POLITECHNIKA CZĘSTOCHOWSKA

Wydział Inżynierii Mechanicznej i Informatyki

The Faculty of Mechanical Engineering and Computer Science

Praca dyplomowa inżynierska

Etapy projektowania, testowania i wdrażania bazy danych, na przykładzie aplikacji wspomagającej sprzedaż samochodów.

Stages of project designs, testing and accustoming of database, on example of application helping car sale.

Bartosz Lewiński

Numer albumu: 109070

Kierunek: Informatyka

Studia: niestacjonarne

Promotor:
Dr inż. Olga Siedlecka-Lamch

Praca przyjęta dnia:

Podpis promotora:

SPIS TREŚCI

SPIS RYSUNKÓW

1. Wstęp

Niniejsza praca jest próbą odpowiedzi na realne zapotrzebowanie, faktycznie istniejącego przedsiębiorstwa, w którym pracuję na stanowisku dyrektora ds. finansowych. Jest to przedsiębiorstwo zajmujące się sprzedażą samochodów osobowych (dealer). W przedsiębiorstwie, jak dotychczas nie ma żadnego dedykowanego oprogramowania, które pomogłoby nadzorować, opisywać oraz usprawniać proces sprzedaży. W świecie, w którym procesy gospodarcze w dynamicznie rozwijającym się przedsiębiorstwie muszą być kontrolowane i zarządzane w sposób profesjonalny, brak rozwiązań z dziedziny IT jest palącym problemem, dopominającym się rozwiązania. Jedną z odpowiedzi na to naturalne zapotrzebowanie jest prezentowana aplikacja bazodanowa. Jest ona zrealizowana w oparciu o faktyczne doświadczenia dotyczące działania organizmu gospodarczego oraz o istniejące zapotrzebowanie – zdiagnozowane i zrozumiane.

1.1 Cel pracy

Celem pracy jest stworzenie od podstaw aplikacji wspierającej sprzedaż samochodów. Pomysł na jej stworzenie jest rezultatem pracy autora w przedsiębiorstwie zajmującym się taką sprzedażą. Jest wynikiem obserwacji potrzeb tego przedsiębiorstwa w kontekście stale zmieniającej się sytuacji ekonomicznej. Jest również wynikiem dogłębnej analizy i znajomości owego przedsiębiorstwa.

Innym, nie mniej istotnym celem, jest analiza samego procesu tworzenia aplikacji, zbadanie problemów związanych z różnymi etapami wytworzenia oprogramowania. Proces tworzenia, testowania i wdrożenia oprogramowania do istniejącego przedsiębiorstwa, jest skomplikowany, dlatego podczas jego tworzenia można odkryć rzeczy, których na etapie planowania, być może, nie jesteśmy w stanie przewidzieć.

1.2 Zawartość pracy

Niniejsza praca składa się z czterech rozdziałów. W pierwszym jest opisany cel pracy oraz jej struktura. W drugim jest wstęp teoretyczny oraz opisane są etapy projektowania bazy danych i aplikacji. W rozdziale trzecim opisany jest proces projektowania bazy danych i aplikacji – środowisko pracy, oraz proces realizacji aplikacji. W rozdziale czwartym opisano metody użyte do testowania oraz proces wdrażania systemu bazodanowego. W piątym rozdziale znajduje się podsumowanie pracy, wnioski oraz stopień realizacji zamierzonego działania.

2. Etapy projektowania aplikacji bazodanowej

Zgodnie z zaproponowaną w [1] metodologią, projektując bazę danych będziemy się posługiwać pewnym schematem działania, porządkującym pracę. Należy dochować starań by baza była zaprojektowana poprawnie pod względem formalnym jak i użytkowym. Baza powinna być zgodna z określonym modelem danych, w miarę potrzeb i możliwości bez redundancji. Dane powinny być spójne, niezależne i trwałe. Powinna również zapewnić możliwość przetwarzania danych, do celów, określonych w wymaganiach i specyfikacji. Etapy projektowania bazy zostaną przedstawione w pkt. 2.2.

2.1 Wstęp teoretyczny

Baza danych jest modelem pewnego wycinka rzeczywistości. Możemy definiując to pojęcie próbować spojrzeć na całe zagadnienie z różnych punktów widzenia, w zależności od tego, jakie cele, po co lub jak chcemy je osiągnąć. Możemy patrzeć na bazę danych, jako na zestaw danych i metadanych, które zachowują spójność, mają określoną strukturę, w której jest zaimplementowana jakaś metoda wyszukiwania i aktualizacji tychże danych. Możemy zwrócić uwagę na cel, do którego została stworzona, akcentując jej aspekt pragmatyczny.

Uszczegóławiając opis możemy rozważać kolejne bardziej subtelne, lub bardziej istotne, dla danego opisu, cechy lub funkcjonalności.

Na przykład Wikipedia podaje: „Baza danych – zbiór danych zapisanych zgodnie z określonymi regułami” [2], a dalej następuje opis wielu cech i właściwości, które baza mieć powinna, zatem jest to definicja poprzez opis środowiska, w którym baza istnieje.

Z kolei portal Wazniak podaje, że: „Zbiór danych opisujący pewien wybrany fragment rzeczywistości będziemy nazywać bazą danych” [3]

Tak, czy inaczej, musimy stwierdzić, że baza danych jest pewnego rodzaju sposobem organizacji patrzenia na otaczającą nas rzeczywistość, sposobem kolekcjonowania faktów i wydarzeń związanych z wybranym fragmentem rzeczywistości. W związku z tym w zależności od tego sposobu patrzenia, również sposób tworzenia takiej bazy danych będzie różny. Istnieje zatem kilka paradygmatów tworzenia baz danych, zwanych modelami. Przyjrzymy się im pokrótce.

Istotne jest również spostrzeżenie, że baza danych nie jest tworem samoistnym, a istnieje w pewnym środowisku, które nazywamy Systemem Zarządzania Bazą Danych (SZBD). Do najpopularniejszych należą MySQL, PostgreSQL, Oracle, Sysbase, IBM DB2 i Microsoft SQL Server. W niniejszej pracy będziemy korzystać z SZBD MySQL.

Opiszemy też podstawowe własności i pojęcia, które są immanentną częścią tej dziedziny wiedzy, jaką są bazy danych.

2.1.1 Języki baz danych

Do opisu, tworzenia i zarządzania bazą danych jest potrzebny język. W tym celu stworzono Strukturalny Język Zapytań (SQL – Structured Query Language). SQL jest tzw. językiem „deklaratywnym", który działa w następujący sposób; piszemy zapytanie do sytemu zarządzającego bazą danych (SZBD) deklarując co chcemy uzyskać i nie interesuje nas w jakich krokach odpowiedź zostanie uzyskana, SZDB sam znajdzie najlepszy dla siebie sposób wykonania. SQL to język, w skład którego wchodzą następujące moduły, pozwalające na wykonywanie działań na BD:

A) DML (ang. Data Manipulation Language – Język Manipulacji Danymi) umożliwia wykonywanie różnych działań na danych w bazie. Zawiera m.in. polecenia SELECT (wyszukiwanie, podstawa „zapytania"), UPDATE (uaktualnienie), INSERT (wstawianie), DELETE (kasowanie)
B) DDL (ang. Data Definition Language – Język Definiowania Danych) pozwala na tworzenie i kasowanie struktur danych (np. tabele, widoki itp.), zawiera m.in. polecenia CREATE (utwórz), DROP (usuń), ALTER (zmień)
C) DCL (ang. Data Control Language – Język Kontroli danych) pozwala zarządzać bazą. Zawiera m.in. polecenia GRANT (nadaje uprawnienia użytkownikom), REVOKE (odbiera uprawnienia), COMMIT (zatwierdza wprowadzone dane), ROLLBACK (cofa wprowadzone dane)
D) DQL (ang. Data Query Language – Język Zapytań o dane) – Wikipedia [2] wyróżnia dodatkowo ten język (polecenie SELECT) jako fragment DML.

2.1.2 Modele baz danych

Modelem bazy danych nazywamy sposób spojrzenia na organizację danych w bazie, jej architektury, relacji pomiędzy poszczególnymi elementami składowymi. Co za tym idzie są różne teorie matematyczne stojące za tymi modelami, a w konsekwencji również języki przy pomocy których tworzony jest dany model. Technicznie model bazy danych musi opisywać zasady, na podstawie których jest implementowana struktura danych w bazie, jakie mogą być rodzaje tych danych (wielkość, ilość, np.: wartość pesel to 11 znaków a imię to 32 znaki), jakie są ich wzajemne relacje i zależności. Określa również, jakie można na nich wykonać działania.

<u>Główne modele:</u>

A) Model hierarchiczny – przestarzały

   Dane są przechowywane w formie związków rekordów nadrzędny – podrzędny (korzeń-gałąź), których najczęstszą relacją jest jeden do wielu lub wiele do jeden. Taka struktura danych występuje np., w komputerach – katalog główny zawiera wiele katalogów podrzędnych.

B) Model sieciowy (drzewiasty) – również przestarzały

Jest to modyfikacja modelu hierarchicznego. Jak sama nazwa wskazuje, jest on przekształcony do postaci drzewiastej. Pozwala na obrazowanie relacji „wiele do wiele". Składa się z dwóch głównych rodzajów elementów: rekordów (zawierających dane), oraz zbiorów (określają relacje miedzy danymi)

C) Model relacyjny

Ten model będzie użyty w niniejszym projekcie. Jest to najpopularniejszy model, najlepiej opisany i udokumentowany, jednocześnie najbardziej intuicyjny. A zarazem wystarczający do tego projektu.

Bardzo użyteczną definicję z praktycznego punktu widzenia umieszcza w swojej książce Jason Price: „Relacyjna baza danych jest zbiorem powiązanych informacji, umieszczonych w tabelach. Dane w tabeli są przechowywane w wierszach i uporządkowane w kolumnach. Tabele są przechowywane w schematach baz danych" [4]

Inne właściwości opisuje dr. inż. Olga Siedlecka-Lamch w [1]: „Dane są prezentowane jako relacje. Relacje posiadają atrybuty o wartościach atomowych. Wykorzystuje operatory algebry relacyjnej. Zapewnia integralność danych poprzez zastosowanie kluczy."

D) Model obiektowy

Baza jest traktowana jako zbiór obiektów. Każdy z obiektów (podobnie jak w językach obiektowych, np.: C++, Java – w których mogą być tworzone obiekty bazy) ma zaimplementowane pewne metody i pola.

E) Model obiektowo – relacyjny

Jak sama nazwa wskazuje, jest to model łączący w sobie cechy (zalety i wady) modeli relacyjnego i obiektowego.

F) Model semistrukturalny

Model oparty na grafach, w którym struktura sama się definiuje (np. format XML). Patrz [1].

2.1.3 Architektura komunikacyjna bazy danych

Architektura komunikacyjna definiuje sposób, w jaki klient ma dostęp do zasobów bazy. Początkowo była to architektura jednowarstwowa (czyli BD aplikowana na komputerze klienta) ewoluowała do struktur bardziej złożonych, pozwalających na dostęp do danych na serwerze przez wiele hostów (komputerów klienckich) jednocześnie. Wyróżniamy dwie podstawowe architektury.

a) Typu „klient – serwer”
   Jest to architektura w której klient poprzez aplikację łączy się z serwerem, na którym BD jest fizycznie przechowywana. Niniejsza aplikacja będzie utworzona w tej architekturze.

b) Typu „trójwarstwowego”
   Architektura, w której pomiędzy aplikacją klienta a serwerem jest jeszcze serwer aplikacji

2.1.4 Podział systemów baz danych

Ze względu na różne użycie, cel, model, sposób reprezentacji danych itd. możemy dokonać podziału baz danych. Najpopularniejsze sposoby to, ze względu na:

a) użyty model logiczny: relacyjne, obiektowe itd.
b) liczbę węzłów: scentralizowane, rozproszone,
c) zastosowanie: OLTP (On-Line Transaction Processing), OLAP (On-Line Analytical Processing), CAD (Computer Aided Design), GIS (Geographical Information Systems), CASE (Computer Aided Software Engineering)
d) rozmiar przechowywanych danych i bazy danych, hurtownie danych, Big Data.

