulująca przeprowadzaną synchronizację asynchroniczna
/// </summary>
public void CancelSynchronize()
{
    try
    {
        this.isSynchronizing = false;
        replication.CancelSynchronize();

        OnSyncCompleted(new SyncComplEventArgs(SynchronizationResult.Cancelled));
    }
    catch (Exception ex)
    {
        LogHelper.Error("Cannot cancel synchronization "+ex.Message);
        throw new Exception("Cannot cancel synchronization");
    }
}
```

Listing 38 - Anulowanie asynchronicznej replikacji, menadżer replikacji

8. Bezpieczeństwo

W dzisiejszych czasach bezpieczeństwo systemów informatycznych jest jednym z najważniejszych aspektów, które trzeba wziąć pod uwagę już podczas fazy jego wstępnego projektowania. Kwestia ta szczególnie ważna jest jeżeli chodzi o aplikacje biznesowe, gdzie zachowanie poufności niektórych informacji jest kluczowe dla firmy i czasami stanowi to dla niej przysłowiowe „być albo nie być”. Dlatego też dobrą praktyką jest zintegrowanie bezpieczeństwa z całym cyklem rozwojowym oprogramowania, korzystając z technik, takich jak np. modelowanie zagrożeń ³¹.



Rysunek 54 – modelowanie zagrożeń

Proces ten można opisać w sześciu punktach (Rysunek 54).

1. Zidentyfikowanie cennych zasobów.
2. Tworzenie przeglądu architektury systemu.
3. Dekompozycja aplikacji
4. Zidentyfikowanie zagrożeń
5. Udokumentowanie zagrożeń
6. Ocenienie zagrożeń

W celu lepszej identyfikacji zagrożeń możemy posłużyć się modelem STRIDE ³² obejmującym:

- Spoofing (podszywanie się po inną osobę przy pomocy nieautoryzowanego dostępu do jej danych identyfikacyjnych)
- Tampering (złośliwa modyfikacja danych)
- Repudiation (wyparcie się przeprowadzenia określonej akcji przez autora)

³¹ „Threat modeling” <http://msdn.microsoft.com/en-us/library/aa302419.aspx?>

³² „Designing for Securability” [http://msdn.microsoft.com/en-us/library/aa291875\(VS.71\).aspx](http://msdn.microsoft.com/en-us/library/aa291875(VS.71).aspx)

- Information Disclose (dotarcie do informacji przez osobę niemającą odpowiednich uprawnień)
- Denial of Service (uniemożliwienie skorzystania z usługi bądź systemu przez uprawnionego użytkownika)
- Elevation of Priviledge (nieautoryzowane zwiększenie przywilejów użytkownika)

Wiedza na temat możliwych zagrożeń systemu jest bardzo przydatna na każdym etapie cyklu tworzenia oprogramowania a poprzez podejmowanie dobrych decyzji projektowych oraz uświadamianie programistów w kwestiach bezpieczeństwa, końcowy produkt zyska na jakości.

8.1 Bezpieczeństwo aplikacji mobilnych

Temat bezpieczeństwa w zakresie tworzenia oraz rozwoju aplikacji mobilnych jest często tematem pomijanym lub spychanym na margines. Dzieje się tak głównie dlatego, że wielu programistów oraz projektantów podchodzi z obojętnością do kwestii bezpieczeństwa.

„Na niektóre przyczyny tej obojętności można natrafić, przyglądając się sposobowi tworzenia aplikacji mobilnych przez zespoły programistów; zdecydowana większość aplikacji pochodzi od małych zespołów, złożonych z dwóch lub trzech osób, które często, dysponując niewielkimi środkami, rozpoczynają działalność gospodarczą i wypuszczają na rynek produkt z zamiarem zagospodarowania niszy rynkowej. Zdarza się, że rozwój aplikacji mobilnych jest jedynie produktem ubocznym większego projektu z zakresu aplikacji biurowych realizowanego przez dział IT, który pragnie jedynie poszerzyć możliwości aplikacji biurowych, tak, aby móc sobie odhaczyć ptaszkiem wybór „opcje mobilne”.”³³

Należy także zwrócić uwagę na specyficzne względy, dla których bezpieczeństwo w środowisku mobilnym nie powinno być zaniedbywane. Są to, np. podwyższony czynnik ryzyka wycieku informacji firmowej poprzez chociażby zagubienie lub kradzież urządzenia przenośnego czy też niebezpieczeństwa podczas korzystania z publicznych, często niezabezpieczonych sieci bezprzewodowych.

8.1.1 Aspekty związane w wyświetlaniem oraz wprowadzaniem danych

Z racji tego, że urządzenie mobilne dysponuje niewielką powierzchnią wyświetlacza oraz małymi możliwościami klawiatury numerycznej projektowanie interfejsu użytkownika nie należy do zadań trywialnych. Od strony bezpieczeństwa najlepszym rozwiązaniem byłoby użycie przymusu wpisywania przez użytkownika hasła, np. co 30 sekund w celu odblokowania dostępu do aplikacji.

³³ „Bezpieczne aplikacje mobilne – oksymoron?”

http://www.microsoft.com/poland/technet/bazawiedzy/centrumrozwiazan/cr227_01.mspix

Jednak takie podejście zniechęci kogokolwiek do korzystania z urządzenia, więc można rozważyć chociażby zastosowanie dwustopniowej lub nawet kilkustopniowej blokady działania aplikacji. Rozwiązanie takie polegałoby na zastosowaniu dwóch haseł. Dłuższe wprowadzane w większych odstępach czasowych lub przy próbie uzyskania dostępu do kluczowych zasobów systemu. Natomiast krótsze (np. cztero cyfrowy kod PIN) w mniejszych odstępach czasowych lub przy dostępie do większości zasobów systemowych. W niektórych urządzeniach dodatkowo można zastosować czytniki danych biometrycznych czy też kart magnetycznych. Mobilne systemy Windows CE oferują dodatkowe interfejsy oraz uproszczenia przy projektowaniu tego typu blokad na wielu poziomach dostępu, w dalszej części zostaną opisane niektóre mechanizmy.

8.1.2 Local Authentication SubSystem (LASS)

System operacyjny Windows Mobile posiada wiele funkcji oraz udogodnień, które mogą być używane do tworzenia strategii bezpieczeństwa dla zabezpieczania urządzenia oraz aplikacji, które się na nim znajdują. Jednym z takich mechanizmów jest komponent architektury systemu operacyjnego zwany LASS (Local Authentication SubSystem)³⁴. Jest to moduł, który pozwala na zastosowanie zaawansowanych mechanizmów i polityk uwierzytelniania użytkowników urządzenia wyposażonego w mobilną wersję systemu Windows. Zdefiniowanie takiego mechanizmu polega na instalacji modułu zwanego LAP (Local Authentication Plugin), który odpowiedzialny jest za proces uwierzytelniania użytkowników. Moduł LAP to dynamicznie ładowana biblioteka (DLL), eksportująca określone funkcje, które wywoływane są przy uruchamianiu systemu w celu uwierzytelnienia użytkownika. Instalowany moduł LAP dodatkowo musi posiadać specjalny podpis cyfrowy, który uzyskuje się poprzez przeprowadzany przez firmę Microsoft proces weryfikacji konkretnego modułu. Tworząc moduł LAP można odwoływać się do szeregu serwisów udostępnianych przez środowisko Windows Mobile takich jak np. CryptoAPI – warstwy systemu Windows odpowiedzialnej za udostępnianie usług kryptograficznych.

8.1.3 Zastosowanie metod kryptograficznych

W projektowanym systemie metody kryptograficzne zostały użyte w celu zaszyfrowania wszystkich haseł użytkowników (Listing 39). Takie rozwiązanie nie pozwala na przypominanie haseł, w przypadku nie pamiętania, dlatego też możliwa jest tylko jego zmiana na inne. Zaszyfrowane hasło tworzone jest przy użyciu algorytmu SHA1³⁵ z ciągu znaków podanych przez użytkownika i wygenerowanego klucza (Salt). Każdorazowo podczas logowania, podawany jako hasło ciąg jest haszowany tym samym

³⁴ „Local Authentication Subsystem (LASS)” <http://msdn.microsoft.com/en-us/library/ms926467.aspx>

³⁵ SHA_hash_functions http://en.wikipedia.org/wiki/SHA_hash_functions

algorytmem z użyciem klucza z bazy danych, co pozwala na weryfikację poprawności wprowadzonego ciągu.

