Politechnika Śląska
Wydział Matematyki Stosowanej
Kierunek Informatyka
Studia stacjonarne I stopnia

Projekt inżynierski

Projekt i wykonanie aplikacji bazodanowej wspomagającej pracę wybranej sieci hotelowej

Kierujący projektem:
dr inż. Jarosław Karcewicz

Autorzy:
Łukasz Lis
Sebastian Nalepka
Mateusz Ogiermann

Gliwice 2015

To jest
dedykacja

Projekt inżynierski:

Projekt i wykonanie aplikacji bazodanowej wspomagającej pracę wybranej sieci hotelowej

kierujący projektem: dr inż. Jarosław Karcewicz

1. **Łukasz Lis** – (1%)
   7.5 strony na 13.12.2015r.

2. **Sebastian Nalepka** – (3%)
   24 strony na 20.12.2015r.

3. **Mateusz Ogiermann** – (0%)
   9 stron na 13.12.2015r.

Podpisy autorów projektu

1. .....
2. .....
3. .....

Podpis kierującego projektem

.....

Oświadczenie kierującego projektem inżynierskim

Potwierdzam, że niniejszy projekt został przygotowany pod moim kierunkiem i kwalifikuje się do przedstawienia go w postępowaniu o nadanie tytułu zawodowego: inżynier.

Data Podpis kierującego projektem

Oświadczenie autorów

Świadomy/a odpowiedzialności karnej oświadczam, że przedkładany projekt inżynierski na temat:
Projekt i wykonanie aplikacji bazodanowej wspomagającej pracę wybranej sieci hotelowej
został napisany przez autorów samodzielnie.
Jednocześnie oświadczam, że ww. projekt:

- nie narusza praw autorskich w rozumieniu ustawy z dnia 4 lutego 1994 roku o prawie autorskim i prawach pokrewnych (Dz.U. z 2000 r. Nr 80, poz. 904, z późn. zm.) oraz dóbr osobistych chronionych prawem cywilnym, a także nie zawiera danych i informacji, które uzyskałem/am w sposób niedozwolony,
- nie była wcześniej podstawą żadnej innej urzędowej procedury związanej z nadawaniem dyplomów wyższej uczelni lub tytułów zawodowych.
- nie zawiera fragmentów dokumentów kopiowanych z innych źródeł bez wyraźnego zaznaczenia i podania źródła.

Podpisy autorów projektu

1. Łukasz Lis, nr albumu:112233, ...(podpis:).....
2. Sebastian Nalepka, nr albumu:225265, ...(podpis:).....
3. Mateusz Ogiermann, nr albumu:226591, ...(podpis:).....

Gliwice, dnia .....

Spis treści

Wstęp **9**

1. Geneza projektu **11**

2. Analiza przedwdrożeniowa **13**

3. Istniejące rozwiązania konkurencyjne **17**

4. Zastosowane rozwiązania **21**

4.1. Wykorzystana technologia 21

4.1.1. .NET Framework/C# 21

4.1.2. WPF 23

4.1.3. XAML 24

4.1.4. T-SQl 24

4.2. Wykorzystane narzędzia 25

4.2.1. Visual Studio 2013 25

4.2.2. SQL Server Management Studio 28

4.2.3. Trello 29

4.2.4. Git/SourceTree/GitHub 32

4.2.5. ReSharper 36

4.2.6. Jenkins 37

4.3. Biblioteki zewnętrzne 37

4.3.1. Extended WPF Toolkit 37

4.3.2. PDFsharp & MigraDoc Foundation 40

5. Implementacja **41**

5.1. Architektura aplikacji 41

5.1.1. Wzorce architektoniczne oprogramowania 41

5.1.2. Zastosowany wzorzec architektoniczny – MVVM 42

5.1.3. Model aplikacji 43

5.2. Architektura bazy danych 43

5.2.1. Definicja bazy danych 43
5.2.2. MS-SQL – Zastosowany system zarządzania bazą danych 44
5.2.3. Budowa bazy danych użytej w projekcie 45
5.3. Komunikacja bazy danych z projektem programistycznym 47
5.3.1. Mapowanie obiektowo-relacyjne 47
5.3.2. Zastosowane narzędzie ORM – Entity Framework 49
5.3.3. Stosowanie poleceń bazodanowych ze strony projektu C# za pomocą Linq to Entities 49

6. Automatyzacja wydawania kolejnych wersji aplikacji 53
6.1. Continuous Integration/Continuous Delivery/Continuous Deployment 53
6.2. Programy i wtyczki użyte w Jenkins 53
6.2.1. SonarQube 53
6.3. Schemat budowania projektu 56

7. Testowanie aplikacji 57
7.1. Przeprowadzenie operacji logowania użytkownika 57
7.2. Tworzenie nowej propozycji cenowej 60
7.2.1. Opis klienta 60
7.2.2. Sala i jej wyposażenie 63
7.2.3. Usługi gastronomiczne 64
7.2.4. Usługi noclegowe 66
7.2.5. Usługi dodatkowe i forma płatności 66
7.2.6. Zapis nowej propozycji cenowej 68
7.3. Edycja istniejącej propozycji cenowej 69
7.4. Tworzenie pliku PDF istniejącej propozycji cenowej 70
7.5. Modyfikacja kont użytkowników/sprzedawców 71
7.5.1. Dodawanie konta 72
7.5.2. Edycja konta 73
7.5.3. Usuwanie konta 74
7.5.4. Resetowanie haseł 75
7.6. Modyfikacja słowników cenowych 76

8. Specyfikacja wewnętrzna 79
8.1. Przeprowadzenie operacji logowania użytkownika 79

8.1.1. Bezpieczeństwo procesu logowania 79
8.1.2. Wymagania złożoności haseł 80
8.2. Konto administratora i użytkownika – ListView 81
8.3. Tworzenie propozycji cenowej 82
8.4. Edycja słowników oraz kont użytkowników 83
8.4.1. DataGridView a dynamiczna komunikacja z Bazą danych . . . 83
8.5. Tworzenie szablonu PDF propozycji cenowej 85

9. Podsumowanie i perspektywy rozwoju oprogramowania 87

10. Tytuł drugiego rozdziału. Bardzo długi tytuł. Jego formatowanie jest trudniejsze 89

Dodatek: Mój specjalny dodatek 91

Załączniki 93

Bibliografia 95

Wstęp

Obecna sytuacja rynkowa związana z istnieniem dużej ilości firm w zakresie każdej branży wymusza na przedsiębiorcach ciągłe zwiększanie swojej atrakcyjności oraz konkurencyjności w celu zdobycia potencjalnego klienta. W realizacji powyższego zadania konieczne jest wdrażanie nowoczesnych metod umożliwiających między innymi obniżenie kosztów pracy, co wprost przekłada się na niższą cenę produktu końcowego. Niższa cena dla klienta jest jednym z najważniejszych elementów, który determinuje wybór konkretnego usługodawcy. Aspekt ten dotyczy także branży hotelarskiej, która w sposób bezpośredni dotyczy naszej pracy. Liczba hoteli w Polsce od kilkunastu lat nieprzerwanie rośnie [1]. Implikuje to konieczność wzmożonej walki o klienta w celu utrzymania się na rynku. Obniżenie cen usług wynajmu pokoi, sal konferencyjnych oraz cateringów przy zachowaniu odpowiedniego poziomu dochodów jest problemem, z którym zmaga się każda sieć hotelowa. Przedsiębiorcy prześcigają się w znajdowaniu coraz to nowszych i efektywniejszych rozwiązań, które mają na celu rozwiązanie owego problemu. W XXI wieku dużą pomocą w tym zakresie okazuje się informatyzacja.

1. Geneza projektu

Podczas przeprowadzania transakcji dotyczącej organizacji szkolenia w jednej z sieci hotelowych, dostrzeżony został potencjalny problem wynikający z manualnego operowania wszelkimi danymi przez pracowników działu sprzedaży. Pracownicy Ci w czasie kontaktu z klientem mają za zadanie ustalenie wszelkich wartości cenowych usług wybranych przez klienta. Usługi te bezpośrednio związane są z typem wydarzenia, które klient chce zorganizować. Przeważnie są to wszelkiego rodzaju konferencje, kilkudniowe szkolenia firmowe, ale także imprezy weselne, urodzinowe, czy spotkania rodzinne. Przez wzgląd na różną specyfikę powyższych przedsięwzięć sieć hotelowa dysponuje szeroką gamą usług z nimi związanych.