2.1.5 O PHP

Jak podaje [5] PHP jest językiem skryptowym, interpretowanym przez darmowy interpreter PHP. Nie wymaga zatem kompilacji. Jego ogromna popularność jest spowodowana łatwością, z jaka można się go nauczyć, oraz tym że efekty pracy można zobaczyć praktycznie od razu korzystając z dowolnej przeglądarki internetowej.

Język powstał w roku 1994 (jego pierwsza wersja) jako PHP/FI (Personal Home Page / Forms Interpreter) stworzony przez Rasmusa Lerdorfa. Od 1998, kiedy powstała jego trzecia wersja rozpoczął się jego dynamiczny rozwój. Obecnie jest używana wersja 5, ale od 2005 coraz szerzej słyszy się o pojawieniu się kolejnej – szóstej wersji.

Ogromną zaletą, oprócz wymienionych wcześniej, jest wsparcie społeczności internetowej – dokumentacja (php.net), rozmaite tutoriale w formie filmów na YouTube, książki i opracowania. Praktycznie każdy dostawca usług internetowych dostarczający serwery WWW obsługujące interpretery PHP. Język ten daje bardzo dużą uniwersalność. Pozwala tworzyć serwisy internetowe, własne rozwiązania dla serwerów WWW, FTP.

Ważna cechą PHP, jest łatwość tworzenia zmiennych. W językach takich jak C++, java, C# jest konieczność deklarowania zmiennych. Powoduje to częste błędy podczas kompilacji. Tymczasem w PHP zmienna jest tworzona tam, gdzie jest potrzebna, może zawierać dowolne

dane – od liczbowych po ciągi tekstowe (stringi). Oczywiście jest to również wada tego języka – konwersje typów mogą powodować trudne do wychwycenia błędy programu – wymaga to ogromnej dyscypliny przy tworzeniu kodu.

PHP dobrze współpracuje z innymi językami, dzięki temu w kodzie tworzonej strony WWW można wykorzystywać możliwości, jakie dają inne technologie. Popularnym zestawieniem jest „współpraca" PHP z HTML, javascript oraz bibliotekami, jak na przykład jQuery, extJS i innymi.

2.2 Etapy projektowania bazy danych

Aby poprawnie zaprojektować bazę danych, musimy przede wszystkim szczegółowo przemyśleć do czego ona ma służyć, jakie są uwarunkowania fizyczne i gospodarcze podmiotu, dla którego jest tworzona. Następnie, znając możliwości hardwar'u i software'u jakim dysponujemy, musimy zaplanować kształt systemu. Kroki te da się usystematyzować, np. zgodnie z koncepcją zaproponowaną w [1]:

2.2.1 Analiza wymagań dziedziny modelowej

Tworzona baza danych musi uwzględniać specyfikę przedsiębiorstwa dla którego jest tworzona. W tym przypadku jest to małe przedsiębiorstwo zatrudniające do 10 pracowników. Dwie osoby stanowią warstwę zarządzającą – powinny mieć zatem pełen dostęp do wytworzonych danych, możliwość dokonywania zmian merytorycznych w systemie, jak również czerpania zestawień i raportów dotyczących funkcjonowania firmy. Dwóch do trzech pracowników, stanowi pion sprzedażowy i powinni mieć możliwość zmian w zakresie swoich obowiązków – jak ewidencja sprzedaży, ewidencja klientów i/lub inne uprawnienia w zakresie swoich obowiązków związane z przydzielonymi przez zarząd obowiązkami (np. do tworzenia raportów). Pozostali pracownicy stanowią personel wykonawczy, który powinien mieć dostęp do bazy klientów firmy w zakresie tworzenia zleceń i zamówień na usługi warsztatowe.

W tak małym przedsiębiorstwie istotnym czynnikiem jest zmienność. Przydział pracowników – a zatem i uprawnień - powinien być możliwie elastyczny, np. kierownik może uczestniczyć w sprzedaży, sprzedawca może kierować zespołem mechaników i wyznaczać im cel pracy.

Ze względu na rodzaj personelu (mechanicy samochodowi) interfejs aplikacji powinien być prosty i nie powodować niepotrzebnych, zbyt rozbudowanych operacji logicznych.

Ponieważ firmą jest salon samochodowy, szata graficzna interfejsu powinna być możliwie związana z profilem przedsiębiorstwa, estetyczna, ale również ascetyczna, nie epatująca kolorem.

2.2.2 Wymagania funkcjonalne i niefunkcjonalne

Wymaganiami funkcjonalnymi nazywamy wymagania, jakie ma realizować system, przypadki użycia systemu – posługując się terminologią UML. Wymagania te wynikają bezpośrednio z analizy wymagań dziedziny modelowej oraz wiedzy praktycznej i teoretycznej, jakiej dostarczyć powinien zamawiający taką bazę.

Programista, który ma stworzyć taki system, zwykle nie jest zorientowany w specyfice przedsiębiorstwa, dla którego ma tworzyć aplikacje. Z kolei zamawiający często nie ma wiedzy na temat tworzenia aplikacji, zwykle nie ma również wiedzy na temat projektowania takiego systemu. Dlatego niezwykle ważne jest określenie języka, jakim porozumiewa się zamawiający z projektantem, wyznaczenie pewnych celów do których będą dążyć obie strony.

Jednym ze sposobów, po określeniu i zdefiniowaniu dziedziny modelowej (patrz 2.2.1), jest wyznaczenie konkretnych funkcjonalności, które system ma spełniać.

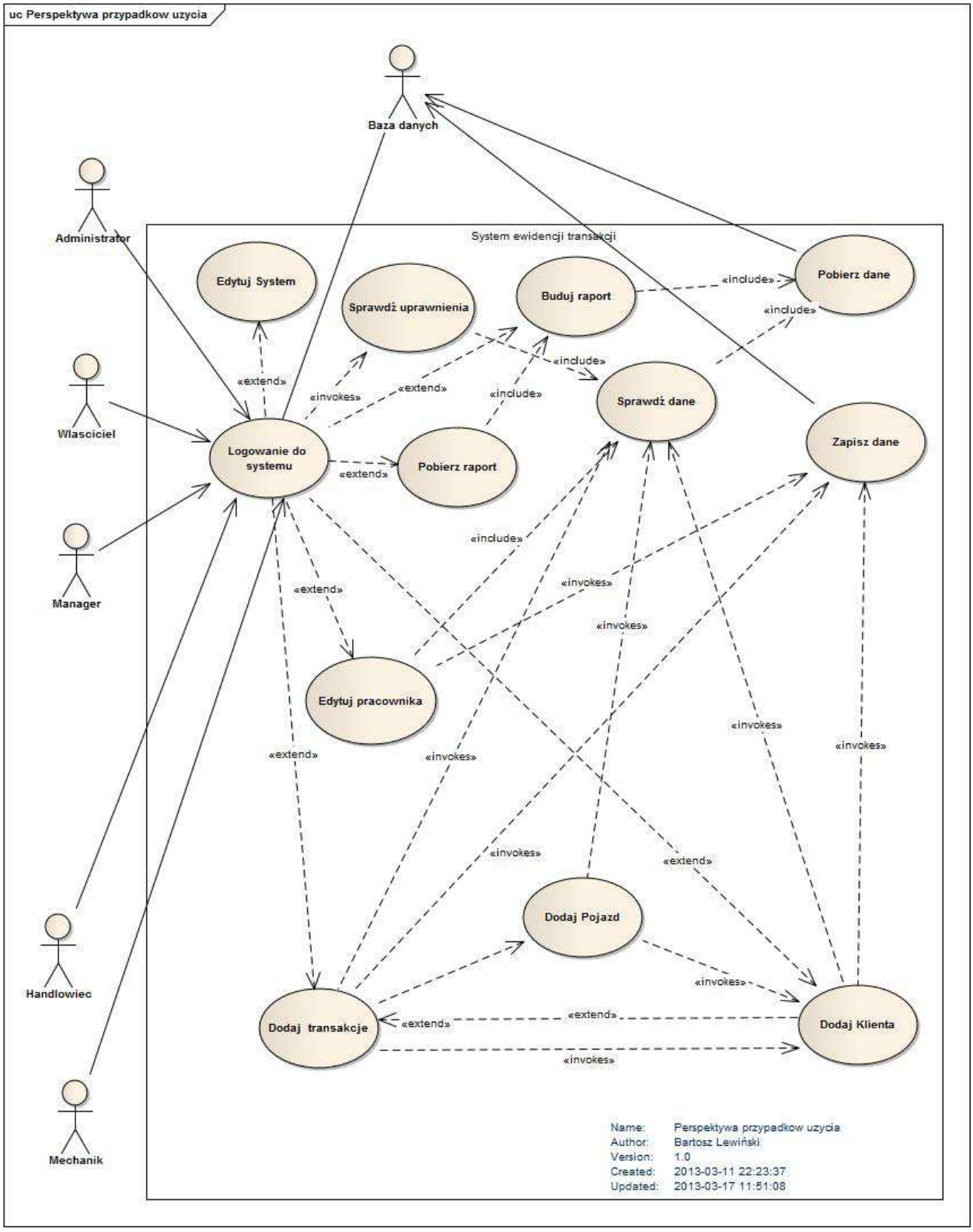
W przypadku tworzonej bazy będzie to :
- logowanie do systemu i walidacja uprawnień użytkowników
- rejestracja i edycja klientów
- rejestracja i edycja pracowników
- rejestracja i edycja pojazdów
- rejestracja i edycja transakcji
- rejestracja i edycja partnerów biznesowych

Strukturę powyższych wymagań pokazuje Rysunek 1

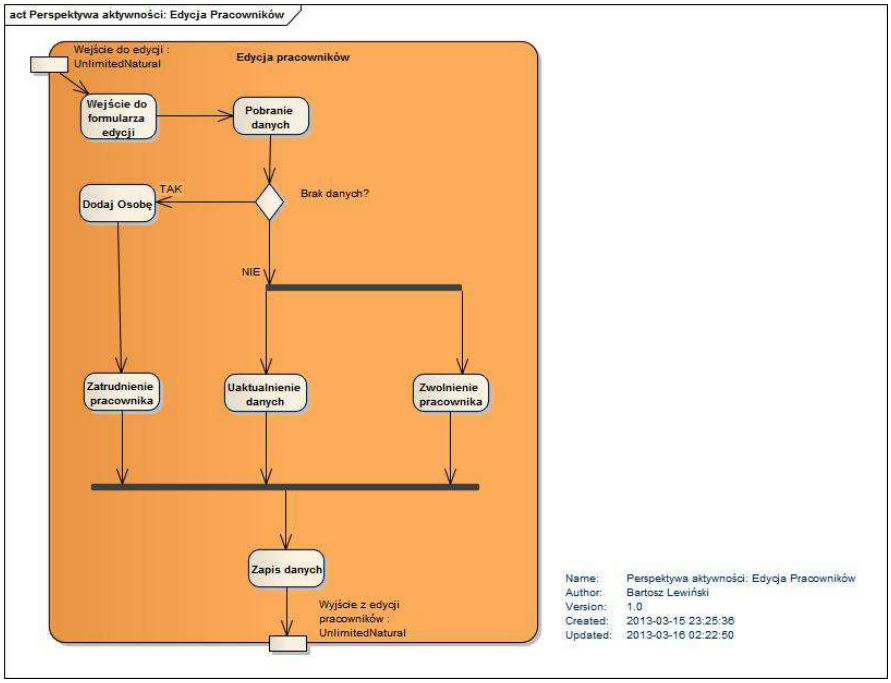
Każda z zarysowanych sytuacji powinna być skonkretyzowana i opisana w sposób, który umożliwia jej praktyczne wykonanie podczas kodowania. Powinno być także pomocne podczas późniejszej analizy gotowej aplikacji. Dobrze jest, jeśli jest elementem składowym dokumentacji technicznej, stanowiąc cenne źródło informacji o systemie dla przyszłych użytkowników, ale i serwisantów systemu. Poniżej przedstawiam przykładowe diagramy wytworzone dla poszczególnych procesów (Rysunek 2 i Rysunek 3).