```
private static string CreatePasswordHash(string Password, string Salt)
{
    string hashed = "";
    SHA1 sha1 = new SHA1CryptoServiceProvider();
    byte[] hash =
        sha1.ComputeHash(System.Text.Encoding.UTF8.GetBytes(Password+Salt));

    foreach (byte b in hash)
        hashed += String.Format("{0,2:X2}", b);

    return hashed;
}

private static string CreateSalt(int size)
{
    RNGCryptoServiceProvider provider = new RNGCryptoServiceProvider();
    byte[] data = new byte[size];
    provider.GetBytes(data);
    return Convert.ToBase64String(data);
}
```

Listing 39 - Metody kryptograficzne Pocket PC

8.1.4 Podpisywanie kodu

Inny mechanizm podstawowej ochrony jest implementowany dzięki podpisywaniu kodu. W celu potwierdzenia tego czy kod jest podpisany, czy sygnatura jest ważna i zgadza się z autoryzowanym certyfikatem zainstalowanym na urządzeniu, Windows Mobile sprawdza każdy moduł wykonywalny, taki, jak biblioteki dołączane dynamicznie (.dll), i pliki wykonywalne (.exe), w momencie kiedy są one ładowane na urządzenie.

Windows Mobile może być tak skonfigurowany, aby blokować wykonywanie kodu, który nie spełnia wyżej wymienionych kryteriów, w ten sposób urządzenie jest zabezpieczane przed złośliwym kodem. Instalowanie oprogramowania poprzez pliki CAB jest również chronione przez ten proces z własną bazą certyfikatów, a w urządzeniu dostępny jest magazyn odwołań w celu blokowania wykonywania i instalacji podstępnych aplikacji

8.1.5 Bezpieczeństwo danych w aspekcie aplikacji mobilnych – bezpieczna replikacja

Replikacja jest procesem powielania informacji pomiędzy różnymi serwerami baz danych poprzez wykorzystanie często niebezpiecznych mediów transmisyjnych takich jak np. Internet. Ważne jest, aby zrozumieć i stosować najlepsze podejścia do przeprowadzania bezpiecznych replikacji

(*Replication Security Best Practices* ³⁶). Do środków wzmacniających bezpieczeństwo danych podczas replikacji należą między innymi:

- zapewnienie szyfrowania przesyłanych danych używając standardów przemysłowych takich jak Virtual Private Networks (VPN), Secure Sockets Layer (SSL), czy też IP Security (IPSEC).
- uruchomienie każdego agenta replikacji w obrębie innego konta użytkownika systemu Windows i użycie metody Windows Authentication w celu uwierzytelniania każdego połączenia agenta replikacji.
- przydzielenie tylko niezbędnych uprawnień każdemu z agentów.
- używanie udostępnianych udziałów sieciowych zamiast ścieżki lokalnej do folderu z migawką.

W projektowanym systemie w celu zapewnienia lepszej ochrony danych, pierwsza synchronizacja tworząca mobilną bazę danych, nie jest możliwa, jeżeli użytkownik nie zostanie uwierzytelniony. Uwierzytelnianie przeprowadzone jest poprzez specjalną usługę sieciową dostępną po stronie głównego serwera. Dodatkowo dostęp do utworzonej mobilnej bazy danych jest zabezpieczony hasłem, natomiast sam plik, jest zaszyfrowany.

8.2 Bezpieczeństwo aplikacji internetowych ASP .NET

Aplikacje uruchamiane poprzez przeglądarkę internetową z racji swojej zwykle szerszej dostępności, są szczególnie narażone na ataki oraz próby włamań. Wprawdzie w przypadku systemów takich jak projektowany, czyli aplikacji działających w sieci korporacyjnej łatwiej jest ograniczyć dostęp osobom z zewnątrz, poprzez zabezpieczenie całej sieci lub w przypadku zdalnego dostępu używanie VPN. Należy także zadbać o odpowiednią politykę bezpieczeństwa na poziomie projektowanej aplikacji, używając w sposób przemysłowy dostępnych rozwiązań.

8.2.1 Mechanizmy MembershipProvider oraz RoleProvider

Korzystając z technologii .NET możemy wykorzystać funkcjonalność ASP .NET Membership Provider, która udostępnia metody, kontrolki oraz interfejsy niezbędne do zarządzania oraz uwierzytelniania użytkowników systemu. Dzięki przemyślanej architekturze opisywany mechanizm zapewnia współpracę z różnymi dostawcami danych, przechowującymi dane użytkowników oraz umożliwia zastosowanie własnej koncepcji składowania danych. Podczas pracy z bazą danych typu SQL Server, często używaną pochodną abstrakcyjnej klasy `MembershipProvider` jest dostarczana w platformie .NET klasa `SqlMembershipProvider`. Aby korzystać z tego rozwiązania niezbędne jest

³⁶ "Replication Security Best Practices" - [http://technet.microsoft.com/pl-pl/library/ms151227\(en-us\).aspx](http://technet.microsoft.com/pl-pl/library/ms151227(en-us).aspx)

zdefiniowanie oddzielnej bazy danych składującej dane użytkowników lub skonfigurowanie głównej bazy danych aplikacji, co wiąże się z dodaniem do niej kilku tabel. Z wymienionej bazy danych korzysta także drugi mechanizm zabezpieczeń aplikacji ASP .NET, jakim jest mechanizm autoryzacji oparty o klasę abstrakcyjną [RoleProvider](#). Jego celem jest przydzielanie, lub zabranianie dostępu do określonych zasobów systemowych (np. stron) zgodnie z rolą jaką dany użytkownik posiada. Obie wspomniane klasy są abstrakcyjne, więc aby je wykorzystać należy użyć albo dostępnych już w platformie .NET klas dostosowanych do wykorzystywanego dostawcy danych, lub napisać własne implementując tylko niezbędne funkcje.

Podczas projektowania systemu sprzedaży wybrano drugą możliwość, gdyż uznano, że standardowy model wprowadza zbyt wiele nadmiarowych tabel do bazy danych natomiast wystarczy użycie już posiadanych tabel Employee oraz Role. W celach poglądowych na listingu 40 przedstawiono fragment własnej implementacji klasy typu [MembershipProvider](#), którą zastosowano w systemie. Podczas tworzenia i weryfikacji użytkownika zastosowano odpowiednio funkcje CreatePasswordHash oraz CreateSalt, których działanie opisano już na przykładzie zabezpieczeń dostępu do aplikacji mobilnej.

```
public class SFAMembershipProvider : MembershipProvider
{
    public override MembershipUser CreateUser(string username, string password,
        string email, string passwordQuestion, string passwordAnswer, bool isApproved,
        object providerUserKey, out MembershipCreateStatus status)
    {
        MembershipUser user = null;
        Employee employee = new Employee();
        employee.Login = username;
        employee.PasswordSalt = CreateSalt(5);
        employee.PasswordHash = CreatePasswordHash(password,
            employee.PasswordSalt);
        employee.Active = true;
        employee.CreationDate = DateTime.Now;
        employee.LastActivityDate = DateTime.Now;
        EmployeeService.SaveEmployee(employee);
        status = MembershipCreateStatus.Success;
        user = new MembershipUser(this.Name, employee.Login, null, string.Empty,
            string.Empty, string.Empty, true, false, DateTime.Now, DateTime.Now,
            DateTime.Now, DateTime.Now, DateTime.Now);
        return user;
    }
    public override bool ValidateUser(string username, string password)
    {
        Employee employee = EmployeeService.GetEmployeeByLogin(username);

        if (employee == null)
            return false;
        if (!employee.Active)
            return false;

        string passwordHash = CreatePasswordHash(password, employee.PasswordSalt);
        bool result = employee.PasswordHash.Equals(passwordHash);
        if (result)
        {
            return true;
        }
    }
}
```

```

        }
        return false;
    }
    [...]
}

```

Listing 40 - Fragment klasy SFAMembershipProvider

ASP .NET udostępnia kilka kontrolki przyspieszających wprowadzanie funkcji bezpieczeństwa do systemu, które to wykorzystują wybraną klasę typu `MembershipProvider`. W skład kontrolki tych wchodzi między innymi:

- `Login` – kontrolka uwierzytelniająca, przedstawia widok gotowego formularza z dostępnym polem na login i hasło.
- `LoginStatus` – kontrolka wyświetlająca zdefiniowany tekst w zależności, czy użytkownik jest zalogowany, czy też nie.
- `CreateUserWizard` – kontrolka pozwalająca w łatwy sposób zdefiniowanie kroków podczas rejestracji użytkownika.

Niemniej ważnym mechanizmem, jakim jest uwierzytelnianie jest również autoryzacja. Stosunkowo łatwo wyobrazić sobie sytuację, w której sama identyfikacja nie jest wystarczająca do zapewnienia pełnej kontroli nad odpowiednim zarządzaniem dostępem do treści publikowanych na stronach aplikacji. Braki te implementuje wspomniana wcześniej klasa `RoleProvider`, która zajmuje się chociażby weryfikacją, czy podany użytkownik należy do danej roli. Implementację sprawdzania tej kontroli zaprezentowano na listingu 41, który przedstawia metodę, będącą fragmentem klasy `SFARoleProvider`.