W czasie tworzenia przez sprzedawcę propozycji cenowej organizowanego wydarzenia, klient ma możliwość wyboru odpowiedniej dla jego potrzeb sali konferencyjnej lub bankietowej wraz z jej dodatkowym wyposażeniem, ustalenia aspektów gastronomicznych, takich jak liczba i rodzaj posiłków oraz napojów, a także wyborem odpowiedniego typu pokoju hotelowego. Po wstępnym wyborze zakresu usług, elementy te uzupełniane są o liczbę osób, które będą w wydarzeniu uczestniczyć, a także ilość dni jego trwania. Kolejnym etapem jest negocjacja rabatów oraz finalnie ustalenie formy zapłaty za organizowane przedsięwzięcie. Aktualnie operacja ta wraz z procesem tworzeniem dokumentu propozycji cenowej przeprowadzana jest za pomocą aplikacji Microsoft Excel jak również w wersji papierowej. Obydwie metodologie generują liczne problemy, które bezpośrednio wpływają na zwiększenie kosztów generowanych przez pracowników, a co za tym idzie ceny oferowanych usług. Pierwszy problem odnosi się do aspektu utrudnienia aktualizacji danych, który jest nieodłącznym elementem branży. Ceny pokoi hotelowych ulegają częstym modyfikacjom oraz uzależnione są od sytuacji rynkowej, organizowanych okolicznych wydarzeń rozrywkowych i kulturalnych oraz samego faktu wolnej ich ilości w danym czasie. Analogicznie sytuacja przedstawia się w przypadku produktów gastronomicznych oraz pozostałych wynajmowanych pomieszczeń.

Aktualnie wszelkie modyfikacje cenników pokoi hotelowych, sal możliwych do wynajęcia oraz dostępnych towarów dokonywane są przez menadżera sprzedaży, który za pośrednictwem poczty elektronicznej przekazuje uaktualnione wersje arkuszy

kalkulacyjnych oraz listy cen określonym sprzedawcom. Częste zmiany zobowiązują pracowników działu sprzedaży do ciągłej kontroli skrzynki elektronicznej oraz powodują presję spowodowaną posiadaniem potencjalnie nieaktualnych danych. Zaawansowane arkusze kalkulacyjne wykonane w aplikacji Excel podatne są na błędy ludzkie, wymuszają na pracownikach dobrą znajomość oprogramowania oraz umiejętność jego obsługi, co z kolei niekorzystnie przekłada się na nowych pracowników oraz pracodawcę, który zobowiązany jest przeprowadzać długie i kosztowne szkolenia w tym zakresie. Kolejnym dostrzeżonym problemem jest zagadnienie czasochłonności wyszukiwania danych dotyczących produktów znajdujących się na hotelowej restauracji i ich cen. Pracownik otrzymuje rozbudowaną listę produktów wraz z przyporządkowanymi im cenami i we własnym zakresie zobligowany jest znaleźć interesującą go pozycję w czasie przygotowywania propozycji cenowej.

Dostrzegając powyższe problemy zdecydowano się wykonać dedykowaną aplikację bazodanową, której celem jest ich rozwiązanie. Założono, że konieczne by było, aby każdy ze sprzedawców posiadał możliwość uruchomienia na swoim służbowym komputerze programu, który dzięki połączeniu z bazą danych i wykorzystaniu jej funkcjonalności umożliwiłby przyspieszenie procesu przeprowadzania transakcji z klientem poprzez automatyczną aktualizację cenników oraz przedstawienie listy produktów gastronomicznych w skategoryzowany sposób. Dzięki przechowywaniu wszystkich propozycji cenowych w bazie danych, zanikałaby konieczność samodzielnej ich kategoryzacji i dbania o bezpieczeństwo na lokalnym komputerze. Umożliwiłoby to pracownikom pracę na różnych maszynach, co w znaczny sposób zwiększyłoby ich elastyczność.

Kolejnym atutem przedstawionego rozwiązania byłoby umożliwienie kierownictwu z poziomu kont administratorskich kontroli pracowników poprzez zdalny wgląd w przygotowywane przez nich propozycje cenowe, a co za tym idzie ustalanych z klientem cen oraz zniżek. Menadżer sprzedaży za pomocą swojego uprzywilejowanego konta posiadałby również możliwość aktualizacji odpowiednich tabel danych, dzięki czemu zmiany natychmiastowo byłyby widocznie na kontach sprzedawców. Zmiany te dotyczyłyby możliwości modyfikacji cen oferowanych usług i produktów oraz kontroli składu pracowniczego poprzez funkcjonalność dodawania, usuwania a także modyfikacji ich kont.

2. Analiza przedwdrożeniowa

W rozdziale *Geneza pracy* przedstawiono problemy sieci hotelowej, występujące podczas przeprowadzania transakcji z klientem. Są one podstawowymi czynnikami determinującymi docelową formę aplikacji, dobór narzędzi i technologii oraz wymagania które należy spełnić w celu wdrożenia proponowanego rozwiązania. Pierwszym istotnym elementem infrastruktury IT w firmach jest to, jaki typ systemów operacyjnych jest używany. Jak pokazują rankingi, jednymi z najczęściej używanych systemów operacyjnych są systemy rodziny Windows [2] takie jak Windows 7, 8. W kontekście doboru technologii oznacza to, iż warto zastosować rozwiązania firmy Microsoft które są w pełni zintegrowane z systemami Windows, ze względu na ich wysoką wydajność oraz popularność. Fundamentalnym elementem aplikacji jest baza danych w której przechowywane są detale transakcji, szczegółowe listy oferowanych usług czy dane osobowe pracowników. Firma Microsoft oferuje produkt SQL Server, który posiada darmowe dystrybucje i jest oprogramowaniem bazodanowym dedykowanym dla języków platformy programistycznej .NET Framework. SQL Server w wersji darmowej może zostać użyty do w projektach czysto hobbystycznych jak i w pełni komercyjnych co jest jego dużym atutem. W rankingu portalu db-engines.com SQL Server zajmuje 3 pozycję, nieprzerwanie od 2014 r. pośród ponad 200 sklasyfikowanych silników baz danych [3]. Wystarczającym kryterium które musi zostać spełnione aby firma mogła korzystać z aplikacji bazującej na tym silniku, jest przygotowanie jednego serwera centralnego z którym za pośrednictwem aplikacji łączą się pracownicy. Opisane rozwiązanie eliminuje jeden z problemów, którym jest synchronizacja danych pomiędzy pracownikami, takich jak aktualne wersje propozycji cenowych oraz listy oferowanych usług.

Popularnym językiem platformy programistycznej .NET Framework jest język C#. Umożliwia on tworzenie nowoczesnych aplikacji okienkowych, przy użyciu bibliotek graficznych takich jak WPF. Tworzenie okien i ich komponentów takich jak przyciski, listy rozwijane czy pola tekstowe jest proste i wygodne dla programistów, ponieważ nie wymagają dużych ingerencji w kod źródłowy. Manipulacja komponentami okien odbywa się poprzez modelowanie elementów graficznych w kreatorze. Umożliwia on tworzenie graficznego interfejsu użytkownika podobnie jak w popular-

nych programach służących do obróbki grafiki. Do zaprojektowanego w ten sposób interfejsu, wystarczy podpiąć konkretne funkcjonalności zaimplementowane w kodzie źródłowym. Mogą to być operacje takie jak zapisywanie propozycji cenowych do pliku PDF poprzez kliknięcie przycisku. W konsekwencji czas implementacji aplikacji ulega redukcji. Istotną kwestią podczas implementacji oprogramowania jest komunikacja aplikacji z zasobami bazy danych z poziomu kodu źródłowego. Dla języka C# firma Microsoft zaprojektowała system Entity Framework który służy do odwzorowania relacyjnej bazy danych na obiekty dostępne z poziomu kodu. Zapytania bazodanowe takie jak usuwanie czy modyfikacja tabel są implementowane w kodzie operującym na obiektach utworzonych przez Entity Framework.

Jednym z ostatnich aspektów wytwarzania oprogramowania jest wybranie, jakiego typu będzie to aplikacja. Założeniem niniejszego projektu inżynierskiego jest implementacja aplikacji desktopowej. Zalety tradycyjnych aplikacji desktopowych, które spowodowały wybór tego typu są następujące:

- *Prostota w implementacji GUI*

  Szereg bibliotek graficznych dla języka C# pozwala na szybkie i przyjemne projektowanie graficznego interfejsu użytkownika co przekłada się na szybkość i koszty implementacji projektów.

- *Łatwość obsługi aplikacji desktopowych*

  Aplikacje desktopowe są zazwyczaj prostsze w obsłudze, ze względu na spójny i prosty interfejs tworzony za pomocą kreatorów.

- *Brak problemów zgodności*

  Użytkownicy korzystają z różnych przeglądarek internetowych takich jak Mozilla Firefox, Google Chrome, Safari czy Opera. Dostosowanie wyglądu, skalowalności oraz funkcjonalności dla różnych przeglądarek jest czasochłonne, podczas gdy w przypadku aplikacji desktopowych wymieniony problem nie występuje.

W kontekście wdrożenia, aplikacja powinna zostać zainstalowana na stacjach roboczych pracowników co jest drugim warunkiem koniecznym do wprowadzenia nowej infrastruktury IT dla pracowników działu sprzedaży. Jeżeli w przyszłości aplikacja

byłaby rozwijana, za pomocą narzędzi Continuous Integration kolejne wersje oprogramowania mogą zostać szybko dostarczone sieci hotelowej w postaci paczki zip, co zostało szerzej opisane w rozdziale *Automatyzacja wydawania kolejnych wersji aplikacji*.