Do wymagań niefunkcjonalnych zaliczamy wszystkie wymagania, które stawia potencjalny użytkownik systemu, które nie wynikają wprost z wymagań funkcjonalnych [6] Np. dotyczące szybkości i niezawodności systemu:

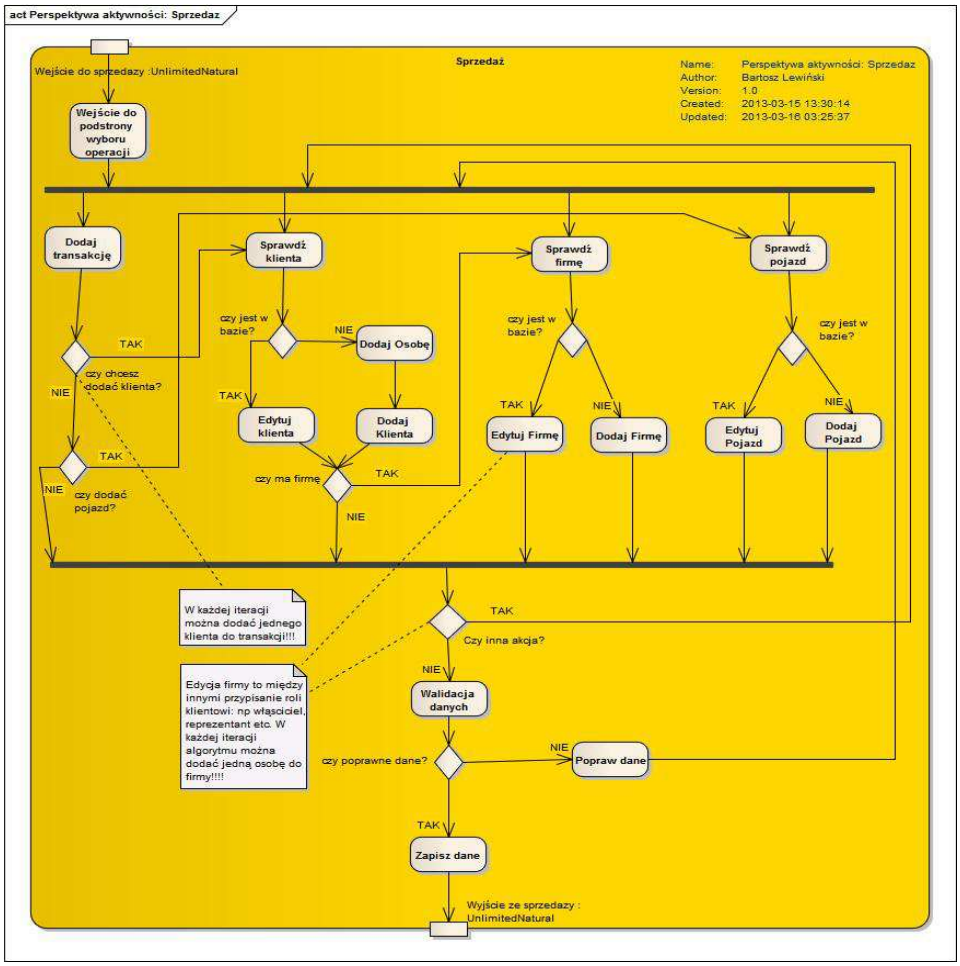
- Wprowadzenie nowego klienta - do 3 minut
- Wprowadzenie nowego pojazdu - do 3 minut
- Zapis nowej transakcji do - 4 minut
- Maksymalny czas szkolenia pracownika – 1 godzina



Rysunek 1 Perspektywa przypadków użycia



Rysunek 2 Perspektywa aktywności - Edycja Pracowników



Rysunek 3 Perspektywa aktywności - Sprzedaż

2.2.3 Modelowanie koncepcyjne i logiczne (UML, diagramy EER)

Jak podaje [1] modelowanie koncepcyjne (pojęciowe, semantyczne) jest przetworzeniem fragmentów rzeczywistości na ich abstrakcyjną reprezentację. Język takiego schematu pojęciowego daje semantyczne[1] i syntaktyczne[2] narzędzia służące do precyzyjnego opisu jego znaczeń. Model pojęciowy opisany za pomocą języka schematu pojęciowego nazywa się schematem pojęciowym. Taki opis można zrobić na przykład za pomocą:

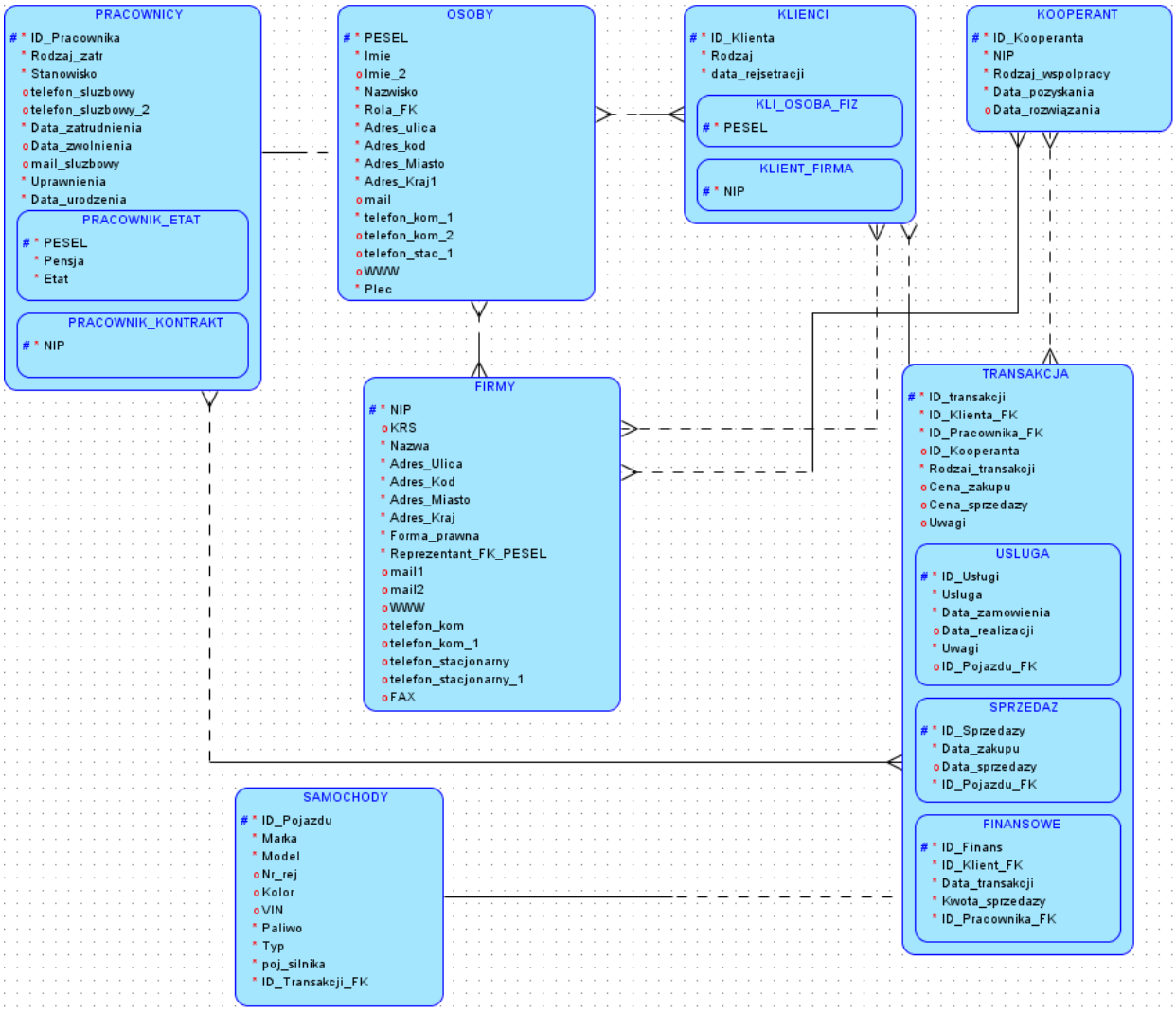
- diagramów związków encji (EER), taki przykładowy diagram to Rysunek 5. Ale też przykład diagramu klas na Rysunek 7
- diagramów UML – przykłady podane powyżej to Rysunek 2 oraz Rysunek 7

Kolejnym krokiem jest przejście do modelowania logicznego. O ile model semantyczny jest abstraktem, niezależnym od konkretnego modelu logicznego, o tyle model logiczny jest już konkretny – wybór determinuje kolejne kroki. Ponieważ wybrany został model relacyjny, konieczne jest stworzenie i opisanie konkretnych relacji pomiędzy encjami. Ze względu na wybór SZBD MySQL określone będą sposoby realizacji określonych relacji, np. relacja typu n:m (wiele do wielu) jest realizowana przez dodanie do bazy dodatkowej tabeli realizującej fizycznie tą relację (Rysunek 4).