```

public override bool IsUserInRole(string username, string roleName)
{
    Employee employee = EmployeeService.GetEmployeeByLogin(username);
    if (employee == null)
        return false;
    IList<Role> empRoles = employee.Roles;
    foreach (Role role in empRoles)
    {
        if (role.Name.Equals(roleName))
            return true;
    }
    return false;
}

```

Listing 41 - Metoda sprawdzająca czy użytkownik należy do określonej roli

Dzięki zaimplementowaniu najważniejszych funkcji klasy `RoleProvider`, możliwe jest blokowanie dostępu do określonych stron aplikacji internetowej z poziomu pliku konfiguracyjnego. Na listingu 42, przedstawiono fragment pliku konfiguracyjnego `web.config`, odpowiadający za zablokowanie dostępu do pliku z raportem o sprzedaży według regionów wszystkim użytkownikom za wyjątkiem tych,

którzy posiadają prawa „SalesDirector”. Blokada dostępu do całego katalogu raportów, wymaga usunięcia nazwy pliku „SalesByTerritory.aspx”.

```
<location path="Administration/Reports/SalesByTerritory.aspx">
  <system.web>
    <authorization>
      <allow roles="SalesDirector"/>
      <deny users="*" />
    </authorization>
  </system.web>
</location>
```

Listing 42 - Konfiguracja, blokowanie dostępu do określonej strony, lub katalogu

8.3 Bezpieczeństwo głównej bazy danych

Uwierzelnianie użytkowników bazodanowych w systemie dokonywane jest poprzez autoryzację domenową (Windows Authentication). Dla każdego przyszłego użytkownika systemu niezbędne jest skonfigurowanie, jego konta domenowego z bazą danych. Przy tym procesie, jedną z podstaw systemów bezpieczeństwa jest stosowanie zasady jak najmniejszych uprawnień. Użytkownicy mają mieć dostęp, tylko do tych danych, do których mieć powinni, nawet można zastosować ograniczenie co do wykonywania tylko niezbędnych działań na nich (np. SELECT, ale nie UPDATE, czy DELETE).

W aspekcie bezpieczeństwa bazy danych, należy zwrócić także uwagę na bardzo popularną metodę uzyskania nieautoryzowanego dostępu, jaką jest wstrzykiwanie SQL (SQL Injection³⁷). Mechanizm ten może przyjmować wiele for, głównie wykorzystywany jest kod aplikacji, który używa dynamicznie budowanych łańcuchów i dodaje niespodziewany fragment do tej konstrukcji. Zastosowanie narzędzia do mapowania relacyjno-obiektowego, jest dobrą praktyką przeciwdziałania atakom tego typu.

8.4 Zabezpieczenie przed awariami

Mimo zastosowania wielu zabezpieczeń technicznych oraz środków pomocniczych, nie zawsze udaje się zapobiec totalnej awarii systemu informatycznego. Jednakże, zautomatyzowany przebieg przygotowanych i wypróbowanych planów awaryjnych może znacząco zmniejszyć szkody.

Przyczyną tak zwanej „katastrofy informatycznej” (całkowitego zaprzestania pracy systemu informatycznego) mogą być katastrofy naturalne, takie jak powódź, czy trzęsienie ziemi. Są to jednak czynniki stanowiące jedynie mały procent wszystkich przypadków utraty danych. Natomiast błędy w oprogramowaniu, czy sprzęcie to przyczyna niemal połowy wszystkich katastrof informatycznych.

³⁷ SQL Injection - http://pl.wikipedia.org/wiki/SQL_injection

Skutki awarii systemu, która nie zostaje rozwiązana przez dłuższy czas (kilka dni) mogą doprowadzić do całkowitego upadku firmy. Dlatego też działania na rzecz zapobiegania katastrofom informatycznym, winny znaleźć się w centrum wszelkich wysiłków. W rzeczywistości, wiele przyczyn utraty danych można wyeliminować poprzez odpowiednie działania techniczno-organizacyjne. Zastosowanie macierzy RAID³⁸, zasilania awaryjnego UPS³⁹ wykluczają, lub w znacznym stopniu minimalizują klasyczne awarie sprzętu. Ważne są także zabezpieczenia programowe, takie jak odpowiednio skonfigurowane oprogramowanie do wykonywania okresowych kopii zapasowych newralgicznych danych, czy też dobry system antywirusowy oraz firewall do zabezpieczenia przez infekcją z Sieci.

8.5 Podsumowanie

Podczas projektowania zarówno aplikacji mobilnych jak i innych aplikacji działających w środowisku firmowym czy korporacyjnym aspekt zachowania bezpieczeństwa, poufności oraz integralności informacji jest na pewno jednym z głównych aspektów, które powinny być rozpatrywane już nawet w fazie projektowania systemu. Spychanie tego rodzaju kwestii na margines tłumacząc się nawet brakami budżetu może prowadzić do późniejszych jeszcze większych strat zarówno materialnych jak i nawet moralnych. Zarówno system operacyjny jak i komponenty firm trzecich często umożliwiają łatwe dostosowania aplikacji w kierunku uodpornienia na wiele zagrożeń płynących z zewnątrz, trzeba tylko odpowiednie mechanizmy stosować w sposób przemyślany.