3. Istniejące rozwiązania konkurencyjne

Wraz ze wzrostem liczby firm [4] działających w Polsce istnieje konieczność lepszego, szybszego zarządzania. Wiąże się to z rozwojem informatyzacji. Można w ten sposób zautomatyzować wiele procesów w firmie miedzy innymi: zarządzanie magazynem, kontrolę sprzedaży, zapotrzebowanie przedsiębiorstwa na różne produkty. Do tego celu wdrażane są zintegrowane systemy informatyczne, których zadaniem jest połączenie wszystkich elementów przedsiębiorstwa, w jeden scentralizowany system. Wykorzystuje się do tego oprogramowania klasy ERP (ang. enterprise resource planning – ERP)[5]. W skład takiego oprogramowania może wchodzić wiele modułów: finansowość, stany magazynowe, kontakt z klientem, w zależności, co w danym przedsiębiorstwie jest wymagane. Wymogiem takiego systemu jest by wszystkie moduły były niezależne, ponieważ każda firma nie potrzebuje ich wszystkich, ale muszą się umieć z sobą komunikować by móc w lepszy sposób zarządzać. W wypadku kontaktu z klientem wprowadza się systemy, które sprawnie obsługują sprawy związane z reklamacjami, wsparciem technicznym i innymi rzeczami związanymi z jego obsługą. Z tym związane jest wystawianie propozycji cenowej. Jest to proces czasochłonny ze względu na konieczność aktualizowania cenników, możliwego wyposażenia sal i innych spraw związanych z organizowanym wydarzeniem. Moduł CRM(ang. customer relationship management – CRM), który wspomaga proces kontaktu z klientem zawiera komponent wspomaganie procesu wystawiania propozycji. Komponent ten nie tylko automatyzuje proces wystawiania propozycji cenowej, ale jest również odpowiedzialny, za sprzedaż oferowanych towarów i usług. W następnej części tego rozdziału przedstawiono różne systemy CRM, których istnieje wersja darmowa.

Do sprawdzenia dostępnych rozwiązań zostały wytypowane trzy programy, które spełniają wymagania sieci hotelowej są to: ProfitCRM[6] i Asystent CRM[7] Gestor GT [8].

- *Interfejs użytkownika*

  Wygląd aplikacji w niektórych przypadkach pozostawiał wiele do życzenia a niektóre zaskoczyły nietypowym interfejsem. W przypadku programu ProfitCRM interfejs użytkownika jest trochę za bardzo pościskany, co utrudniało

znalezienie odpowiednich elementów potrzebnych w danym momencie. Najistotniejszą wadą programu jest bardzo wolno reagujący interfejs użytkownika, co utrudnia pracę. Często zdarza się, że jakieś okienko np. edycja szablonów zniknie i nie ma możliwości robienia czegokolwiek w programie. Bardzo mało intuicyjnie zorganizowany jest zapis oferty do formatu PDF, ponieważ nie robi się tego poprzez przycisk drukuj tylko poprzez podgląd dokumentu i eksportuj do PDF. Kiedy dodawany jest nowy towar lub też usługa otwiera się nowe okno z polami, które trzeba wypełnić. Są to nazwa, kod towaru. Niestety usługi te muszą mieć jednostkę miary. W tym też oknie istnieje możliwość dodawania zdjęć, komentarzy i dodawania ich do grup. Drugim ze sposobów dodawania towaru jest kliknięcie w ołówek, który można zaobserwować na rysunku 1

w procesie tworzenia oferty.

Rysunek 1: Sposób dodawania towaru w ofercie, Źródło: Opracowanie własne.

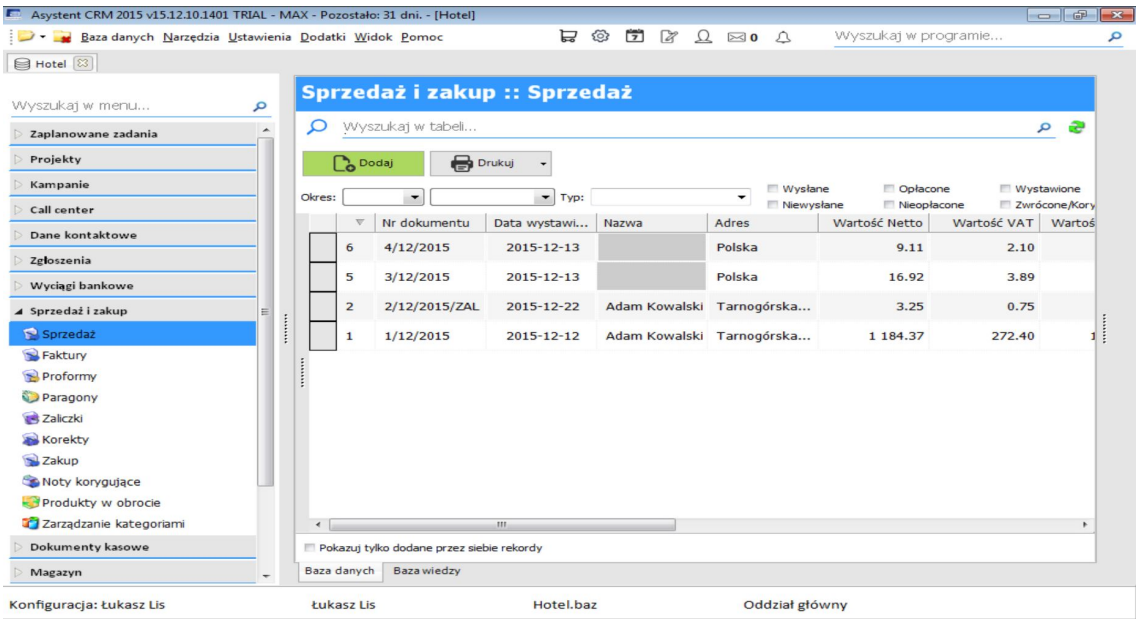
Wymagane wtedy są tyko podstawowe elementy jak nazwa, kod i jednostka miary towaru. Podobny interfejs użytkownika prezentuje Asystent CRM z tą tyko różnicą, że jego przyciski są zdecydowanie większe i jest ich też mniej w jednym oknie. Dzięki przejrzystemu interfejsowi użytkownika przedstawia rysunek 2

jest bardzo prosta nawigacja w programie.

Dodawanie nowych produkty odbywa poprzez magazyn lub też bezpośrednio przy tworzonej ofercie. Program Gestor GT całkowicie zmienił podejście do interfejsu użytkownika przedstawia rysunek 3 z tych trzech jest najbardziej przejrzystym interfejsem, ale niestety mało intuicyjnym na przykład dodawanie nowych produktów jest tutaj bardzo utrudnione i bardziej skomplikowane niż w poprzednich. Wynika to z faktu ukrycia części elementów w głównym interfejsie.

- *Długość wersji darmowej*

  Testowane programy ProfitCRM i Asystent CRM działają w wersji darmowej przez 30 dni, jedynie Gestor GT w wersji darmowej można używać



Rysunek 2: Interfejs programu Asysten CRM, Źródło: Opracowanie własne.

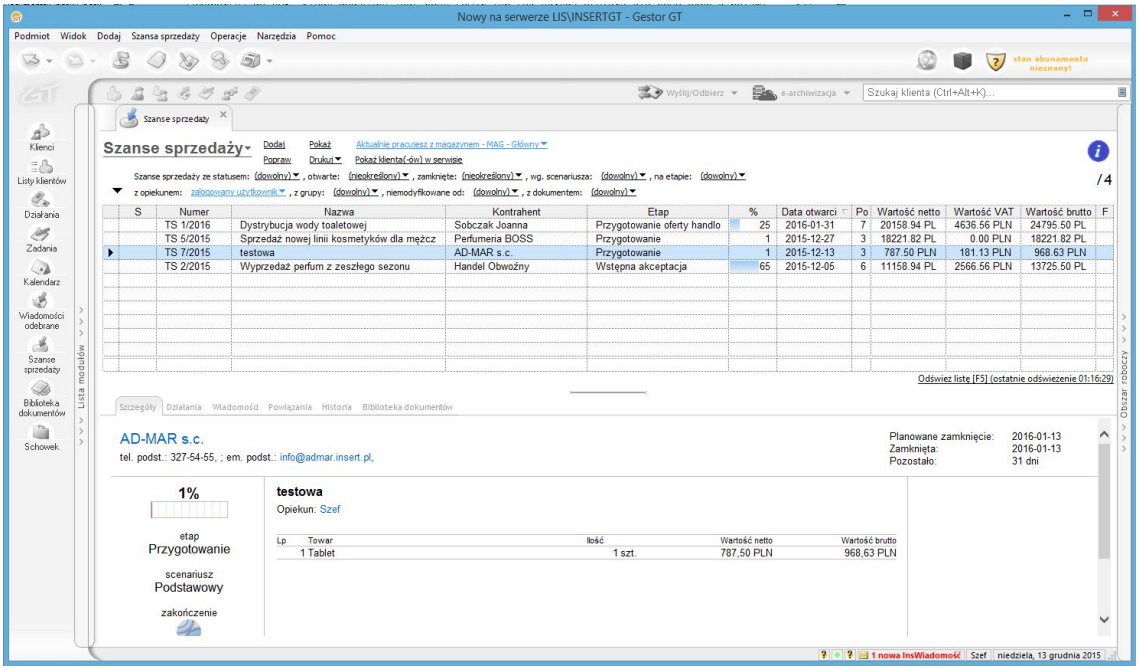
przez 45 dni. W ProfitCRM i Gestor GT dodawany jest Microsoft SQL Server 2008. Asystent CRM jest pozbawiony dodatkowych możliwości.