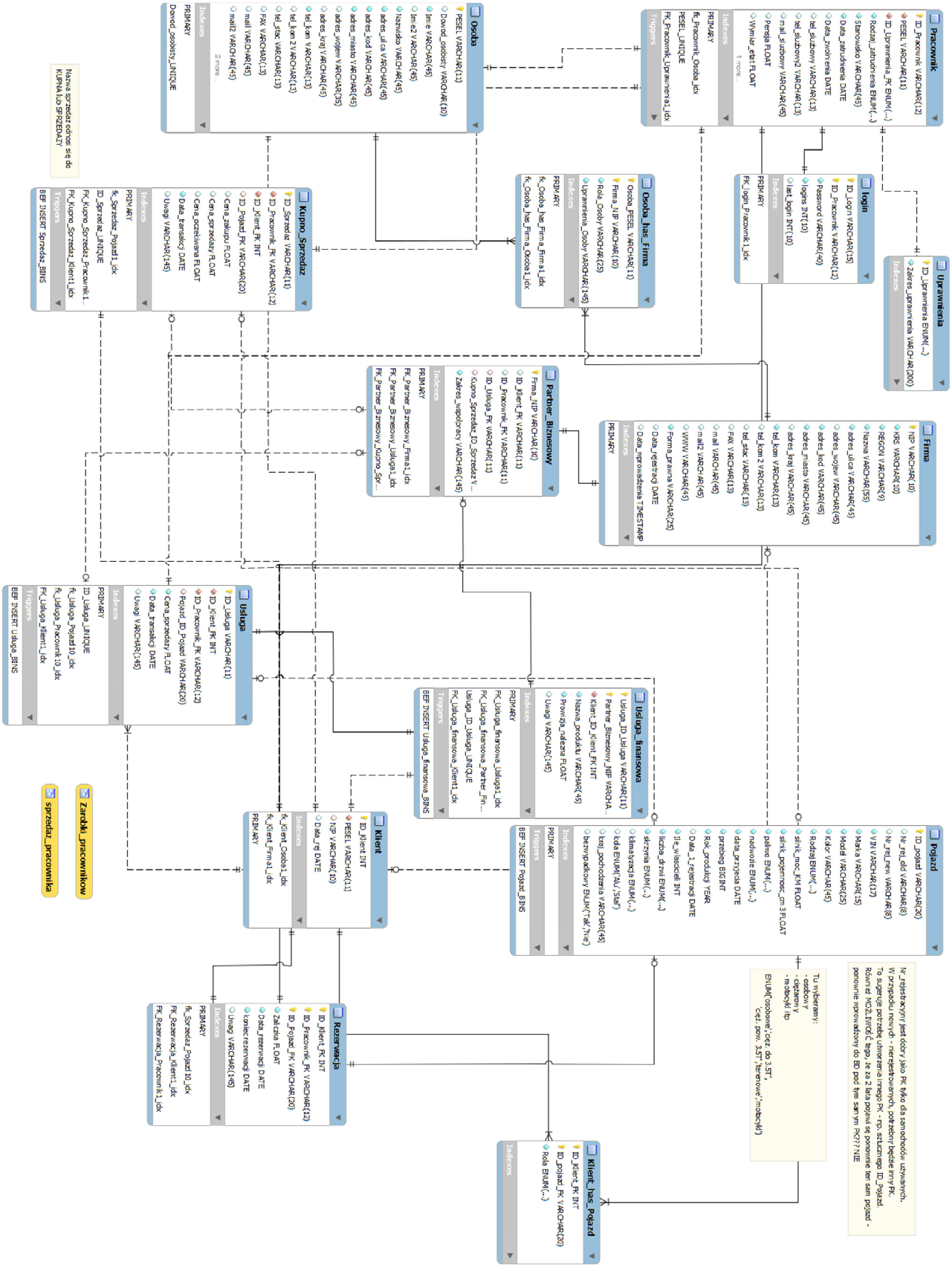
Rysunek 4 Relacja n:m

[1] dział semiotyki zajmujący się badaniem związków, jakie zachodzą między wyrażeniami języka a przedmiotami, do których się one odnoszą - [8]

[2] dział semiotyki badający wzajemne stosunki i właściwości budowy wyrażeń języka w procesie porozumiewania się ludzi [8]



Rysunek 5 Diagram związków encji



Rysunek 6 Diagram modelu logicznego

2.2.4 Implementacja modelu w ramach SZBD

Poprawnie zaprojektowana baza powinna odpowiadać modelowi, który przyjęliśmy (patrz 2.1.2), zapewniać brak redundancji, spójność danych. SZDB powinien przy tym dawać możliwość efektywnego, bezpiecznego przetwarzania danych. Dane powinny być odpowiednio niezależne i trwałe. Dobrze jest, jeśli zaprojektowana aplikacja, wraz z SZBD daje możliwość (jeśli jest taka potrzeba) zapewnienia współbieżnego dostępu z odpowiednio autoryzowanym dostępem do danych.

Do implementacji bazy powinno się korzystać z oprogramowania, które daje wszystkie założone w projekcie możliwości, lub pozwala, adekwatnym do potrzeby kosztem, ewentualny brak zastąpić bądź wyeliminować.

2.3 Etapy projektowania aplikacji

Aplikacja jest bytem całościowym, będącym syntezą niezależnych, lub częściowo niezależnych modułów. Pierwszym etapem powinno być przemyślane dopasowanie narzędzi i technologii do opisanego projektu. Jeśli mamy już model koncepcyjny, znamy opis przypadków użycia, wymagania funkcjonalne i niefunkcjonalne, tworzymy opis algorytmów, które będą działać w aplikacji. Następnie w celach dokumentacyjnych te procedury, algorytmy i funkcje systemu staramy się możliwie dokładnie opisać.

W niniejszej pracy takie procedury zostały opisane w notacji UML w formie diagramów przypadków klas, diagramów przypadków użycia, diagramu związku encji. Chodzi o możliwie najdokładniejsze zwizualizowanie i opisanie działania systemu, zanim jeszcze zaczniemy implementację wynikowego kodu.

Taki opis pozwala lepiej określić zakres technologii, które powinny być użyte. Pozwala podzielić pracę na etapy, moduły – czyli zaplanować kolejne prace. Wreszcie, co może okazać się niezbędne – ze względu na obszerność materiału, pozwala podzielić konkretne prace na zespół ludzi tworzących dany projekt. W niniejszej pracy zespół jest jednoosobowy.

3. Projekt bazy danych

Projekt prezentowanej bazy danych został stworzony po gruntownej analizie potrzeb konkretnego przedsiębiorstwa. Zawiera w sobie bardzo określony wycinek rzeczywistości i pozwala go zapisać oraz poddać analizie. Projekt bazy danych jest stworzony w środowisku MySQL Workbench 5.2 CE, o którym szerzej w następnym punkcie.

Baza danych składa się z 14 tabel obrazujących faktyczne byty w istniejącym przedsiębiorstwie. Pozwalają one na przechowywanie informacji dotyczących bytów fizycznych oraz abstrakcyjnych: pracowników, klientów, partnerów, samochodów oraz transakcji czy usług. Baza pozwala także zobrazować wzajemne relacje między tymi bytami – encjami, jak na przykład:

- osoba – pracownik, która jest właścicielem firmy,
- osoba – klient, nie będący właścicielem firmy,
- osoba – przedstawiciel firmy będącej klientem,
- osoba – współwłaściciel spółki będącej partnerem biznesowym,

Ponieważ ważnym wymaganiem było stworzenie bazy jak najbardziej elastycznej – istnieje bardzo duża liczba możliwości układania wzajemnych relacji między encjami – co daje ogromne bogactwo możliwych do przedstawienia bytów.

Osobno zostały w bazie zaimplementowane dwie perspektywy (ang.View). Reprezentują one wybraną zawartość tabel [1], dając dostęp do danych przekrojowych, które są zapisane w bazie danych. Można do nich sięgać efektywnie z poziomu aplikacji, jak do realnej tabeli np. wykonując zapytanie typu SELECT * from `nazwa_perspektywy`.

W bazie utworzono także 5 wyzwalaczy (ang. trigger) czyli funkcji, wykonujących się w trakcie interakcji z bazą. Ich zadaniem jest budowanie odpowiednich wartości w samej bazie – nie angażując do tego warstwy aplikacji.

3.1 Środowisko pracy

Bardzo ważnym zagadnieniem jest środowisko pracy i technologie użyte do budowania aplikacji i bazy danych. W zależności od użytych programów, pierwotne założenia i wymagania będą odzwierciedlone w różny sposób.

Baza danych jest bazą MySQL, wykonana została za pomocą MySQL Workbench 5.2 CE na licencji freeware. MySQL jak podaje [2] jest SZDB (System Zarządzania Bazą Danych), stworzonym przez szwedzką firmę MySQL. W 2008r kupiony został przez SunMicrosystem, a obecnie jest własnością firmy Oracle. Jest to nowoczesny, wydajny system. Jednak jego największą zaletą jest prostota, bogata dokumentacja webowa i… cena. Jest dostępna na licencji freeware.

Popularnym programem deweloperskim (designerem) służącym do tworzenia baz danych Oracle jest SQL Developer Data Modeler, jednak w tej pracy został użyty głównie w początkowej fazie, przy modelowaniu bazy danych. W niniejszej pracy została użyta wersja 3.1.4.710 pobrana ze strony www.oracle.com na licencji freeware.

Wszystkie diagramy UML zostały stworzone za pomocą demonstracyjnej, 30-o dniowej wersji Enterprise Architect firmy SparxSystem, pobranej ze strony http://www.sparxsystems.com

Całość aplikacji została napisana w języku PHP, z wykorzystaniem javascript i JQuery w wolnodostępnym programie Notepad ++.

Testowanie i prace nad aplikacją były prowadzone przy użyciu darmowego pakietu XAMPP w wersji 3.1.0.3.1.0, zawierającego serwer Apache, interpreter PHP oraz bazę danych MySQL oraz w dwóch popularnych przeglądarkach Mozilla Firefox oraz Chrome.

3.2 Etapy projektu

Ponieważ tworzona baza danych obsługuje niewielkie przedsiębiorstwo, w którym liczba transakcji na systemie z założenia nie będzie duża, a celem jej utworzenia jest w większości analiza danych w niej przechowywanych – nie jest wskazane normalizowanie bazy. Baza ma być prosta, wydajna i łatwa w obsłudze.

Proces projektowania bazy danych, powinien składać się z przemyślanych etapów, których realizacja powinna być logicznym łańcuchem czynności. Na początku każdego procesu, powinno się zadać pytanie o cel do którego dążymy. Następnym etapem jest dopiero określenie wymagań i ich specyfikacja. Posługując się wykładem dotyczącym inżynierii oprogramowania należy zaznaczyć następujące etapy jego wytworzenia, za [7]:

• Określenie wymagań
- specyfikacja wymagań
- studium wykonalności
- negocjacje biznesowe

<---- decyzja realizacji

Ten etap powinien być zainicjowany przez negocjacje z docelowym klientem – z przyczyn oczywistych zostanie pominięty. Jednocześnie trzeba by podkreślić, że autor w tym momencie jest również poniekąd klientem. Taki proces powinien być zatem nazwany autonegocjacją.

• Wytworzenie systemu
- analiza wymagań
- projektowanie systemu
- opracowanie oprogramowania
- przygotowanie infrastruktury
- integracja i testowanie

<---- odbiór

Ten etap jest opisany szerzej w punkcie 2.2 i 3.3, etap testowania – będący integralną częścią tworzenia aplikacji to punkt 4.1 niniejszej pracy

• Wdrożenie
  - przeniesienie danych
  - szkolenie użytkowników
  - zmiana procesu biznesowego
<---- odbiór ostateczny

Ten etap, będzie częściowo opisany w puncie 4.2, jednak ze względu na brak możliwości – ze względów formalnych – faktycznego wdrożenia w realnym przedsiębiorstwie, będzie opisany teoretycznie.

• Eksploatacja i konserwacja
  - używanie systemu
  - usuwanie pozostałych błędów
  - modyfikacja i ewolucja

Ten etap nie będzie opisany w niniejszej pracy – zgodnie z tym co zostało opisane wyżej – nie ma zastosowania faktycznego.