³⁸ RAID - <http://pl.wikipedia.org/wiki/RAID>

³⁹ UPS - http://en.wikipedia.org/wiki/Uninterruptible_power_supply

9. Testowanie

Testowanie jest procesem mającym na celu sprawdzenie oprogramowania pod kątem zweryfikowania, czy spełnia ono określone wymagania oraz, czy nie zawiera błędów. Systemy informatyczne są na tyle złożone, że praktycznie niemożliwym jest stworzenie większego oprogramowania, które nie zawierałoby błędów. Testowanie oprogramowania urosło do rozmiarów dziedziny problemowej, której poświęcana jest coraz większa liczba książek, czy publikacji. Niniejszy rozdział poświęcony jest krótkiemu opisowi małego fragmentu tej dziedziny oraz omówieniu procesu testowania zaprojektowanego systemu sprzedaży.

W procesie testowania można wymienić różne rodzaje przeprowadzanych testów, takie jak np.:

- Testy funkcjonalne – mają na celu sprawdzenie czy aplikacja spełnia założone wymagania - innymi słowy, czy umożliwia poprawne wykonanie wszystkich pożądaných i ujętych w ramach kontraktu operacji i funkcji.
- Testy użyteczności – pozwalają ocenić stopień zadowolenia użytkownika końcowego z pracy z aplikacją. Na użyteczność składa się prostota nauki i obsługi systemu, wygoda i szybkość użytkowania, wydajność.
- Testy akceptacyjne – przeprowadza się je w celu ostatecznej weryfikacji funkcjonalności systemu przed wdrożeniem na produkcję. Testy te z reguły wykonuje wybrana grupa użytkowników końcowych na podstawie opracowanych scenariuszy. Często w testach tych uczestniczą także specjaliści z branży danego biznesu (np. analitycy biznesowi) – celem ostatecznego potwierdzenia poprawności systemu.

9.1 WebAii – testowanie aplikacji ASP .NET

„WebAii Automation Framework”⁴⁰ jest darmową biblioteką dedykowaną aplikacjom napisanym w ASP.NET, służącą automatyzacji testów stron WWW. Framework pozwala na sprawdzanie dostępności i widoczności kontrolek HTML/ASP.NET, wywoływania funkcji JavaScript, śledzenie renderingu elementów strony, ustawiania wartości kontrolek czy wywoływanie zdarzeń (np. wciśnięcia określonego przycisku). Możliwości te pozwalają, np. na opracowanie programu symulującego działania użytkownika, który może sprawdzić poprawność funkcjonowania poszczególnych głównych ścieżek aplikacji.

⁴⁰ WebAii Automation Framework - <http://www.artoftest.com/home.aspx>

Pierwszym krokiem, który należy wykonać jest pobranie i zainstalowanie platformy do testów ze strony producenta (<http://www.artoftest.com/>). Po dokonaniu tej czynności w środowisku Visual Studio powinny się pojawić nowe typy elementów możliwych do dodania oraz doinstalowane zostają wtyczki do zainstalowanych przeglądarek. Jednakże, często ta konfiguracja nie wystarcza i trzeba dokończyć ją „ręcznie” kierując się instrukcjami zawartymi w dokumentacji użytkownika. Szczególnie należy zwrócić uwagę na wersję używanego systemu operacyjnego oraz typ i wersję przeglądarki. Do celów testowych najlepiej użyć innej przeglądarki niż wykorzystywanej na co dzień do korzystania z Internetu. Fakt ten jest istotny z racji tego, że podczas konfiguracji przeglądarki wyłączane są funkcje mające spory wpływ na bezpieczeństwo.

Przykład testu – logowanie użytkownika do aplikacji WWW

Możliwości platformy WebAii zostaną pokazane na przykładzie implementacji testu logowania użytkownika do systemu. Do utworzonego projektu „Test” należy dodać nowy element New Item-> Test -> WebAii 2.0 -> VsUnit. Tworzona jest klasa testowa, wykorzystująca wbudowany w Visual Studio 2008 moduł do testów. Najważniejsze metody klasy to:

- MyTestInitialize() – konfiguracja ustawień globalnych dla testów,
- MyTestCleanUp() – metody wywoływana po testach, wyłącza menedżera testów, zamyka przeglądarkę itp.

Metoda inicjalizacyjna została przedstawiona na listingu 43. Wykonywane są w niej takie operacje jak wybranie typu serwera, ustawienie ścieżki do aplikacji internetowej czy ustawienie ścieżki do pliku z logami.