- *Tworzenie szablonów typu dokumentów*

  W programie ProfitCRM istnieje również możliwość zdefiniowania własnego szablonu. Proces tworzenia nowego szablonu jest procesem skomplikowanym. Najlepszym rozwiązanie jest skopiowanie istniejących w systemie szablonów, ponieważ ich edycja jest zablokowana. Następnie edytowanie skopiowanego. Nie powiodło się również wygenerowanie PDF o wyglądzie zbliżonym do wyglądu przedstawionego przez sieć hotelową. Niestety w programie Asystent CRM edycja wzorów dokumentów do druku jest bardzo trudna w realizacji dla zwykłego sprzedawcy. Jest to spowodowane sposobem edycji. Bez znajomości programowania nie ma możliwości stworzenia nowej wersji dokumentu, co utrudnia zmianę wyglądu pod dana firmę. W ostatnim przetestowanym programie czyli Gestor GT nie ma żadnej możliwości edycji szablonów dokumentów.

- *Cena*

  Koszt licencji programu ProfitCRM na jedno stanowisko pracy wynosi 736



Rysunek 3: Interfejs programu Gestor GT, Źródło: Opracowanie własne.

zł w przypadku programu Asystent CRM, w zależności od wersji od 79-369 zł. Gestor GT kosztuje 859.77zł jest to najdroższy produkt z pośród wybranych

Podsumowując powyższe programy można zauważyć, że proces dodawania produktów jest bardzo zbliżony. Podobnie sprawa wygląda z procesem wystawiania ofert, faktur. Największą możliwością personalizacji wystawianych dokumentów ma program ProfitCRM. Ponieważ żaden z tych programów nie oferował możliwości wystawianie propozycji cenowej, a tak żądanie miała sieć hotelowa dlatego wynikła konieczność stworzenia dedykowanej aplikacji.

4. Zastosowane rozwiązania

4.1. Wykorzystana technologia

Niniejszy rozdział zawiera opis technologii wykorzystanych do implementacji aplikacji. Zakres opisu obejmuje następujące komponenty:

- *Język programowania*
- *Silnik graficzny*
- *Język opisu graficznego interfejsu użytkownika*
- *Język zapytań bazy danych*

4.1.1. .NET Framework/C#

.NET Framework jest platformą programistyczną firmy Microsoft, po raz pierwszy wydany w 2002 r. Służy do wytwarzania oprogramowania dla systemów operacyjnych Windows, Windows Phone, Windows Server i Microsoft Azure [9]. Głównymi komponentami platformy są:

- *Kompilatory języków wysokiego poziomu*

  Umożliwiają kompilację programów napisanych w językach C++/CLI, C#, F#, J#, Visual Basic .NET.

- *CLR (ang. Common Language Runtime)*

  CLR jest tzw. środowiskiem uruchomieniowym języka wspólnego. Jego zadaniem jest kompilacja i uruchamianie tzw. kodu zarządzanego (ang. managed code) zapisanego w standardowym języku pośrednim CIL (ang. Common Intermediate Language), gwarantując funkcje wymagane do działania aplikacji [10].

  BCL jest standardową biblioteką klas, zawiera standardowe funkcjonalności programistyczne takie jak definicje typów danych, odczyt i zapis plików czy obsługa kolekcji zawarte w przestrzeni nazw *System* [11]

- *FCL (ang. Framework Class Library)*

  FCL jest rozszerzeniem BCL, obejmuje rozszerzony zestaw bibliotek m. in. silniki graficzne Windows Forms i WPF [12].

- *CTS (ang. Common Type System)*

  Wspólny system typów CTS definiuje jak używane i deklarowane są typy danych w językach programowania platformy .NET udostępnianych przez CLR [13].

- *CLS (ang. Common Language Specification)*

  Wspólna specyfikacja języka CLS jest zestawem podstawowych zasad które spełniają języki programowania rodziny .NET [14].

Platforma .NET zapewnia wsparcie dla języka C#, który jest obecnie jednym z najpopularniejszych języków programowania na świecie. Według rankingu firmy TIOBE Software z grudnia 2015 r. znajduje się on na piątym miejscu, spośród pięćdziesięciu sklasyfikowanych języków. Począwszy od 2000 r. język C# awansował z dziesiątego miejsca na piąte [15].

C# jest nowoczesnym, obiektowym językiem programowania, wyprodukowanym przez firmę Microsoft. Został wydany po raz pierwszy w lipcu 2002 r. Głównym projektantem języka jest duński inżynier oprogramowania Anders Hejlsberg [16]. Może zostać wykorzystywany do pisania aplikacji desktopowych, internetowych oraz mobilnych. Programy napisane w C# kompilowane są do kodu pośredniego zapisanego w języku CIL (ang. Common Intermediate Language) i wykonywane w środowisku uruchomieniowym .NET Framework. Jest on uznawany za prosty w nauce język, ponieważ posiada szereg modułów oraz udogodnień ułatwiających pracę programistom, takich jak [17]:

- *Odśmiecanie pamięci (ang. Garbage collection)*

  Dynamicznie przydzielona pamięć jest zwalniana automatycznie.

- *Brak konieczności tworzenia plików nagłówkowych .h w porównaniu do języka C++*

- *Możliwość przeciążania operatorów w porównaniu do języka Java*

- *Zmienne inicjalizowane są swoimi domyślnymi wartościami*
- *Wprowadzenie dodatkowych elementów składowych klas, takich jak właściwości i indeksery*

4.1.2. WPF

WPF (ang. Windows Presentation Foundation) [18] jest technologią, którą wprowadzono w .Net 3.0. Wykorzystuje język XML, a dokładnie jego odmianę XAML. Do wyświetlania okien wykorzystywana jest grafika wektorowa, co jest wspomagane przez karty graficzne. Cechuje się on bardzo dużą elastycznością w tworzeniu interfejsu, między innymi tworzenie przycisku z obrazkiem. Technologia ta została zaprojektowana tak, żeby w jak największym stopniu odseparować wygląd aplikacji od innych warstw programu. W niektórych przypadkach wiąże się z dodatkowym nakładem pracy w stosunku do technologii Winforms np. wyświetlany obrazek w kontrolerze ListView . Jeśli chodzi o wygląd aplikacji dzięki dużej elastyczności jest możliwe tworzenie programów z interfejsem 3D. Nie tylko różni się możliwościami tworzenia interfejsów, ale i wewnętrznymi mechanizmami. WPF wykorzystuje dedykowany mechanizm do prezentacji aplikacji, w stosunku do Winformsów, który wywołuje tylko elementy WinAPI. Wiąże się to z brakiem większych możliwości edycji wyglądu aplikacji. Następną cechą istotną z punktu działania programu jest odświeżanie elementów okna. W przypadku Winformsów najechanie na jakikolwiek element z właściwością zmiany wyglądu np. zmiany podświetlania po najechaniu powoduje odświeżenie całego okna programu, co nie ma miejsca w przypadku programów tworzonych w WPF. Natomiast technologia WPF obejmuje dużą liczbę paneli, kontrolek, jest możliwe łączenie też tych elementów, wyzwalacze i dodanie dynamicznych elementów. Możliwe jest tworzenie stylów, szablonów, które ułatwiają proces tworzenia aplikacji. Wykorzystuje się te elementy bardo często. Tych elementów jest pozbawiony Winformsach. Jest też możliwe tworzenie wstążki znanej z innych produktów Microsoftu miedzy innymi Office. Pisząc program z wykorzystaniem WPF jest się powiązanym z hierarchią elementów jakie po sobie występują. Jeśli we wnętrzu jakiegoś elementu znajduje się inny, który ma połączenie z jakimś polem klasy, a nie jest uściślona nazwa klasy, z której to właściwość pochodzi następuje sprawdzenie czy element nadrzędny nie ma podłączonej klasy zawierającym tą właściwość. Drugim istotnym element jest kolejność dodawanych kontrolek. Ma

wpływ na widoczność elementów, to co jest najbardziej na dole w kodzie XAML znajduje się najbliżej użytkownik.

Dzięki bardzo dobremu wsparciu dla wzorca architektonicznego MVVM. Istnieje możliwość podłączania (binding) właściwości z zewnętrznych klas. Ten typ podejścia umożliwia szybkie tworzenie elementów programu, rozwijanie w sposób niezależny interfejsu użytkownik od silnika aplikacji.