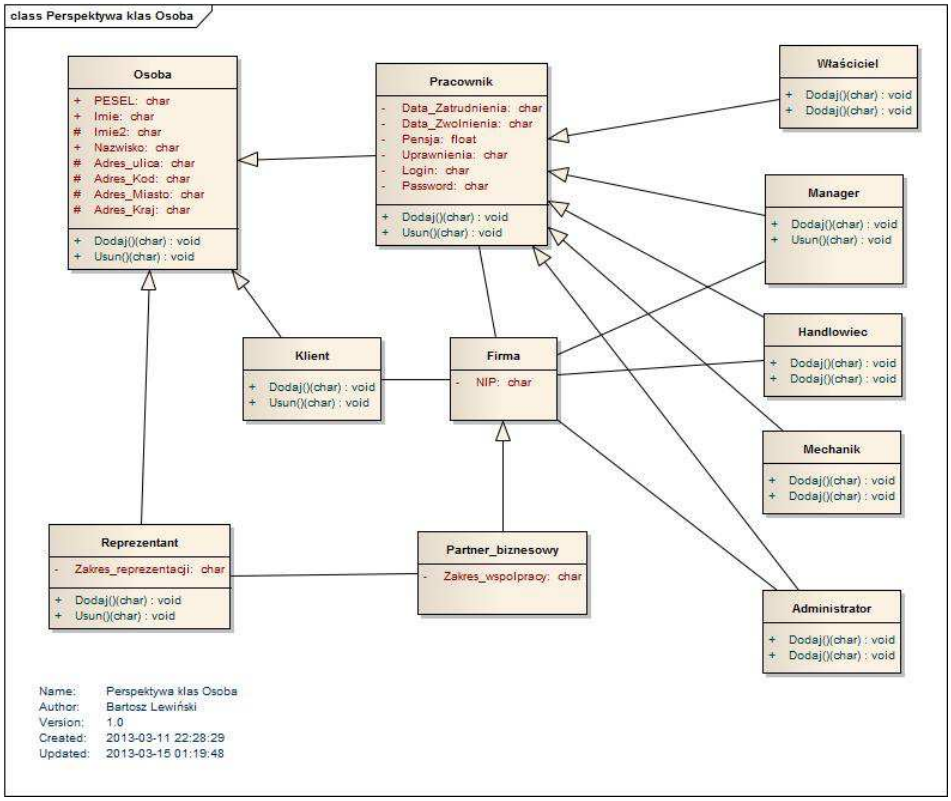
3.2.1 Modelowanie pojęciowe

Modelowanie pojęciowe bazy danych zostało zrealizowane poprzez wytworzenie diagramów obrazujących poszczególne encje (byty), występujące w bazie danych. Rysunek 7 Perspektywa klas - Osoba.

Przykładowo istnieje encja „pracownik” (encja słaba), jest ona szczególnym przypadkiem bytu „osoba”. Czyli „osoba” jest generalizacją „pracownika”. Byt „osoba” ma pewnego rodzaju określające go własności – PESEL, wiek, płeć, adres itd., są to własności, które ma również „pracownik” – są one zatem przez tą encję dziedziczone. Encja „pracownik” może mieć jednak również swoje własne własności (pensja, stanowisko), które nie są znane „osobie”.

Kolejnym krokiem jest określenie liczebności poszczególnych encji, relacji pomiędzy nimi - Rysunek 5.

Ze względu na sposób wytworzenia bazy danych, jak i jej rozmiar nie będziemy się skupiać na wytworzeniu pełnego modelu. Narzędzie MySQL Workbench pozwala na szybkie tworzenie modelu logicznego z pominięciem modelowania pojęciowego, dlatego ten element jest w niniejszej pracy jedynie zaznaczony. W większych bazach danych, o znacznie większej ilości encji byłoby to jednak konieczne.



Rysunek 7 Perspektywa klas - Osoba

3.2.2 Modelowanie logiczne

Jest zrealizowane poprzez specyfikację wymagań i modelu pojęciowego. Ponieważ jest powiązany z modelem bazy danych, musi określać struktury modelu danych, nie zaś struktury fizyczne. W niniejszym przypadku taki model jest jednocześnie fragmentem dokumentacji całej bazy danych.

Poniżej zamieszczam fragment dokumentacji technicznej omawianej bazy danych. Jak widać określone tutaj są poszczególne tabele, jak również ich (główne) atrybuty. Specjalnie zaznaczone zostały klucze główne (PK – Primary Key) oraz klucze obce (FK – Foreign Key). Atrybuty mają zaznaczone również typy danych, ich obowiązkowość (może być NULL – tu brak oznaczenia, lub NOT NULL – tu NN). Baza danych zawiera również takie elementy jak wyzwalacze i perspektywy, wymienione na końcu specyfikacji. SZBD automatycznie, na podstawie konkretnej relacji występującej między tabelami, generuje indeksy. Ze względu na rozmiar bazy i brak potrzeby tworzenia dodatkowych struktur na tym poziomie, nie zostały utworzone takie struktury, jak: partycje, sekwencje i funkcje.

Z założenia autor pracy jest jedynym administratorem, dlatego na poziomie bazy danych brak tego typu użytkowników. Baza danych umieszczona na serwerze zewnętrznym, powinna być zabezpieczona również hasłem. Tutaj baza jest umieszczona na serwerze Apache (pakiet XAMPP) na „localhost”. Rolę zabezpieczająca jest spełniona na poziomie aplikacji poprzez system logowania i tabele: logowanie i uprawnienia.

tabele (13szt):

Osoba

PK `PESEL` VARCHAR(11) NN ,
`Dowod_osobisty` VARCHAR(10) Unique ,
`Imie` VARCHAR(45) NN ,
`Imie2` VARCHAR(45),
`Nazwisko` VARCHAR(45) NN,
`adres_ulica` VARCHAR(45) NN,
`adres_kod` VARCHAR(45) NN,
`adres_miasto` VARCHAR(45) NN,
`adres_wojew` VARCHAR(35) default 'śląskie' ,
`adres_kraj` VARCHAR(45) default 'Polska' ,
`tel_kom` VARCHAR(13) NN,
`tel_kom2` VARCHAR(13),
`tel_stac` VARCHAR(13),
`FAX` VARCHAR(13),
`mail` VARCHAR(45),
`mail2` VARCHAR(45) default 'unset' ,
`Plec` ENUM('M','K') NN,
`Data_urodzenia` DATE,

Pracownik:

PK `ID_Pracownik` VARCHAR(12) NN,
FK `PESEL` VARCHAR(11) NN Uniqe *REF*.: `Osoba` (`PESEL`),
FK `ID_Uprawnienia_FK`
ENUM('admin','wlasciciel','manager','handlowiec','mechanik','brak')NN
default 'brak' *REF*.: `Uprawnienia` (`ID_Uprawnienia`),
`Rodzaj_zatrudnienia` ENUM('etat','kontrakt','um-zlec') NN default 'etat',
`Stanowisko` VARCHAR(45) NN ,
`Data_zatrudnienia` DATE NN
`Data_zwolnienia` DATE,
`tel_służbowy` VARCHAR(13),
`tel_służbowy2` VARCHAR(13),
`mail_służbowy` VARCHAR(45),
`Pensja` FLOAT default 0 ,
`Wymiar_etat` FLOAT default 1.0 ,
`stawka_kontraktu` FLOAT default 0 ,

Login:

PK `ID_Login` VARCHAR(15) NN,
PK FK `ID_Pracownik` VARCHAR(12) NN *REF*.:`Pracownik`(`ID_Pracownik`),
`Password` VARCHAR(40) NN ,
`logins` INT(10) NN default 0 ,
`last_login` INT(10) ,

Uprawnienia:

PK `ID_Uprawnienia`
ENUM ('admin','wlasciciel','manager','handlowiec','mechanik','brak') NN
default 'brak',
`Zakres_uprawnienia` VARCHAR(200) NN ,
Tabela jest już wypełniona!!!!!!!!!!!!!!!! Bez edycji!!!!!!

Firma:

PK `NIP` VARCHAR(10) NN,
`KRS` VARCHAR(10) NN default '',
`REGON` VARCHAR(9) NN ,
`Nazwa` VARCHAR(55) NN ,
`adres_ulica` VARCHAR(45) NN ,
`adres_wojew` VARCHAR(45) default 'śląskie' ,
`adres_kod` VARCHAR(45) NN ,
`adres_miasto` VARCHAR(45) NN ,
`adres_kraj` VARCHAR(45) default 'Polska' ,
`tel_kom` VARCHAR(13) NN ,
`tel_kom2` VARCHAR(13) default '',
`tel_stac` VARCHAR(13), `FAX` VARCHAR(13),
`mail` VARCHAR(45),
`mail2` VARCHAR(45),
`WWW` VARCHAR(45),
`Forma_prawna` VARCHAR(25) NN,
`Data_rejestracji` DATE,
`Data_wprowadzenia` TIMESTAMP default now() ,

Osoba_has_firma:

PK FK `Osoba_PESEL` VARCHAR(11) NN *REF*.: Osoba` (`PESEL`),
PK FK `Firma_NIP` VARCHAR(10) NN *REF*.: `Firma` (`NIP`),
`Rola_Osoby` VARCHAR(25) NN default 'wlasciciel',
`Uprawnienia_Osoby` VARCHAR(145) NN default 'Pelna reprezentacja'

Pojazd:

PK `ID_pojazd` VARCHAR(20) NN ,
`Nr_rej_old` VARCHAR(8) ,
`Nr_rej_new` VARCHAR(8),
`VIN` VARCHAR(17) NN,
`Marka` VARCHAR(15) NN ,
`Model` VARCHAR(25) NN,
`Kolor` VARCHAR(45) NN,
`Rodzaj` ENUM('osobowe','cięz. do 3.5T','cięż. pow. 3.5T','terenowe','motocykl') NN default 'osobowe',
`silnik_moc_KM` FLOAT NN,
`silnik_pojemnosc_cm3` FLOAT NN ,
`paliwo` ENUM('benzyna','diesel','benzyna+LPG','elektryczny','hybryda') NN default 'benzyna' ,
`nadwozie` ENUM('sedan','hatchback','kombi','SUV','VAN','kabriolet','terenowy','pickup', 'specjalne','inne') NN,
`data_przyjecia` DATE NN,
`przebieg` BIGINT NN,
`Rok_produkcji` YEAR NN,
`Data_1_rejestracji` DATE,
`Ile_wlascicieli` INT NN,
`liczba_drzwi` ENUM('brak','3/4','4/5','inne') NN,
`skrzynia` ENUM('manualna','automat','**sekwencyjna**') NN default 'manualna' ,
`klimatyzacja` ENUM('brak','manualna','automatyczna'),
`kola` ENUM('Alu','Stal') default 'Stal' ,
`kraj_pochodzenia` VARCHAR(45) default 'Polska' ,
`bezwypadkowy` ENUM('Tak','Nie') default 'Tak' ,

Klient:
PK `ID_Klient` INT NN AI,
FK `PESEL` VARCHAR(11) NN *REF*.: `Osoba` (`PESEL`),
FK `NIP` VARCHAR(10) *REF*.: `Firma` (`NIP`),
`Data_rej` DATE,

Partner_biznesowy:
PK FK `Firma_NIP` VARCHAR(10) NN REF.: `Firma` (`NIP`),
`ID_Klient_FK` VARCHAR(11),
FK `ID_Usluga_FK` VARCHAR(11) REF.: `Usluga` (`ID_Usluga`),
FK `Kupno_Sprzedaz_ID_Sprzedaz` VARCHAR(11) REF.: `Kupno_Sprzedaz` (`ID_Sprzedaz`),
FK `ID_Pracownik_FK` VARCHAR(11), `Zakres_wspolpracy` VARCHAR(145) NN ,

Kupno_Sprzedaz:
PK `ID_Sprzedaz` VARCHAR(11) NN,
FK `ID_Pracownik_FK` VARCHAR(12)NN REF.:`Pracownik`(`ID_Pracownik`),
FK `ID_Klient_FK` INT NN *REF*.: `Klient` (`ID_Klient`),
FK `ID_Pojazd_FK` VARCHAR(20) *REF*.: `Pojazd` (`ID_pojazd`),
`Cena_zakupu` FLOAT default 0,
`Cena_sprzedazy` FLOAT default 0,
`Cena_oczekiwana` FLOAT default 0,
`Data_transakcji` DATE NN,
`Uwagi` VARCHAR(145) ,