```
[TestInitialize()]
public void MyTestInitialize()
{
    #region WebAii Initialization

    Settings settings = GetSettings();
    settings.LocalWebServer = LocalWebServerType.AspNetDevelopmentServer;
    settings.WebAppPhysicalPath =
        @"D:\Tommy\Studia\Magisterka\Solucja\SfaSystem[NHibernate]\Web";
    settings.LogLocation = TestContext.TestLogsDir;
    Initialize(settings, new TestContextWriteLine(this.TestContext.WriteLine));

    SetTestMethod(this, (string)TestContext.Properties["TestName"]);

    #endregion
}
```

Listing 43 - Metoda inicjalizacyjna

Sama funkcja testująca logowanie użytkownika została przedstawiona na listingu 44. Metoda ta podczas wywołania, uruchamia aplikację WWW systemu sprzedaży w przeglądarce. Następnie przechodzi do adresu strony z formularzem logowania i poprzez pobranie kontrolek pól tekstowych, za pomocą zdefiniowanych identyfikatorów, wypełnia je podanymi danymi. Na koniec wywołuje zdarzenie przyciśnięcia dostępnego przycisku logowania i sprawdza, czy strona została zmieniona na pulpit przedstawiciela handlowego, dostępny tuż po zalogowaniu dla pracownika o takiej roli.

```
[TestMethod]
[Description("Test logowania przedstawiciela handlowego.")]
[DeploymentItem(@"", "AspNetApp")]
public void CheckSalesRepLogIn()
{
    // Inicjalizacja aplikacji ASP .NET
    Manager.LaunchNewBrowser();
    //Nawigacja do strony logowania
    ActiveBrowser.NavigateTo("~/Login.aspx");
    if (ActiveBrowser.BrowserType == BrowserType.AspNetHost)
    {
        AspNetHostBrowser hostBrowser = (AspNetHostBrowser)ActiveBrowser;
        Assert.IsTrue(hostBrowser.Status == 200);
    }

    HtmlInputText txtLogin =
        Manager.ActiveBrowser.Find.ById<HtmlInputText>("~/UserName");
    HtmlInputPassword txtPassword =
        Manager.ActiveBrowser.Find.ById<HtmlInputPassword>("~/Password");