4.1.3. XAML

XAML jest to oparty na XML[19] deklaratywny język znaczników, którego zadaniem jest opis interfejsu użytkownika obecnego w zastosowanej przez zespół technologii WPF. W technologii tej język XAML umożliwia zaprojektowanie oraz ułożenie wszystkich elementów wizualnych takich jak kontrolki, ramki oraz okna, a także pozwala na rozdzielenie pracy pomiędzy programistami (back-end) oraz grafikami (front-end), którzy tworzą graficzny interfejs użytkownika. Graficy, przez wzgląd na zakres obowiązków, nie często znają język programowania C#. Problem ten rozwiązuje właśnie XAML, który umożliwia zrozumienie przez nich zasady działania poszczególnych okien, powiązań między nimi oraz projektowanie interfejsu w prosty sposób z poziomu drzewiastej struktury lub dedykowanego narzędzia Expression Blend, które umożliwia przeprowadzenie wszystkich powyższych operacji z poziomu swojego środowiska graficznego.

4.1.4. T-SQl

Jest to krótkie określenie Transaction-SQL[20], które jest rozwinięciem standardowego SQL. Język SQL został stworzony specjalne dla relacyjnych baz danych opracowała go na początku lat 70 firma IBM [21], jest to standard otwarty. Współcześnie jest on wykorzystywany przez większość firm do tworzenia zapytań bazodanowych. Wiele firm dodaje swoje usprawnienia w zależności, co odczuje za stosowne by rozszerzyć możliwość i przyspieszyć pracę baz danych np. PL/SQL firmy Oracle. T-SQL jest językiem transakcyjny, polega na wykonywanie operacji, jeśli operacja została zaakceptowana to następuje zapis do bazy, można to porównać do przelewów bankowych zabranie z jednego konta i przelaniu na drugie konto, jeśli na pierwszym koncie jest suma potrzebna do przelewu. W przeciwnym wypadku przelew nie następuje tak jak zapis do bazy, jeśli dane będą nieprawidłowe. Umożliwia również tworzenie zmiennych, pętli jak również instrukcji warunkowych. Pozwala na

tworzenie obiektów, takie jak widoki, procedury składowane, wyzwalacze i funkcje zdefiniowane przez użytkownika.

4.2. Wykorzystane narzędzia

4.2.1. Visual Studio 2013

Visual Studio jest bardzo rozbudowanym środowiskiem pracy programistycznej, jest produktem typu IDE (ang. Integrated development environment-zintegrowane środowisko programistyczne).Wspiera on również bardzo wiele języków programowania miedzy innymi JavaScript, HTML, C#, VB, XAML, C++. Produkt tego typu zwiera w sobie:

- *Edytor tekstu*

  W Visual Studio edytor tekstu jest bardzo rozbudowany dzięki wielu systemom wspomagających proces wytwarzania oprogramowania. Ważnym z punktu widzenia programisty jest wygląd kodu, by móc się szybko i sprawnie w nim odnaleźć. Jednym z mechanizmów wspomagających wygląd jest kolorowanie składni. Typy zmiennych w zależności, czy jest to klasa kolor, wtedy jest zielony, jeśli inny typ przedefiniowany np. instrukcje warunkowe, pętle, domyślne zmienne niebieski. Drugim mechanizmem jest system podpowiadania elementów składowych klas, nazywanym IntelliSense.Technologia to wspomagana jest również przez odpowiednie komentowanie kodu tak by te komentarze też się wyświetlały w podpowiedzi. Dzięki temu mechanizmowi istnieje możliwość szybszego odnajdywania poszczególnych elementów klasy, który są w danym momencie wymagane. Wykorzystując odpowiednie udogodnienia istnieje możliwość szybszego pisania kodu, co przekłada się na ilość kodu do napisania przez programistę.

- *Kompilator*

  Głównym i najważniejszym elementem tego środowiska jest kompilator, który wytwarza program z linii kodu, a dokładniej tłumaczy kod zapisany w języku programowania na kod zrozumiały dla sprzętu to znaczy maszynowy. W wypadku produktu linii Visual studio może on pracować w dwóch

trybach release i debug, każdy z nich charakteryzuje się innymi parametrami. W trybie debug gdy program jest kompilowany jest możliwe debugowanie z wykorzystaniem wbudowanego debuggera, ale o tym w dalszej części tego rozdziału. Niestety w tym trybie jest brak optymalizacji kodu, jest tworzona cała otoczka umożliwiająca debugowanie. W drugim trybie są uruchamiane optymalizacje miedzy, innymi tak zwane inline, która polega w dużym uproszczeniu na przekopiowane na przykład ciała funkcji w miejsce w którym została ona wywołana. Dodatkowe mechanizmy optymalizacji powodują niemożliwość znajdowania błędów.

- *Debugger*

  Debugger wbudowany w Visual Studio umożliwia szybkie znajdowanie błędów, dzięki możliwości śledzenia programu krok po kroku w trakcie jego pracy. Ten mechanizm umożliwia podglądanie wartości zmiennych, co też dana funkcja zwróciła. Dzięki punktom zatrzymania (ang. breakpoint) jest możliwe wyznaczenie miejsca zatrzymania, a w trakcie debugowania jest możliwe tworzenie nowych punktów w innych elementach programu. Program można śledzić w dwojaki sposób. Dokładnie krok po kroku z wchodzeniem w poszczególne funkcje, klasy nawet te, które zostały wykorzystane z zewnętrznych źródeł lub też wbudowane w środowisko programistyczne. Drugą możliwością jest śledzenie programu w danym bloku kodu, jest to dużym udogodnieniem, bo szukamy tylko w danej funkcji. Istnieje również opcja mieszania tych trybów co umożliwia znajdowanie błędów nie tylko w całym projekcie ale w jego części.

Visual Studio istnieje w wielu wersjach tego środowiska programistycznego. Każda charakteryzuje się innymi elementami.

- *Express Edition*

  Jest to całkowicie darmowe narzędzie do użytku domowa i komercyjnego. Ograniczone jest do konkretnych rozwiązań na przykład aplikacji komputerowych, stron internetowych lub na platformy mobilne.

- *Community Edition*

  Tak jak powyższa wersja programu ta jest również darmowa ograniczona liczbą możliwych użytkowników korzystających z niej do 5, jeśli ktoś chce

korzystać komercyjne to maksymalnie do 250 użytkowników lub przychód nie większy niż milion dolarów amerykańskich.[23]. W stosunku do poprzedniej wersji ma możliwość tworzenie aplikacji dla większej ilości rozwiązań.

- *Professional*

  Do napisania projektu został wykorzystany Visual Studio w wersji 2013 [22] edycja Professional. Jest to produkt komercyjny, który został wykorzystany w ramach licencji studenckiej udostępnionej w programie MSDN Academic Alliance (DreamSpark) dla studentów Wydziału Matematyki Stosowanej [24]. Dzięki temu studenci mogą poznać produkt komercyjny. Wersja ta umożliwia również pisanie testów jednostkowych, tworzenie aplikacji na Windows Phone, aplikacji chmurowych.

- *Premium*

  Zawiera wszystkie elementy poprzednich wersji. Ma bardzo bogate rozbudowane środowisko testowe miedzy, innymi o przykładowe testy, plany testów i testy interfejsu użytkownika.

- *Ultimate*

  Jest to najbardziej rozbudowana wersja oprogramowania, umożliwia tworzenie instalatorów serwisów internetowych. Pozwala również na nagrywanie w trakcie działania aplikacji. Jeśli w trakcie nagrywania zdarzy się błąd nie jest wymagane jego ponowne symulowanie, lub też próba jego odtworzenia. Umożliwia również przedstawiania zależności w kodzie w postaci diagramów UML. Dzięki temu mechanizmowi jest możliwe zauważenie wielu zależności, które często potrafią być bardzo zawiłe. Bardzo ciekawą mechanizmem jest edytowanie kodu w trakcie debugowania co umożliwia na bieżąco naprawianie błędów.

Standardowym środowiskiem dla technologii jakimi są WPF .NET/C# jest Visual Studio. Dzięki wbudowanym mechanizmom istnieje również możliwości ich rozszerzania poprzez napisanie dodatków z wykorzystaniem SDK. Można również pobrać już gotowe rozszerzenia z Visual Studio gallery [25] stworzonych przez innych użytkowników zrzeszonych wśród produktów firmy Microsoft. Wykorzystując odpowiednie udogodnienia istnieje możliwość szybszego pisania kodu, co przekłada się na

ilość kodu do napisania przez programistę. Kolejnym istotnym elementem z punktu widzenia projektu jest możliwość projektowania okien w dwóch technologiach WPF i Winforms. Ułatwia to pracę nad interfejsem użytkownika, ponieważ wyeliminowana jest konieczność ciągłego uruchamiania pisanego programu, by zobaczyć wygląd aplikacji. Zawiera również wiele predefiniowanych elementów dla tworzonego wyglądu miedzy, innymi pola tekstowe, przyciski. Wykorzystując technikę drag-and-drop można w ławy sposób ustawiać elementy w wybranym przez siebie miejscu.