Usluga:
PK `ID_Usluga` VARCHAR(11),
FK `ID_Klient_FK` INT NN *REF*.: `Klient` (`ID_Klient`),
FK `ID_Pracownik_FK` VARCHAR(12)NN *REF*.:`Pracownik`(`ID_Pracownik`),
FK `Pojazd_ID_Pojazd` VARCHAR(20) *REF*.: `Pojazd` (`ID_pojazd`),
`Cena_sprzedazy` FLOAT NN default 0 ,
`Data_transakcji` DATE NN ,
`Uwagi` VARCHAR(145) ,

Usługa_fianasowa
PK FK `Usluga_ID_Usluga` VARCHAR(11) NN *REF*.: `Usluga` (`ID_Usluga`),
FK `Partner_Biznesowy_NIP` VARCHAR(10) *REF*.:`Partner_Biznesowy` (`Firma_NIP`) ,
FK `Klient_ID_Klient_FK` INT NN *REF*.: `Klient` (`ID_Klient`),
`Nazwa_produktu` VARCHAR(45) NN ,
`Prowizja_nalezna` FLOAT NN default 0 ,
`Uwagi` VARCHAR(145) ,

Rezerwacja:
PK FK `ID_Klient_FK` INT NN *REF*.: `Klient` (`ID_Klient`), ,
PK FK `ID_Pracownik_FK` VARCHAR(12) NN *REF*.: `Pracownik` (`ID_Pracownik`),
PK FK `ID_Pojazd_FK` VARCHAR(20) *REF*.:`Pojazd` (`ID_pojazd`),
`Zaliczka` FLOAT NN default 0,
`Data_rezerwacji` DATE NN,
`koniec rezerwacji` DATE NN,
`Uwagi` VARCHAR(145),

Osoba_has_Pojazd
PK FK `ID_Klient_FK` INT NN *REF*.: `Klient` (`ID_Klient`),
PK FK `ID_pojazd_FK` VARCHAR(20) NN *REF*.: `Pojazd` (`ID_pojazd`),
`Rola` ENUM('Sprzedajacy', 'Kupujacy') NN,

Perspektywy (2szt.):

1) Zarobki_pracowników

```
CREATE VIEW `Inzynierska`.`Zarobki_pracownikow` AS
SELECT ID_Pracownik , stanowisko, pensja, stawka_kontraktu AS kontrakt, round
(pensja+stawka_kontraktu,2) AS suma
FROM pracownik pr JOIN osoba os ON (pr.pesel = os.pesel);
```

2) sprzedaz_pracownika

```
CREATE VIEW `Inzynierska`.`sprzedaz_pracownika` AS
SELECT pr.ID_Pracownik , stanowisko,
round(sum(cena_zakupu),2) as S_cena_zakupu,
round(sum(cena_sprzedazy),2) as S_cena_sprzedazy,
round(sum(cena_oczekiwana),2) as S_cena_oczekiwana
from pracownik pr join kupno_sprzedaz tr
ON (pr.ID_Pracownik = tr.ID_Pracownik_FK)
GROUP BY pr.ID_Pracownik;
```

Powyżej przedstawiona jest cała struktura tabel, opisano wszystkie atrybuty, a szczególne pola zawierające klucze główne i obce, struktury typu wyliczeniowego (ENUM). Bardzo ważne jest, żeby odpowiadające sobie atrybuty miały zgodne typy (klucze główne – PK i klucze obce FK). Konieczne jest również przy tworzeniu bazy danych, zwrócenie uwagi na typy i wartości niektórych atrybutów. Jeśli jest atrybut np. PESEL, to musi mieć on 11 znaków VARCHAR(11) – nie mniej, ani więcej, w przeciwnym wypadku może dojść do utraty integralności danych.

O zgodność danych zadbano również na poziomie aplikacji – jest zaprogramowany skryptowy (javascript) system walidacji danych w formularzach.

3.2.3 Projekt fizyczny

Projekt fizyczny, jest tworzony zgodnie ze specyfikacją i określonymi wymaganiami. Oczywiście w praktyce, używając konkretnego oprogramowania dokonuje się rewizji planów lub korekt – drobnych lub zasadniczych – w zależności od ograniczeń technologicznych SZBD lub oprogramowania, w którym tworzy się aplikację. Faktycznie projekt fizyczny jest zbiorem skryptów wytworzonych w SQL definiujących elementy składowe bazy danych, tzn.:

a) tabele (CREATE TABLE),
b) wszelkie ograniczenia integralności (CONSTRAINT)
c) inne obiekty, jak perspektywy, wyzwalacze, indeksy,

Niniejsza baza danych zawiera także zdefiniowane tabele, które są już wypełnione – tabela z uprawnieniami – przypisane są one „na sztywno”, na przykład:

INSERT INTO `Inzynierska`.`Uprawnienia` (`ID_Uprawnienia`, `Zakres_uprawnienia`) VALUES ('wlasciciel', 'Pełen dostęp do zasobów BD, przeglądanie i edycja danych, raporty BD, raporty zdministratora BD');

3.3 Projekt aplikacji

Projekt samej aplikacji był współbieżny z procesem projektowania bazy danych, opisanym w punktach od 2.2 do 3.2.3.

Istotną różnicą jest natomiast to, że o ile sama baza danych jest niewidoczna dla użytkownika, o tyle aplikacja jest interfejsem, za pomocą którego porozumiewa się on z bazą. Dlatego istotne było na tym etapie:

- zaprojektowanie wyglądu pasującego do profilu firmy i użytkownika.
- zabezpieczenie aplikacji przed działaniem szkodliwym – umyślnym lub przypadkowym
- dopasowanie poziomu trudności obsługi do profilu użytkownika.

Głównym założeniem jest także możliwość dostępu, zgodnie z uprawnieniami, do wszelkich potrzebnych funkcjonalności i danych, określonych na podstawie wymagań funkcjonalnych i niefunkcjonalnych użytkownika.

3.4 Proces realizacji

Proces projektowania bazy danych był dobrze zaplanowany, technologia była dostatecznie opisana i przejrzysta. Równolegle z realizacją bazy danych była sporządzana dokumentacja techniczna (UML).

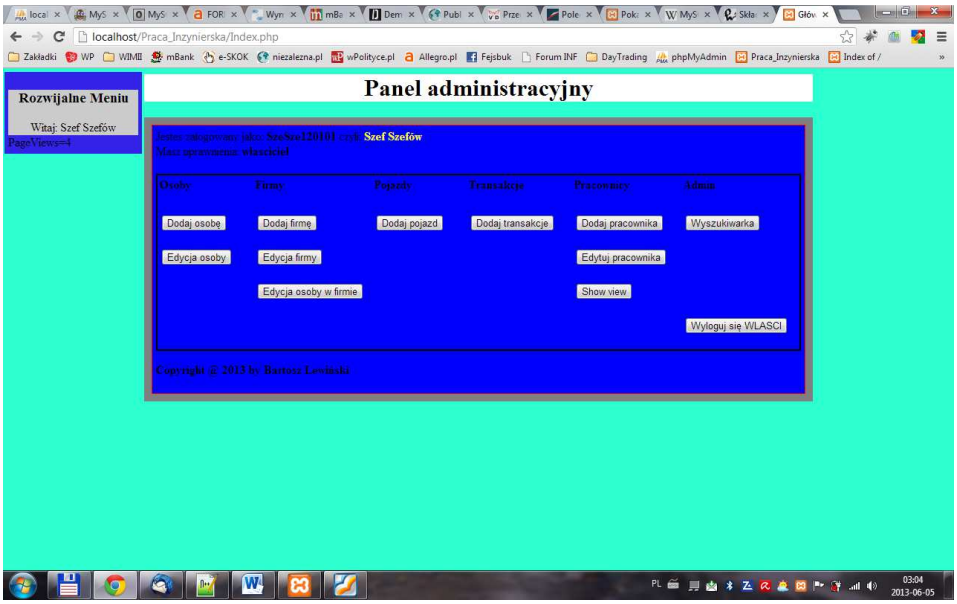
Praktycznie cała baza została wytworzona na podstawie wcześniejszego modelu związków encji (EER) zaprojektowanego w MySQL Workbench. Program ten pozwala szybko zobaczyć efekt pracy, automatycznie generując kod SQL (DDL). Korzystając z wbudowanego w XAMPP pakietu phpMyAdmin, można operować na bazie danych, wstawiać i kasować dane. MySQL Workbench pozwala zaimplementować wszystkie potrzebne elementy bazy danych oraz na bieżąco je testować.

Proces tworzenia aplikacji jest jednak dużo bardziej żmudny i pracochłonny. Relacje w bazie danych są realizowane szybko i łatwo, natomiast podobne związki pomiędzy modułami aplikacji są implementowane w sposób znacznie bardziej skomplikowany.

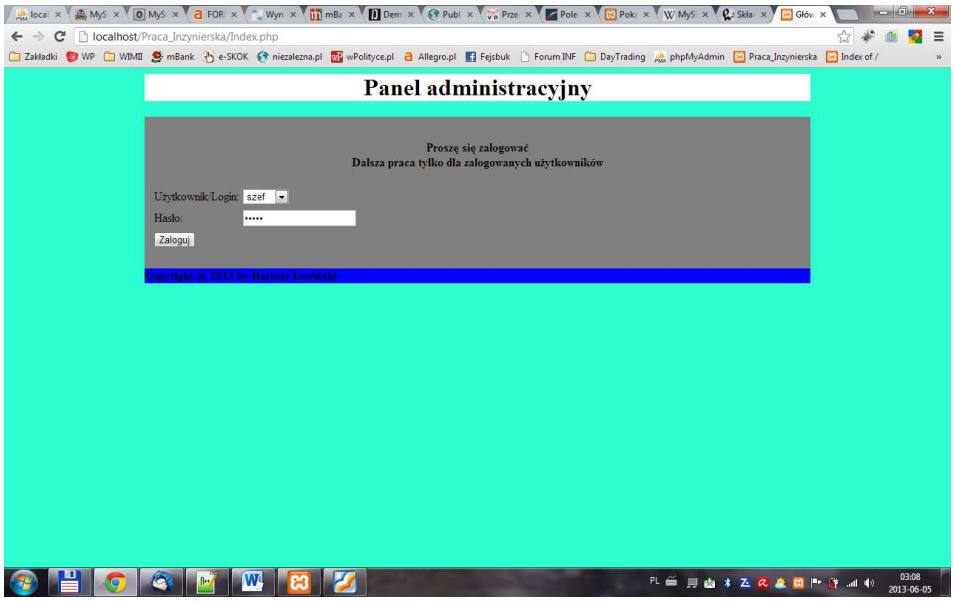
Jest to zrozumiałe: SZBD jest zaprojektowany do konkretnego zadania jakim jest zarządzanie bazą danych. Posiada on wbudowane wszystkie niezbędne narzędzia potrzebne do takiego nadzoru.

PHP natomiast jest językiem wykorzystywanym praktycznie „do wszystkiego”, nie ma wbudowanych mechanizmów logiki – całą logikę aplikacji trzeba zrobić samodzielnie.

Proces programowania aplikacji rozpoczął się od wybrania określonego „layout’u” aplikacji – czyli jej wizerunku, poniżej przedstawiam stroną główną – panel administracyjny przygotowany dla właściciela:



Rysunek 8 Strona główna



Rysunek 9 Panel administracyjny - logowanie

Jak widać panel jest prosty, nie epatuje zbędną szatą graficzną. Przyciski nawigujące do poszczególnych funkcjonalności są pogrupowane w logiczne kategorie. Na panelu znajduje się informacja o tym, kto jest zalogowany i jaki jest jego poziom uprawnień. Do tego okna wchodzimy po zalogowaniu do systemu z Rysunek 9.

Następnym etapem było implementacja jądra aplikacji – logiki która pozwala łączyć się z bazą danych, tworzy system komunikacji wewnątrz aplikacji, odpowiada ze nawigację w jej wnętrzu. Służą do tego dwa pliki **core_v2.0.inc.php** oraz **polaczenie.inc.php**.