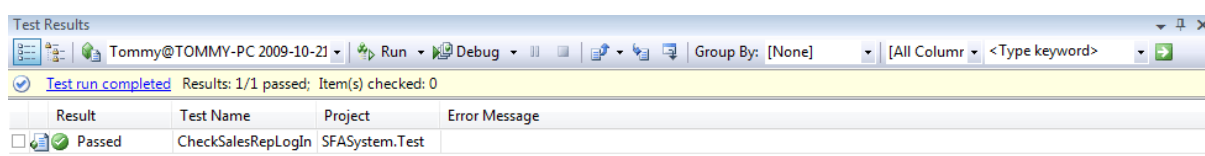
    txtLogin.Text = "salesrep";
    txtPassword.Text = "sr";

    HtmlInputSubmit button =
        Manager.ActiveBrowser.Find.ById<HtmlInputSubmit>("~/LoginButton");
    button.Click();

    Assert.IsTrue(ActiveBrowser.Url.Contains("SalesRepresentative/Dashboard.aspx"));
}
}
```

Listing 44 - Test logowania przedstawiciela handlowego

Po wykonaniu testów przy użyciu środowiska Visual Studio, można sprawdzić ich status w widoku „Test Result” (Rysunek)



Rysunek 55 - Widok Test Result - Visual Studio

Tego typu testy pozwalają zastąpić „żywego” użytkownika automatem, który będzie sprawdzał, czy funkcjonalność oraz działanie aplikacji nie zostały w jakimś stopniu zaburzone ciągle wprowadzanymi zmianami do kodu programu.

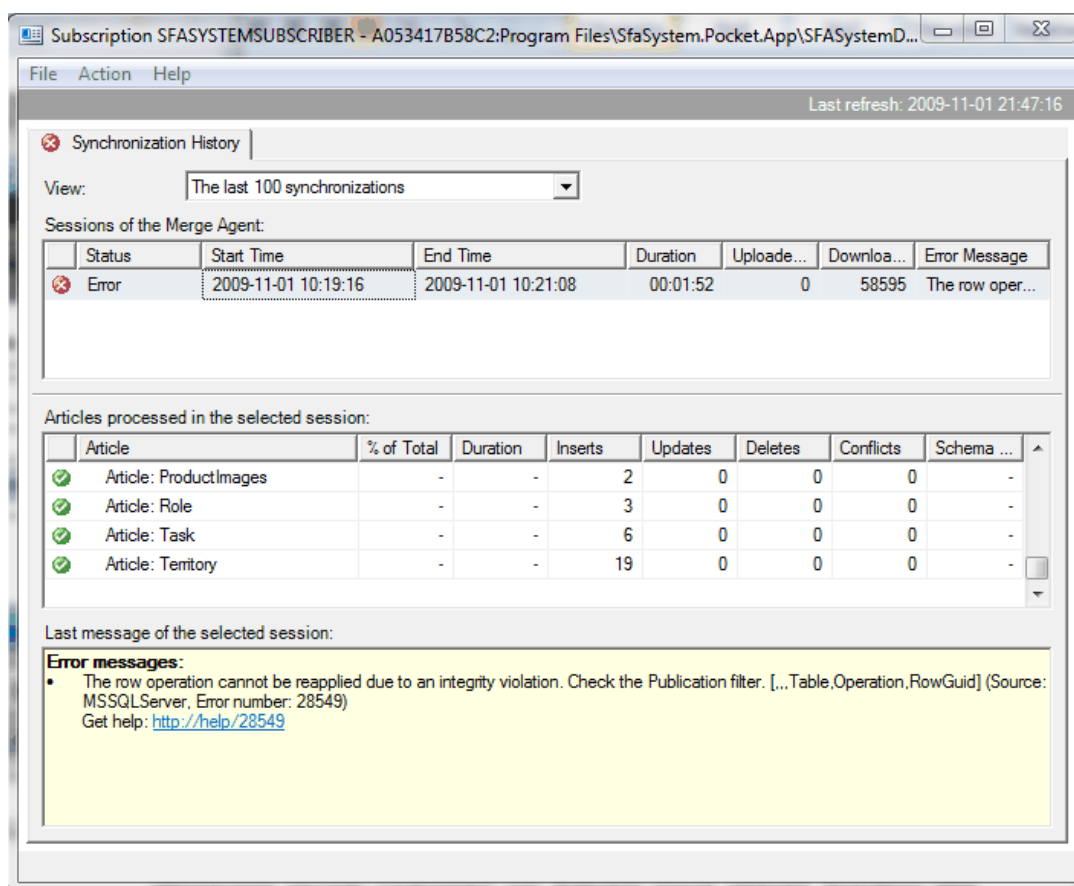
9.2 Testowanie replikacji

Proces testowania replikacji rozpoczęty został niezwłocznie po zaimplementowaniu wspomnianego w rozdziale specyfikacji wewnętrznej menadżera synchronizacji.

Działanie replikacji przetestowano przy użyciu zaimplementowanych aplikacji. Dla przykładu, po wprowadzeniu zamówienia i synchronizacji, sprawdzano jego istnienie z poziomu aplikacji internetowych. W procesie tym wykorzystano także możliwość podłączenia się bezpośrednio do mobilnej bazy danych, poprzez oprogramowanie SQL Server Management Studio. Oczywiście, w takim przypadku urządzenie, musi zostać podłączone do komputera i zsynchronizowane przy użyciu Windows Mobile Device Center (ActiveSync w systemach starszych od Windows Vista). Jeżeli używany jest symulator urządzenia z poziomu Visual Studio 2008 SP1 (Service Pack 1), należy doinstalować Microsoft SQL Server Compact 3.5 Service Pack 1 for Windows Mobile⁴¹. Po podłączeniu się do mobilnej bazy danych, w celu odrębnego od aplikacji przetestowania procesu replikacji, dodawano, usuwano i modyfikowano rekordy za pomocą prostych zapytań. Ze względu na to, że możliwa jest także synchronizacja z poziomu środowiska Management Studio, w tym środowisku sprawdzono również, czy ustawione mechanizmy filtrowania oraz rozwiązywania konfliktów działają poprawnie. Pierwsze z testów, zwróciły uwagę na problem zachowania unikalnego identyfikatora dla wierszy na poziomie całego systemu. Początkowo identyfikatory były typu numerycznego, nadawanego automatycznie przez bazę danych, co stanowiło problem. Rozwiązano go poprzez zastosowanie typu uniqueidentifier, którego generator, zawsze utworzy inną wartość.

Podczas poszukiwania przyczyn błędów występujących przy replikacji, jako niezastąpione okazało się dostępne w SQL Server Management Studio narzędzie Replication Monitor. Na rysunku 56 pokazano przykład błędu „wykrytego” przez ten program, dotyczącego źle skonfigurowanych filtrów.

⁴¹ Microsoft SQL Server Compact 3.5 SP 1 - <http://www.microsoft.com/downloads/details.aspx?displaylang=en&FamilyID=fce9abbf-f807-45d6-a457-ab5615001c8f>



Rysunek 56 - Szczegóły synchronizacji - Replication Monitor

Problem z filtrami replikacji związany był z więzami referencyjnymi do nieistniejących w bazie mobilnej elementów. Zostały pobrane wszystkie faktury, lecz nie wszystkie zamówienia (filtr na określonego pracownika), replikacja nie mogła pozwolić na istnienie faktur do nieistniejących zamówień. Dla usunięcia tego błędu należało rozszerzyć filtr na faktury.

9.3 Pozostałe testy

Pozostałe testy wykonywane były w miarę dodawania kolejnych funkcjonalności.

Aplikacja mobilna została gruntownie sprawdzona pod kątem wspomnianej replikacji, zabezpieczenia pierwszej synchronizacji oraz dodawania nowych zamówień. Sprawdzone także poprawność wyświetlanych danych pod kątem wprowadzanych filtrów (dotyczy to także aplikacji internetowej) oraz utworzono dziesiątki testowych zamówień. Podczas testowania aplikacji mobilnej usunięto, takie problemy jak, np. zbyt wolne wczytywanie okna detali klienta czy niedokładne liczenie wartości brutto, netto oraz należnego podatku dla zamówienia.