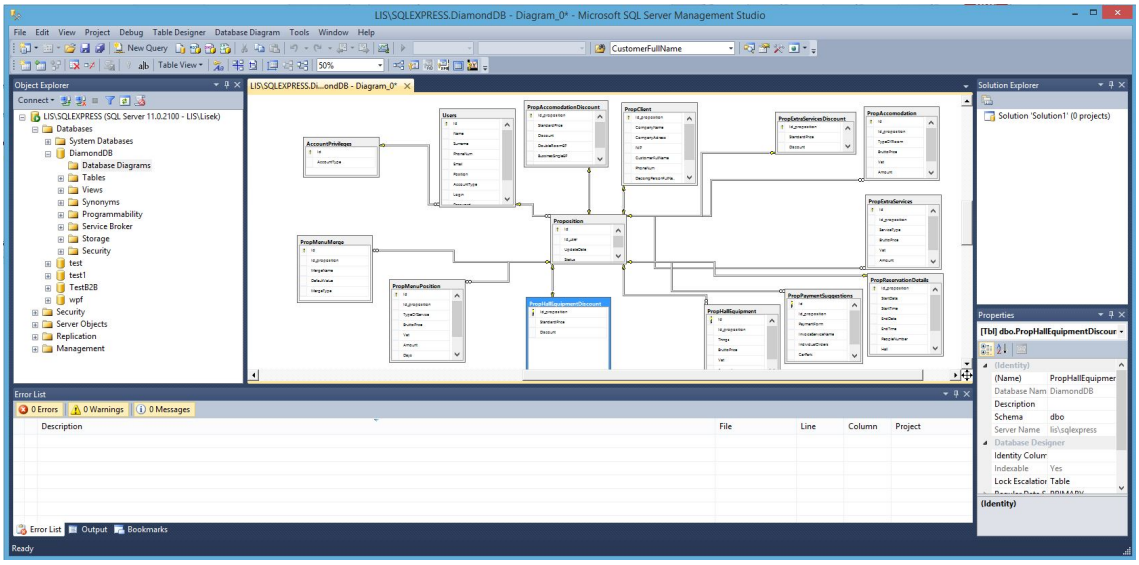
4.2.2. SQL Server Management Studio

W trakcie prac nad projektem wykorzystano darmowe narzędzie do zarządzania bazą danych SQL Server Management Studio 2012 Express [26] wynika to z istniejącej infrastruktury opartej na rozwiązaniach Microsoftu, co ułatwiło pracę i proces projektowania całej bazy danych. Dzięki przejrzystemu interfejsowi użytkownika, można w bardzo łatwy sposób tworzyć nawet bardzo skomplikowane struktury, bez konieczności znajomości języka tworzącego bazę danych. Program ten ułatwił tworzenie wymaganej przez sieć hotelową bazy danych. Istnieje również możliwość generowania skryptów, których zadaniem jest odtworzenie całej struktury bazy danych. W programie jest również możliwość tworzenia diagramów: wszystkich jak również wybranych tabel. Te możliwości przedstawia rysunek 4 Umożliwia to diagnozowanie problemu braku połączeń niektórych tabel z resztą struktury.

Wykorzystując wbudowane mechanizmy możliwe jest łatwe wpisywanie początkowych danych takich jak: użytkownicy systemu, ceny produktów, ich edycja na poziomie bazy. Umożliwia to sprawdzenie, czy baza działa zgodnie oczekiwaniami. W ten sposób można śledzić działanie programu w trakcie pracy. Sprawdzić można czy wszystkie elementy prawidłowo działają i czy w poprawny sposób jest zapisywana dana pozycja. W projekcie wykorzystano:

- *Edytor zapytań*

  Jest to edytor wspomagając języki Transact-SQL, MDX, DMX, lub XML/A. dzięki kolorowaniu słów kluczowych w trakcie pisania skryptów zwiększa to czytelność kodu. Wykorzystuje się tez jako interfejs do zapytań Transact-SQL. Również jak opisany wcześniej Visual Studio wykorzystuje mechanizm IntelliSense odpowiedzialny za podpowiadanie kolumn tabel. Ma możliwość wykrywania błędów nieprawidłowej składni.Wyświetla błędy, ostrzeżenia i ko-



Rysunek 4: Interfejs SQL Server Management Studio Express, Źródło: Opracowanie własne.

munikaty informacyjne, które są zwracane przez serwer.

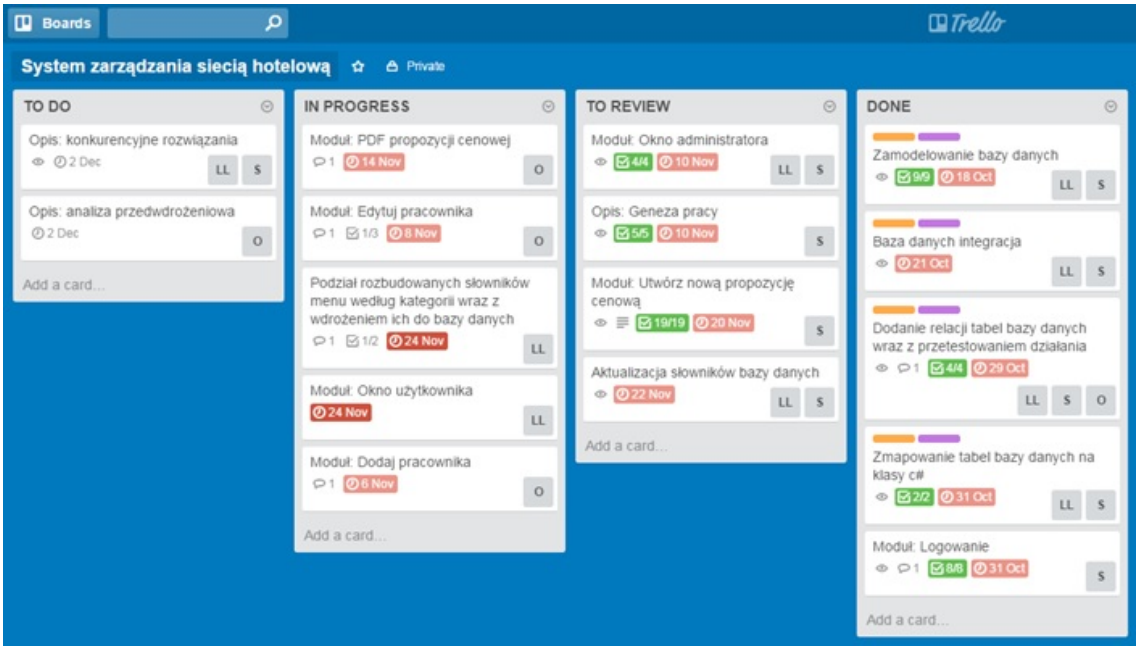
- *Object Explorer*

  Drugim ważnym elementem SQL Server Management Studio jest Object Explorer umożliwiający przeglądanie wielu instancji bazy danych. W tym też elemencie jest możliwość generowania diagramów baz danych, projektowania całej bazy danych. Wyświetla podgląd tabel wyzwalacze, procedury. Podgląd tabel daje też możliwość edycji danych w tabeli jak i wybierania pieszych wierszy w danej tabeli. Ułatwia to sprawdzenie zachowań aplikacji na dane testowe. Dzięki graficznemu interfejsowi jest możliwe w bardzo prosty sposób usuwanie bazy danych poprzez naciśnięcie przycisku „DELETE”. Umożliwia generowanie skryptu dzięki, któremu można odtworzyć całą strukturę bazy danych

4.2.3. Trello

Podczas pracy zespołowej napotyka się liczne problemy związane z organizacją pracy, jej synchronizacją oraz podziałem obowiązków. Stopień zaawansowania przedstawionych problemów dodatkowo wzrasta, w sytuacji, gdy współpraca zespołu przebiega zdalnie. W takich okolicznościach warto posłużyć się dedykowanymi

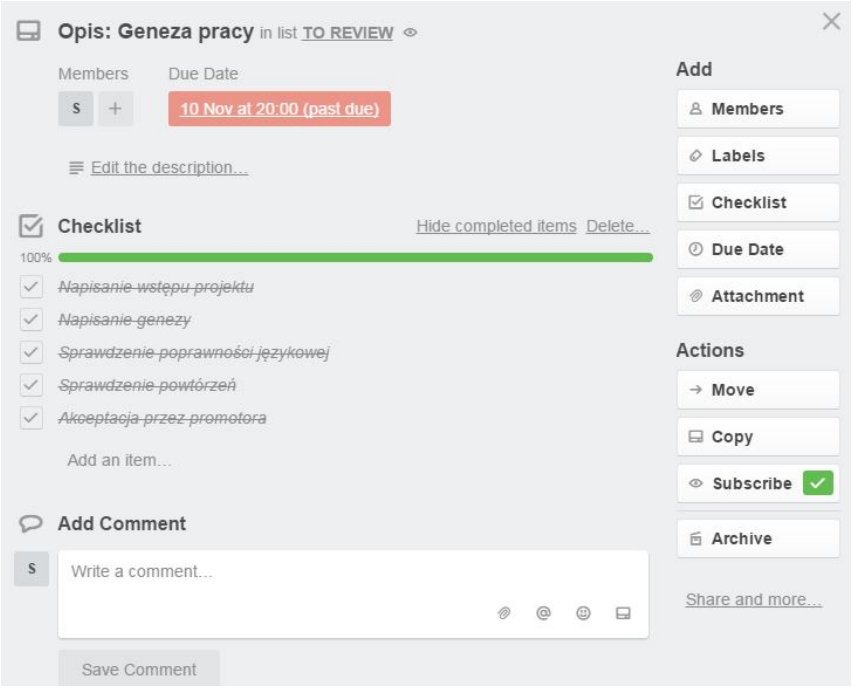
narzędziami, które umożliwiają prostą, szybką i skuteczną komunikację wspierającą pracę wielu osób nad jednym projektem. Trello jest to darmowa aplikacja, której przeznaczeniem jest wspieranie organizacji pracy grupowej opierając się na metodyce Kanban [27], która zaadoptowana została na potrzeby inżynierii oprogramowania. Potencjalny wzrost wydajności pracy przy użyciu Trello ma nastąpić dzięki scentralizowanemu systemowi zarządzania projektem opierającemu się na wirtualnych tablicach oraz ulokowanych na nich list zadań, które można dostrzec na rysunku 5



Rysunek 5: Interfejs aplikacji webowej Trello, Źródło: Opracowanie własne.