Pierwszy plik zawiera pewne funkcje konieczne do działania aplikacji oraz odpowiada za stworzenie ścieżek dostępu do składowych aplikacji, np.

session_start(); -> najważniejsza funkcja w aplikacji – odpowiada za istnienie wszelkich zmiennych, które powstają w danej sesji działania aplikacji – po jej upłynięciu (wylogowaniu – session_destroy()) zmienne są automatycznie usuwane i tracony jest do nich dostęp.

zalogowany_inz(); -> odpowiada za system logowania, tu tworzone są również wszelkie ścieżki do plików aplikacji, na przykład:

$APP_PATH_http = implode (DIR_SEP, array($ROOT_PATH_http, $application)); - tworzy ścieżkę do katalogu “application”.

W drugim pliku łączymy się z bazą danych:

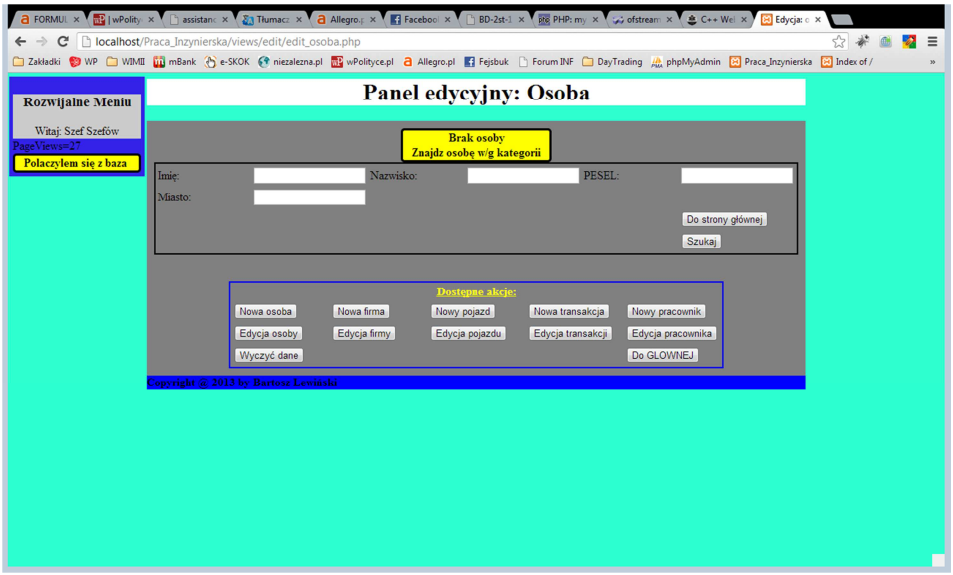
$pol_inz = mysql_connect($host,$user,$pass);

$w_bd = mysql_select_db($bd, $pol_inz);

Po zaimplementowaniu “jądra systemu” zostały utworzone kolejne pliki – podstrony aplikacji. Zgrupowane są one w kilka kategorii:

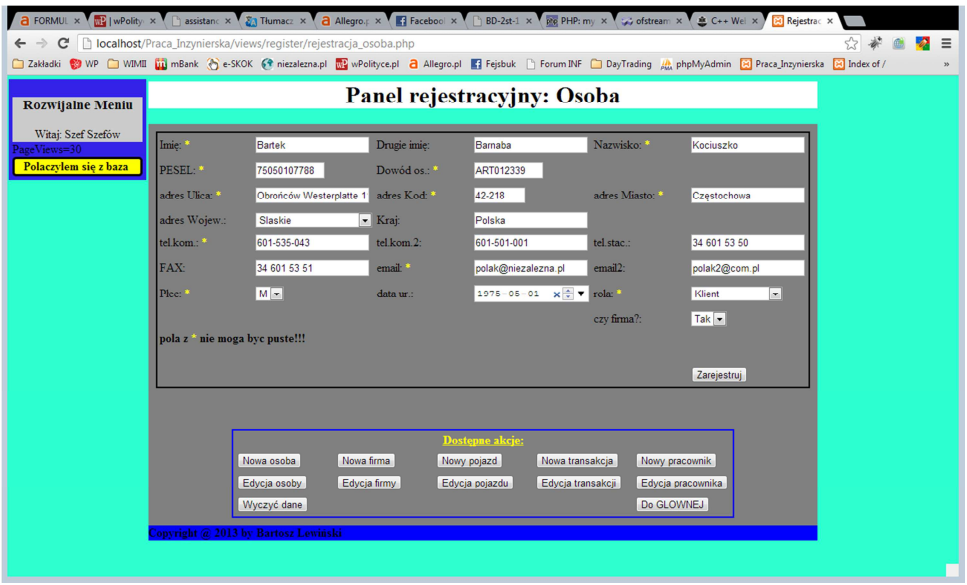
- **modele** – tu znajduje się logika aplikacji – służąca do korespondencji z bazą danych, tworzona jest sieć przesyłanych pomiędzy plikami informacji. Np. po zarejestrowaniu nowego klienta, który jest właścicielem firmy, użytkownik jest przekierowywany automatycznie na podstronę rejestracji firmy.
- **views** (widoki – czyli w miarę czyste pliki HTML, wyświetlane na stronie użytkownika), w katalogu views znajdują się kolejne katalogi z podstronami zgrupowanymi w kategorie:
- **edit** – podstrony zajmujące się edycją danych. Na Rysunek 10 widać formularz, jego pola korespondują z odpowiednim zapytaniem w SQL pobierającym rekordy z bazy danych:

  $SQL_EO_07 = ("SELECT * FROM `osoba` where `pesel` like '".$pes."%' AND `imie` like '".$im1."%' AND `nazwisko` like '".$naz."%' AND `adres_miasto` like '%".$ami."%');



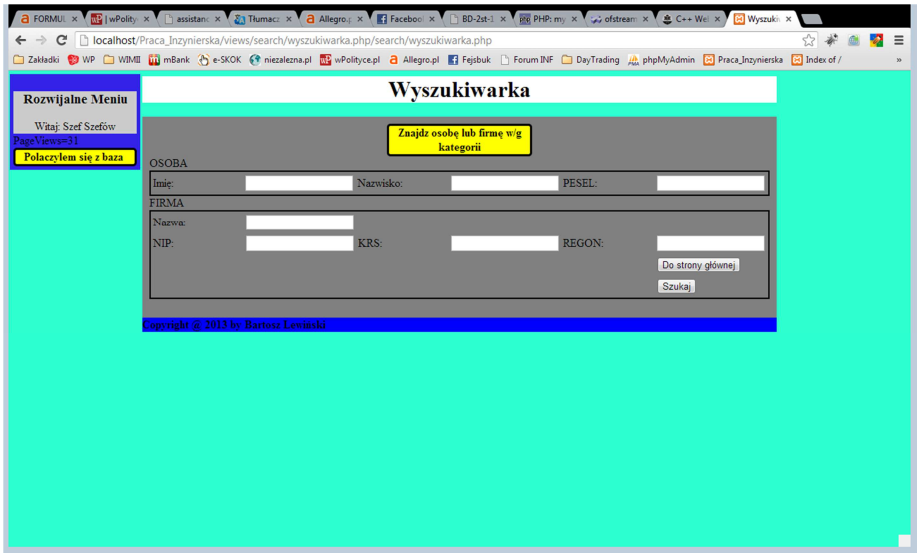
Rysunek 10 Panel edycyjny

- **main** – wyświetlają interfejs użytkownika zgodny z jego poziomem dostępu,
- **menu** - wyświetlają menu użytkownika zgodne z jego poziomem dostępu,
- **register** – wyświetlają podstrony zajmujące się rejestracją nowych danych. Podobnie jak w przypadku edycji formularz jest skorelowany z odpowiednim zapytaniem w SQL.



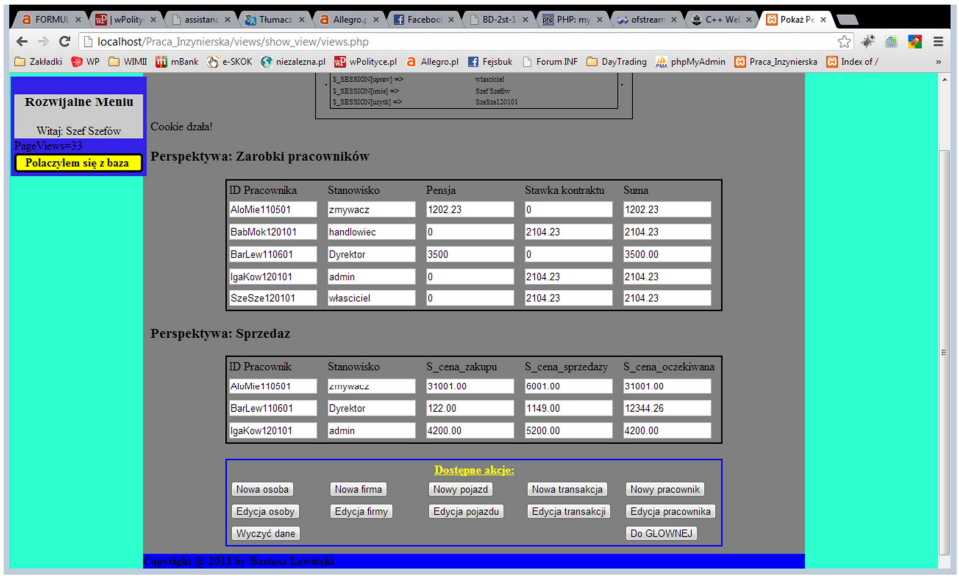
Rysunek 11 Panel rejestracyjny

- **search** - podstrony zajmujące się wyszukiwaniem danych,



Rysunek 12 Wyszukiwarka

- **show_view** – wyświetlanie predefiniowanych perspektyw



Rysunek 13 Show view

Najważniejszym, najtrudniejszym zadaniem jest tworzenie całej logiki przekazywania informacji pomiędzy stronami. Ze względu na założoną elastyczność tworzenia danych oraz konieczność utworzenia logiki, która pozwoli „wyłączyć” myślenie użytkownika, powstała bardzo duża liczba przypadków wymagających rozpatrzenia.

Każdy z nich wymagał przetestowania, często zmian, a nawet przeprogramowania aplikacji. Kilka razy zaszła konieczność niewielkiego przebudowania samej bazy danych.

Należy stwierdzić, że budowa samej aplikacji nie została ukończona. Co prawda najważniejsze jej elementy funkcjonują i jej główne funkcjonalności są uruchomione, ale istnieją elementy, które nie zostały zbudowane w ogóle (np. moduł partnerów biznesowych, rezerwacji). Niektóre moduły są nieukończone bądź nieprzetestowane, albo też są zbudowane w nieodpowiedni sposób. Jest to wynik rozległości i wielowątkowości aplikacji. Wnioski zostaną omówione w podsumowaniu.

4. Testowanie i wdrażanie aplikacji

Testowanie aplikacji należy do najbardziej żmudnych a jednocześnie koniecznych procesów podczas tworzenia aplikacji. [2] oraz [7]. Jest to proces, który ma zapewnić odpowiednia jakość wytwarzanego oprogramowania. Powinien pozwolić oszacować zgodność oprogramowania z postawionymi wymaganiami (weryfikacja) oraz sprawdzić czy spełnia oczekiwania zamawiającego (walidacja).

4.1 Testowanie

Proces testowania trwał od początku powstawania bazy danych i aplikacji. Każdy kolejny krok wiązał się ze sprawdzeniem działania tworzonego elementu oraz działania całego (dotychczas) wytworzonego systemu łącznie.