Aplikacja internetowa powstająca równolegle z mobilną, także przechodziła intensywne testy. Z racji bardziej rozbudowanej funkcjonalności, szczególnie w aspekcie wprowadzania oraz modyfikowania danych, niezbędne było gruntowne sprawdzenie działania walidacji. Proces ten przetestowany został pod różnym kątem, a w szczególności wprowadzania danych błędnych (np. zły format daty), za długich, czy też pozostawiania obligatoryjnych pól pustych. Poprzez symulację błędów, sprawdzono proces ich obsługi oraz zapisywania danych do pliku z logiem.

Do aplikacji internetowej dostęp posiadają użytkownicy o różnych rolach, mający różne poziomy uprawnień. Dlatego, też przetestowano mechanizm blokowania nieupoważnionym użytkownikom dostępu do poszczególnych stron oraz funkcjonalności (np. możliwości dodawania, edycji niektórych danych).

Po usunięciu wszystkich krytycznych błędów i przetestowaniu głównych ścieżek aplikacji idealnym rozwiązaniem, jest przekazanie produktu zespołowi testerów lub użytkowników końcowych po stronie klienta, wdrażającego system. Pozwoliłoby to na wykrycie głównie większej liczby błędów funkcjonalnych, czy uchybień dotyczących specyfikacji. Niestety w przypadku tego systemu, proces ten jeszcze nie miał miejsca. Obie aplikacje w głównej mierze testował autor oraz osoby poproszone o sprawdzenie wygody obsługi systemu i udzielenie ogólnej opinii o nim.

10. Podsumowanie

W ramach wykonanej pracy dyplomowej przedstawiono wiedzę z zakresu charakteru pracy działu handlowego przedsiębiorstwa, a w szczególności przedstawicieli pracujących w bliskich relacjach z klientami firmy. Zebrane wiadomości posłużyły zaprojektowaniu oraz zaimplementowaniu kompleksowego rozwiązania służącego wsparciu sprzedaży. Utworzony system jest systemem rozproszonym, składającym się z centralnej bazy danych, aplikacji WWW dostępnej poprzez przeglądarkę internetową oraz mobilnych aplikacji klienckich przeznaczonych na urządzenia przenośne typu Pocket PC.

Stworzony system charakteryzuje się różnorodnością dostępnych opcji oraz funkcjonalności. Posiada wiele cech systemu klasy CRM takich jak, możliwość zarządzania klientami, pracownikami firmy czy realizowanymi zadaniami. Rozbudowany jest także o moduł sprzedaży, którego implementacja była jednym z głównych celów. Moduł ten, pozwala w szybki oraz wygodny sposób na zbieranie zamówień zarówno bezpośrednio w placówce odwiedzanego klienta jak i telefonicznie.

Zaprojektowany system może zostać wdrożony w dowolnej firmie zajmującej się dystrybucją, bądź sprzedażą, a szczególnie w firmach z branży FCMG (dóbr szybko zbywalnych). Zaprojektowane rozwiązanie może działać w trybie „samodzielnym”, lub zostać zintegrowane z systemem ERP działającym w firmie.

Niniejsza praca pozwoliła autorowi poszerzyć swoją wiedzę z zakresu programowania urządzeń mobilnych oraz zapoznać się z niestosowanymi wcześniej technikami służącymi testowaniu aplikacji internetowych. Rozwiązanie wielu problemów, występujących podczas realizacji projektu pozwoliło ugruntować posiadaną wiedzę oraz zdobyć cenne doświadczenie.

Architektura systemu została przemyślana w sposób pozwalający na stosunkowo łatwą jego rozbudowę, co stwarza duże możliwości dalszego rozwoju. Jednakże, związane jest to już ze współpracą z konkretną firmą wdrażającą projekt i jej oczekiwaniami. W przyszłości podstawowa wersja systemu mogłaby być rozszerzona o takie funkcjonalności, jak np., zarządzanie promocjami, materiałami marketingowymi czy też możliwość przeprowadzania łatwych w definiowaniu ankiet. Kolejną ciekawą, wartą dodania rzeczą byłoby wykorzystanie w aplikacji mobilnej systemu lokalizacji GPS w połączeniu z Google Maps, lub przy współpracy z firmą tworzącą rozwiązanie typu automapa. Pozwoliłoby to być może na inteligentne ustawianie kolejności odwiedzanych firm podczas codziennych wizyt przedstawiciela, lub w prostszej wersji automatyczne nawigowanie do firmy docelowej.