Listy te umożliwiają konstruowanie zadań w postaci tak zwanych kafelek, które w intuicyjny sposób można przenosić pomiędzy listami za pomocą techniki drag-and-drop. Każda kafelka reprezentująca zadanie w Trello posiada szereg dodatkowych funkcjonalności, które można dostrzec na załączonym rysunku 6. W czasie pracy nad projektem wykorzystano opcję przypisania konkretnej osoby lub grupy osób do zadania, co jednoznacznie determinowało odpowiedzialność jego wykonania oraz funkcję utworzenia listy czynności, które należy wykonać w celu realizacji zadania. Dodatkowym ważnym aspektem jest data, która po wspólnych ustaleniach przypisywana jest do konkretnego zadania i stanowi ostateczny termin wprowadzenia danej funkcjonalności.

Ciekawą funkcjonalnością godną przedstawienia są także powiadomienia zmian,



Rysunek 6: Szczegóły zadania na pojedynczej kafelce., Źródło: Opracowanie własne.

które w sposób automatyczny przekazywane są na skrzynkę elektroniczną lub jako powiadomienia push na urządzenia przenośne. Dzięki temu, iż narzędzie Trello dostępne jest w postaci webowej, a także jako aplikacja mobilna na Androida, iOS oraz Windows 8 [28], dostęp do niej jest w znaczny sposób ułatwiony. Umożliwia to otrzymanie niemal natychmiastowej informacji o naniesionych zmianach w projekcie lub zbliżającym się terminie ukończenia zadania, do którego jesteśmy przypisani.

Wszystkie przedstawione funkcjonalności narzędzia Trello umożliwiły zespołowi uproszczenie i przyspieszenie procesu organizacji pracy nad projektem co w konsekwencji doprowadziło do oszczędności czasowych, które przeznaczone zostały na elementy implementacyjne budowanego systemu.

4.2.4. Git/SourceTree/GitHub

Oprogramowanie wytwarzane jest przez kilkuosobowe zespoły programistyczne, a pliki źródłowe projektów przechowywane są na serwerach. Jednym z aspektów pracy zespołowej jest łączenie równolegle wytwarzanego kodu źródłowego przez poszczególnych programistów. W tym celu zespoły używają systemów kontroli wersji (ang. version control system) które umożliwiają integrację zmian wykonanych na plikach. W prostym języku oznacza to, że programista pobiera aktualną wersję plików i odsyła je do serwera wraz z nowym kodem źródłowym [29]. Istnieją trzy rodzaje systemów kontroli wersji [30]:

- *Lokalne (np. RCS)*

  Umożliwiają kontrolę plików tylko na lokalnych komputerach.

- *Scentralizowane (np. SVN)*

  Oparte o architekturę typu klient-serwer. Poprzednie wersje plików przechowywane są tylko na serwerze.

- *Rozproszone (np. Git)*

  Opartę o architekturę P2P. Poprzednie wersje plików przechowywane są na serwerze i w lokalnych repozytoriach, co zabezpiecza przed utratą danych.

Popularnym systemem kontroli wersji jest Git, wydany na darmowej licencji. Został stworzony przez Linusa Torvalds'a, po raz piewszy wydany w 2005 roku [31]. Posiada własną stronę internetową, zawierającą dokumentacje oraz pliki binarne programu [32]. Jest wieloplatformowy, posiada dystrybucje dedykowane zarówno dla systemów rodziny Linux, jak i Windows czy OS X. Posiada następujące zalety, które zdecydowały o użyciu systemu do synchronizacji zmian w niniejszym projekcie [33]:

- *Bezpieczeństwo danych*

  W razie awarii serwera, można odtworzyć poprzednie wersje plików na podstawie lokalnej kopii.

- *Równoległa praca*

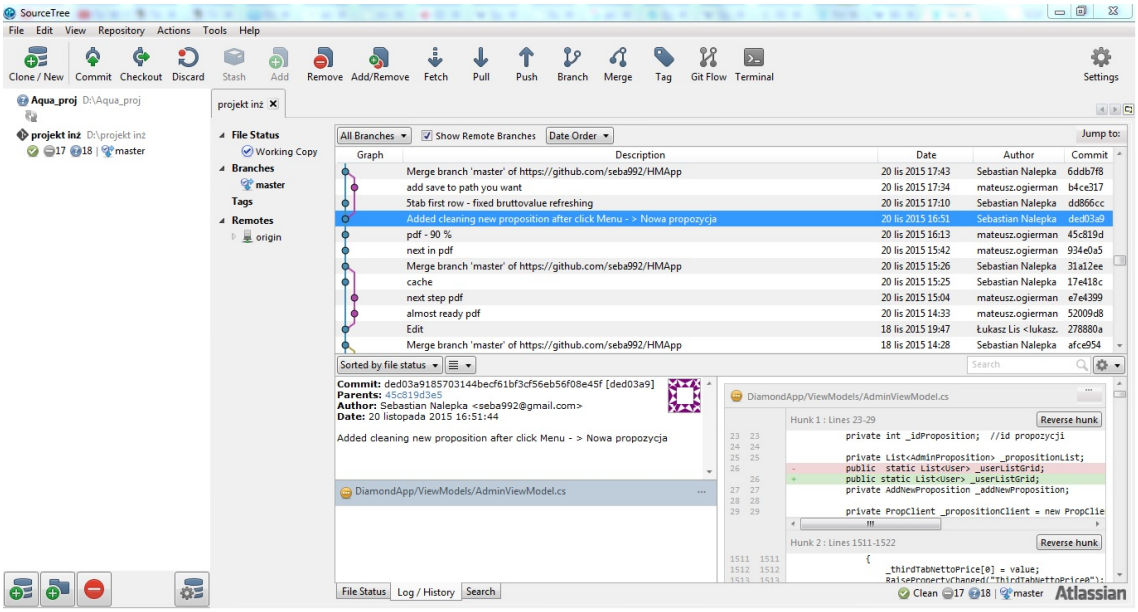
Git umożliwia niezależny rozwój oprogramowania, poprzez tworzenie tzw. gałęzi (ang. branch). Zazwyczaj projekty dzielone są na gałęzie poszczególnych programistów i gałąź główna tzw. master branch. Gałęzie mogą być łączone (ang. merge) w dowolnym momencie.

- *Oszczędność przestrzeni dyskowej*

  W porównaniu do SVN, gdzie każda gałąź jest kopią całego projektu Git posiada mechanizmy synchronizujące różnice pomiędzy poszczególnymi wersjami plików.

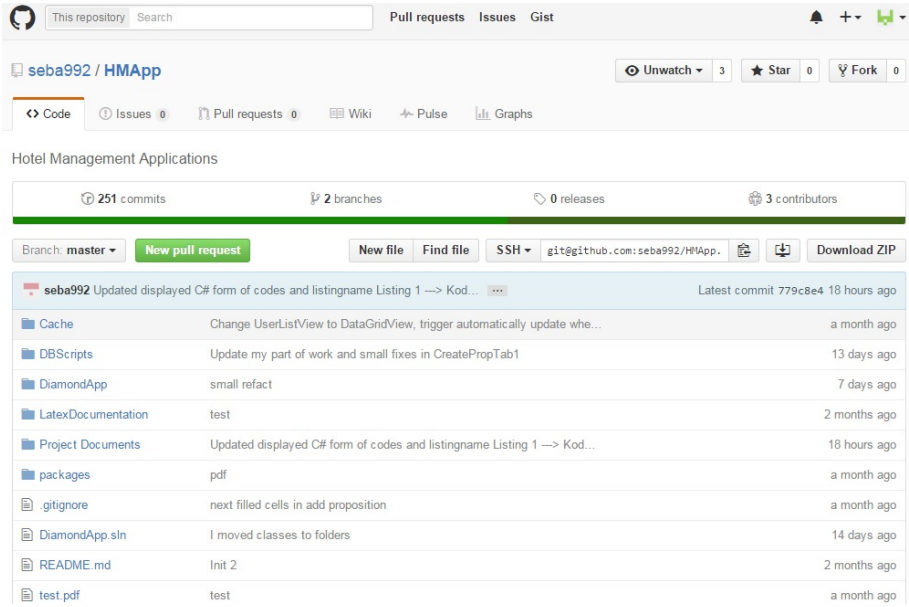
Historia zmian dokonanych w plikach źródłowych zapisywana jest za pomocą polecenia commit, które nadaje identyfikator aktualnej wersji projektu, opis zmian (opcjonalnie) oraz spis zmodyfikowanych plików. Pliki na serwer wysyłane są za pomocą komendy push. Pobieranie plików zamieszczanych na serwerze przez pozostałych członków zespołu umożliwia komenda fetch, natomiast synchronizację zmian z lokalną kopią umożliwia komenda pull. Git jest programem działającym w linii poleceń. Istnieje również graficzna nakładka na Git'a, którą jest Atlassian SourceTree. Jest on darmowym programem, jednak po upływie trzydziestu dni od instalacji, użytkownik zobowiązany jest do rejestracji konta w celu dalszego korzystania z aplikacji [34]. Posiada dystrybucje dla systemu Windows oraz OS X. Oprócz standardowych funkcjonalności Git'a umożliwia podgląd zmian w plikach dla poszczególnych commit'ów [35]. Podczas pracy nad projektem programistycznym, prawdopodobną sytuacją jest dokonanie zmian na tym samym pliku przez dwóch programistów. Jeśli zmiany wzajemnie się wykluczają, problem można rozwiązać za pomocą zewnętrznego narzędzia do rozwiązywania konfliktów SVN. Umożliwia on użytkownikowi porównanie różnic w poszczególnych wersjach pliku. Programista wybiera bloki kodu źródłowego które mają zostać zapisane w pliku, zatwierdza zmiany a konflikt zostaje oznaczony jako rozwiązany. Rysunek 7 przedstawia okno główne SourceTree.