W procesie testowania bazy danych stworzony został ciąg danych testowych. Weryfikowana była wzajemna zgodność danych oraz testowane relacje i połączenia w bazie danych. Testy te wielokrotnie powodowały zmiany w układzie danych bazy, edycje wzajemnych relacji lub zmianę typów atrybutów.

W późniejszym etapie, kiedy tworzona była aplikacja, proces testowania bazy odbywał się dalej, tym razem danych testowych dostarczała sama aplikacja.

Testowanie aplikacji było prowadzone w trzech kierunkach – zgodności danych pochodzących z bazy danych z reprezentowanymi w aplikacji. Zgodności danych przesyłanych z aplikacji do bazy oraz wewnętrznej spójności danych przekazywanych wewnątrz aplikacji.

Do tego celu wytworzono system testów, przykład jednego z nich na Rysunek 14 dotyczy systemu rejestracji osoby. Każda możliwość została zweryfikowana – od najbardziej nieprawdopodobnej – na przykład, że rejestrujemy osobę, która jest już zarejestrowana w bazie (Osoba_has_firma), ale jednocześnie jest nowo rejestrowaną osobą – do sytuacji normalnej – czyli rejestracji klienta w bazie.

Właściwie trzeba stwierdzić, że testowanie danej podstrony trwało więcej czasu niż jej wytworzenie. Należy również stwierdzić, że pomimo stosowania ciągłych testów, wciąż wykrywano nowe błędy i nieścisłości. Trzeba również dodać, że jest to zjawisko normalne.

Przypadek	=>	Start		rola os		czy firma		Osoba		Firma		sess Oso		Klient wBD		OHF		wynik
		firma	osoba	Klient	Pracownik	Tak	Nie	wBD	Nowa	wBD	Nowa	T	N	T	N	T	N	
1		T		T		T		T		T			N	T		T		
2		T		T		T		T		T			N	T			N	
3		T		T		T		T		T			N		N	T		
4		T		T		T		T		T			N		N		N	
5		T		T		T		T			T		N		N		N	
6		T		T		T			T	T			N		N		N	
7		T		T		T			T		T		N		N		N	
8	wykasowanie sessji firma	T		T			N	T		T			N	T	N	T		OK
9		T			T	T		T		T			N		N	T		
10		T			T	T		T		T			N		N		N	
11		T			T	T		T			T		N		N		N	
12		T			T	T			T	T			N		N		N	
13		T			T	T			T		T		N		N		N	
14	wykasowanie sessji firma	T			T		N	T		T			N	T		T		OK
15	case 1 line 469		T	T		T		T		T		T		T		T		OK
16	case 1 line 462		T	T		T		T		T		T		T			N	OK
17	Case 1 line 438		T	T		T		T		T		T			N	T		OK
18	case 1 line 431		T	T		T		T		T		T			N		N	OK
19	Case 0 line 342		T	T		T		T			T	T			N		N	OK
20	case 1 line 431		T	T		T			T	T		T			N		N	OK
21	case 0 line 342		T	T		T			T		T	T			N		N	OK
22	przekierowanie => rej prac		T	T			N	T		T		T		T		T		OK

Rysunek 14 Testy - rejestracja osoba

4.2 Wdrażanie

Proces wdrażania aplikacji może zostać rozpoczęty w momencie tworzenia aplikacji bądź po jej ostatecznym zakończeniu. Powinien on składać się z wytworzenia dokumentacji szkoleniowej dla finalnego użytkownika, systemu szkolenia użytkowników oraz sposobu stałego reagowania w sytuacji awarii systemu (konserwacja).

Założony został następujący (teoretyczny ze względów formalnych) sposób wdrożenia (po ukończeniu aplikacji) :

1. Stworzenie specyfikacji systemu i jej akceptacja przez użytkownika
2. Instalacja aplikacji, instalacja, konfiguracja i uruchomienie aplikacji w wersji testowej
3. Uruchomienie wersji produkcyjnej
4. Szkolenie administratorów i użytkowników
5. Obsługa powdrożeniowa w zależności od potrzeb użytkownika.

5. Podsumowanie

Udało się zrealizować wydzielony fragment systemu bazodanowego dla przedsiębiorstwa, zgodnie z założeniami. W pełni funkcjonalne są moduły:

- rejestracji osób: klientów i pracowników, rejestracji firm, pojazdów i transakcji,

- edycji osób i firm,

- logowania,

- autoryzacji według poziomu dostępu do zasobów.

Nie udało się zrealizować pozostałych modułów – rejestracji partnerów biznesowych i ich usług, edycji pracowników, pojazdów i transakcji. Ważne, że nie są to moduły trudne do budowy i testowania. Faktycznie są one pod względem logicznym takie same, jak te już istniejące.

Podstawowym powodem niezrealizowania całej aplikacji jest jej obszerność. Biorąc pod uwagę, że tworząc oprogramowanie tego typu korzysta się z umiejętności zespołu osób. Każda specjalizuje się w określonej dziedzinie, przez co jest w niej maksymalnie efektywna. Takie oprogramowanie można wówczas wytworzyć szybciej. Jedna osoba, tworząc taką aplikacje, jest niekiedy zmuszona poświęcić czas na naukę niektórych technologii, nie mając wsparcia zespołu.

Tym nie mniej, widać, że tego typu aplikacje bazodanowe dają ogromne możliwości małym i średnim przedsiębiorstwom. Pozwalają optymalizować, poprzez analizę danych, niektóre parametry ekonomiczne (np. sprzedaż). Istnieje ogromne uzasadnienie dla tworzenia takiego oprogramowania oraz duże są potrzeby rynku. Taka aplikacja potrafi zastąpić pracownika, lub lepiej ukierunkować jego wysiłek. Dając potencjalnym użytkownikom właściwie nieograniczone możliwości analityczne.

Język PHP okazał się potężnym narzędziem, pozwalającym stosunkowo łatwo i szybko zacząć tworzyć skomplikowaną aplikację. MySQL okazał się łatwym do opanowania narzędziem pozwalającym tworzyć bazy danych o niewielkim stopniu komplikacji.

Trzeba jednak zauważyć i wady tych rozwiązań.

PHP jest językiem skryptowym, mało ujednoliconym. Z jednej strony daje możliwość rozwiązanie danego problemu na wiele sposobów, z drugiej często ciężko kontrolować kod, albo zrozumieć kod napisany przez kogoś innego. Często ciężko określić, które rozwiązanie jest najbardziej wydajne. Istniejące frameworki PHP (ZEND, Kohana, Symphony i inne) teoretycznie ułatwiają pracę, z drugiej strony bardzo trudno jest zrozumieć strukturę kodu frameworka, bez dokładnej analizy dokumentacji.

MySQL nadaje się do prostych baz danych. Jest jednak bardzo uboga wobec możliwości innych baz (np. Oracle czy MS SQL). Praktycznie nie istnieje możliwość pogłębionej analizy danych (np. OLAP), bardzo ubogo prezentują się także funkcje związane z backupem danych, mało jest narzędzi administracyjnych.

6. Streszczenie

Niniejsza praca ma na celu pokazanie odbiorcy szerokiej problematyki tworzenia aplikacji bazodanowej dla przedsiębiorstwa. Praca pokazuje teoretyczne podstawy oraz cały proces prowadzący do powstania ostatecznego produktu jakim jest gotowa aplikacja.

Poruszając się od zagadnień teoretycznych, rysu historycznego, różnych systemów bazodanowych, ostatecznie koncentrujemy się na konkretnej reprezentacji – bazie MySQL. Jest ona powszechnie wykorzystywana do tworzenia aplikacji bazodanowych.

W pracy pokazana jest analiza zagadnienia z jakim powinien się zmierzyć programista, kolejne etapy projektowania bazy danych, na której opiera się aplikacja. Finalnie pokazany jest również od podstaw proces implementacji aplikacji w języku PHP. Pokazany jest sposób łącznia aplikacji z bazą danych, sposób w jaki następuje przekazywanie informacji wewnątrz samej aplikacji, komunikowanie się ze sobą poszczególnych modułów.

W ostatniej części naszkicowany został etap, jakim jest testowanie a następnie wdrożenie aplikacji do pracy. Jest to najbardziej żmudny moment tworzenia oprogramowania, jednakże to od niego zależy faktyczny sukces finalnego produktu.

7. Abstract

The purpose of my thesis is to show general problems of creating a database application for business. The thesis shows a theoretical base and specific processes aiming at producing a complete application.

Starting from theory, historical background , and different database systems we finally concentrate on specific representation - MySQL database. MySQL is commonly used to create database applications.

My thesis analyses issues a programmer must face, and successive steps of designing the database on which the application is based. Finally, the process of building the application in PHP language from scratch up to the very end is shown. This presentation also includes the manner of linking the database with the application, and the way in which information are transferred within the application, including the intercommunication between the individual modules.

At the end I have presented an outline of testing and final implementation phase. This was the most arduous moment of software development, but the success of the product depended on it.

LITERATURA

A. Źródła książkowe

[4] J. Price, Oracle Database 11g i SQL. Programowanie, Wydawnictwo HELION, 2009.

[5] Komputer expert, Programowanie krok po kroku. Kurs PHP, Axel Springer, 2011.

B. Wykłady

[1] dr inż. O. Siedlecka-Lamch, „Wykłady z Baz Danych," Częstochowa, 2009/2010.

[7] dr A. Jakubski, „Inżynieria oprogramowania," w Wykład, Częstochowa, 2012.

C. Strony WWW

[2] Wikipedia.org, „Wikipedia," [Online]. Available: http://pl.wikipedia.org/wiki/Strona_glowna. [Data uzyskania dostępu: 01 10 2012].

[3] T. Morzy, „Wazniak - Bazy Danych," [Online]. Available: http://wazniak.mimuw.edu.pl/index.php?title=Bazy_danych. [Data uzyskania dostępu: 0 10 2012].

[6] „Portal zarządzania," 13 05 2013. [Online]. Available: http://biznet.gotdns.org/index.php?title=Wymaganie_niefunkcjonalne. [Data uzyskania dostępu: 13 05 2013].

[8] W. n. P. SA, „http://sjp.pwn.pl/slownik," 05 06 2013. [Online]. Available: http://sjp.pwn.pl/slownik.

Numeracja źródeł – w kolejności pojawienia się w tekście.

ZAŁACZNIKI

1) Opis płyty CD

   a. Folder „Praca dyplomowa” zawiera niniejszą pracę dyplomową w formacie .doc.
   b. Folder „Projekt aplikacji” zawiera omawianą w pracy aplikację.
   c. Folder „Baza danych” zawiera kod źródłowy bazy danych w formacie .sql
   d. Folder „Prezentacja” zawiera prezentację dotyczącą niniejszej pracy

2) Płyta CD

Imię i nazwisko ……..
Nr albumu ……..
Kierunek ……
Wydział …….

Częstochowa, dn. ……..

Politechnika Częstochowska

Szanowny Pan (i) Dziekan

OŚWIADCZENIE

Pod rygorem odpowiedzialności karnej oświadczam, że złożona przez mnie praca dyplomowa

pt. ……..
jest moim samodzielnym opracowaniem.

Jednocześnie oświadczam, że praca w całości lub we fragmentach nie została dotychczas przedłożona w żadnej szkole.

Niezależnie od art.239 Prawo o szkolnictwie wyższym wyrażam/ nie wyrażam* zgodę na nieodpłatne wykorzystanie przez Politechnikę Częstochowską całości lub fragmentów w/w pracy w publikacjach Politechniki Częstochowskiej.

……
podpis

*/ nieodpowiednie skreślić.