Bibliografia

1. Microsoft patterns&practices „Enterprise Solution Patterns Using Microsoft .NET”.
2. Jesse Liberty, Dan Hurwitz “Programowanie ASP.NET”.
3. Jay Hillyard, Stephen Teilhet „C# Receptury”.
4. Jesse Liberty “C# Programowanie”.
5. Robert C. Martin, Micah Martin „Agile Programowanie zwinne” – wzorce projektowe.
6. Jill Dyche „CRM. Zarządzanie klientami”.
7. Stanisław Wrycza, Bartosz Marcinkowski, Krzysztof Wyrzykowski „Język UML 2.0 w modelowaniu systemów informatycznych”.
8. J. Ullman, J. Widom „Podstawowy wykład z systemów baz danych”.
9. Robert Vieira „SQL Server 2005. Programowanie. Od podstaw”.
10. A. Wigley, S. Wheelwright, R. Burbidge, R. MacLoed, M. Sutton “Microsoft .NET Compact Framework (Core Reference)”
11. Microsoft Architecture Journal 14 „Mobile Architecture”.
12. Software Developer Journal 3/2009 „Testowanie oprogramowania”.
13. P. Henri Kuate, C. Bauer, G. King, T. Harris „NHibernate in Action”.
14. Jorg Luther „Ochrona przed katastrofą z planem” IDG Artykuł.
15. Grzegorz Mazur „Opracowanie system kompleksowej obsługi kliniki lekarskiej uwzględniającego współpracę z urządzeniami mobilnymi” praca dyplomowa.
16. <http://www.asp.net>
17. <http://www.artoftest.com/home.aspx>
18. <http://www.mostlyclean.com/category/NHibernate.aspx>
19. <http://www.simonrhart.com>
20. <http://www.iconfinder.net>
21. <http://www.codeproject.com>
22. <http://www.codeplex.com>
23. <http://www.nopcommerce.com>

Spis ilustracji

Rysunek 1 - Diagram przypadków użycia	13
Rysunek 2 - Architektura systemu.....	14
Rysunek 3 - Architektura wielowarstwowa.....	15
Rysunek 4 Logowanie - użytkownicy.....	16
Rysunek 5 Klienci, placówki, kontakty.....	17
Rysunek 6 Dane adresowe	18
Rysunek 7 Zadania pracowników	18
Rysunek 8 Przechowywanie informacji o produktach	19
Rysunek 9 Zamówienia, faktury	20
Rysunek 10 Tabele zawierające informacji słownikowe	21
Rysunek 11 - Ekran logowania.....	30
Rysunek 12 – Proces synchronizacji	30
Rysunek 13 - Menu główne	30
Rysunek 14 - Lista klientów	31
Rysunek 15 - Informacje o kliencie	31
Rysunek 16 - Lista placówek.....	32
Rysunek 17 - Informacje o placówce.....	32
Rysunek 18 - Lista kontaktów klienta.....	32
Rysunek 19 - Lista zamówień	32
Rysunek 20 - Lista płatności	32
Rysunek 21 - Grupy produktów.....	33
Rysunek 22 - Lista produktów	33
Rysunek 23 - Detale produktu.....	33
Rysunek 24 - Lista zadań	34
Rysunek 25 - Szczegóły zadania	34
Rysunek 26 - Kompletacja zamówienia.....	35
Rysunek 27 - Składanie zamówienia	35
Rysunek 28 - Ustawienia konta osobistego.....	36
Rysunek 29 - Logowanie, aplikacja WWW	37
Rysunek 30 - Wygląd aplikacji WWW.....	38
Rysunek 31 - Menu główne.....	39
Rysunek 32 - Pulpit przedstawiciela handlowego.....	40
Rysunek 33 - Pulpit przedstawiciela regionalnego.....	41
Rysunek 34 - Pulpit dyrektora handlowego	41
Rysunek 35 - Dodawanie danych przykład.....	42
Rysunek 36 - Walidacja danych.....	42
Rysunek 37 - Okno edycji produktu	43
Rysunek 38 - Produkt dodawanie zdjęcia	43
Rysunek 39 – Okno modalne - wprowadzanie informacji o kontakcie klienta	44
Rysunek 40 - Okno modalne - anulowanie zamówienia	44
Rysunek 41 - Informacja "Tryb zamówienia"	45
Rysunek 42 - Proces dodawania produktów do zamówienia	45

Rysunek 43 - Wybieranie placówki dla zamówienia.....	46
Rysunek 44 - Ekran kompletowania zamówienia	46
Rysunek 45 - Zatwierdzenie zamówienia	47
Rysunek 46 - Email realizacja zamówienia	48
Rysunek 47 - Lista faktur	48
Rysunek 48 - Wydruk faktury VAT	49
Rysunek 49 - Raport sprzedaży - grupy produktów.....	50
Rysunek 50 - Zakładki kontrolki TabControl	56
Rysunek 51 - Raport, zakładka Data	66
Rysunek 52 - Raport, zakładka Layout.....	66
Rysunek 53 - SQL Server Agent Diagnoza	73
Rysunek 54 – modelowanie zagrożeń	76
Rysunek 55 - Widok Test Result - Visual Studio	87
Rysunek 56 - Szczegóły synchronizacji - Replication Monitor	89