GitHub.com jest darmowym serwisem internetowym który pozwala na przechowywanie projektów programistycznych, wspiera system kontroli wersji Git. Pierwsza wersja została wydana, w kwietniu 2008 roku [36]. GitHub umożliwia między innymi tworzenie i zarządzanie repozytoriami (np. nadawanie uprawnień dostępu współpracownikom), podgląd plików danego projektu. Rysunek 8 przedstawia repozytorium niniejszego projektu. Ciekawą funkcjonalnością serwisu przedstawioną na rysunku

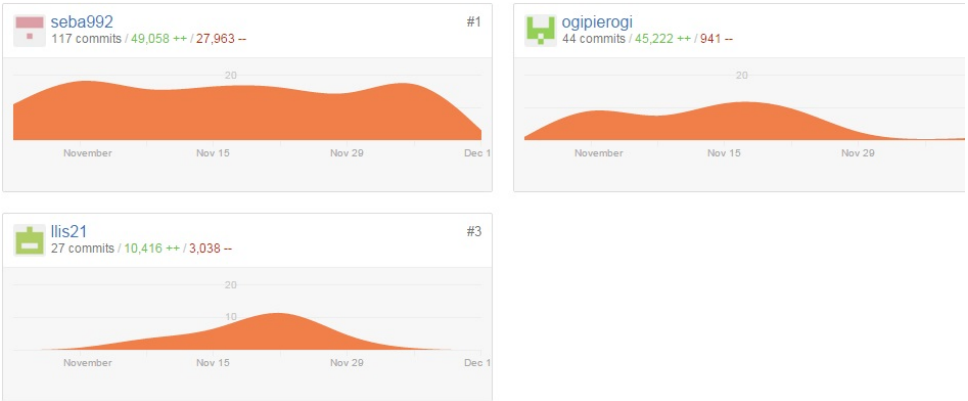


Rysunek 7: Atlassian SourceTree - okno główne, Źródło: Opracowanie własne.

9, jest opracowywanie statystyk użytkowników, i prezentowanie ich wkładu w konkretny projekt w formie wykresów.



Rysunek 8: Interfejs repozytorium w witrynie GiHub.com, Źródło: Opracowanie własne.



Rysunek 9: Statystyka wkładu pracy z dnia 13 grudnia 2015 r., Źródło: Opracowanie własne.

4.2.5. ReSharper

Wydajność pracy w projektach developerskich jest kluczową kwestią, która bezpośrednio przekłada się na korzyści finansowe i zaoszczędzony czas. Istnieją narzędzia, których przeznaczeniem jest zwiększenie efektywności pracy programisty poprzez zautomatyzowanie często powtarzających się czynności oraz nadzorowanie pisanego kodu według ustalonych wcześniej zasad.

ReSharper jest to narzędzie zaprojektowane z myślą o programistach .NET [37] pracujących w Microsoft Visual Studio, które w znacznym stopniu rozszerza dostępną funkcjonalność wyżej wymienionego środowiska, ułatwiając tym samym pisanie oraz refaktoryzację kodu.

Funkcje dostępne z poziomu ReSharpera można podzielić na kilkanaście modułów, z których jednym z najważniejszych jest moduł zajmujący się inspekcją kodu. Podczas pracy programisty z uruchomionym w tle ReSharperem, w czasie rzeczywistym sprawdzane jest ponad 1700 reguł dotyczących prawidłowości kodu i w sytuacji znalezienia nieścisłości, znaleziony wyjątek natychmiast wyświetlany jest na poziomie graficznego interfejsu Visual Studio z dokładnym jego opisem i miejscem wystąpienia. Wyjątki te dotyczą między innymi możliwości zastąpienia fragmentu kodu jego wydajniejszą wersją, ostrzeżeniem programisty przed kodem, który potencjalnie może doprowadzić do nieprawidłowego działania całego oprogramowania lub informacją o fragmencie, który jest kodem martwym [38]. Przykład kodu, który spowodować może nieprawidłowe działanie aplikacji oraz przykładowe jego rozwiązania dostępne są do wglądu na rysunku 10.

Rysunek 10: Informacja Resharpera dotycząca potencjalnej możliwości wystąpienia wyjątku NullReferenceException, Źródło: Opracowanie własne.

Kolejnym bardzo istotnym modułem jest moduł generowania kodu. Podczas programowania wiele czynności takich jak pisanie nowych klas, metod, implementacja interfejsów czy zmiana nazw wielokrotnie się powtarza. Dzięki odpowiednim skrótom klawiszowym wszystkie te elementy ReSharper wykona za programistę. Prze-

niesie on wybraną klasę do oddzielnego pliku, utworzy na podstawie nazwy i jej typu szablon brakującej metody, wygeneruje wymagane przy dziedziczeniu interfejsu wszystkie jego elementy oraz automatycznie zamieni wybraną nazwę na nową, w każdym miejscu jej występowania. Szczególnie ostatnia opcja w pracy przy rozbudowanym systemie jest wyjątkowo przydatna i znacznie skraca czas wykonywanej czynności.

Przedstawione funkcjonalności ReSharpera wraz z wieloma innymi dostępnymi do zapoznania się pod [39] elementami, umożliwiły skrócenie etapu implementacji wymaganych w projekcie modułów poprzez zautomatyzowanie powtarzających się czynności i zniwelowanie pojawiających się zagrożeń już we wczesnej fazie pisania kodu. Korzyścią płynącą z tego rozwiązania była oszczędność czasowa, która umożliwiła zwrócenie większej uwagi na aspekt przetestowania zastosowanych rozwiązań.

4.2.6. Jenkins

Jenkins Continuous Integration Server jest darmowym, wieloplatformowym narzędziem służacym do ciągłej integracji projektów programistycznych. Został napisany w języku Java, wywodzi się od narzędzia Hudson. Został wydany po raz pierwszy w 2011 r. [40]. Posiada szereg wtyczek zapewniających funkcjonalności takie jak statyczna analiza kodu, automatyczne tworzenie dokumentacji kodu źródłowego, powiadomienia e-mail, pobieranie plików z repozytoriów. Pozwala na budowanie oraz testowanie projektów w popularnych językach programowania, m. in. w jezyku C# czy Java. Jenkins jest aplikacją internetową, domyślnie działającą na porcie 8080. Zadania do wykonania definiowane są w postaci planów (ang. job) które mogą być uruchamiane cyklicznie, zgodnie z określonym harmonogramem lub manualnie. Plany wykonywane są w tzw. obszarach roboczych (ang. wokrspace) które są katalogami na dysku [41].

4.3. Biblioteki zewnętrzne

4.3.1. Extended WPF Toolkit

Ważnym aspektem każdego rodzaju oprogramowania użytkowego jest stopień trudności jego obsługi. Interfejs graficzny aplikacji powinien być budowany w sposób przejrzysty i intuicyjny dla użytkownika w celu bezproblemowego korzystania

z wszystkich jej funkcjonalności. Każdy element programu, który przysparza użytkownikowi problem w obsłudze jest potencjalnym fragmentem, który może doprowadzić do nieprawidłowego działania oprogramowania. Dla przykładu pole, do którego użytkownik zobowiązany jest wprowadzić datę może przysporzyć sporo problemów w sytuacji, gdy jest ono zwykłym polem tekstowym. W takim przypadku nie wiadomo jaki jest pożądany format daty, co implikuje konieczność dokładnej walidacji wpisanych przez użytkownika ciągów znaków oraz informacji zwrotnej w sytuacji błędnej ich formy.

Domyślnie technologia WPF podobnie jak WinForms posiada wbudowany pakiet kontrolek graficznych wraz z ich licznymi własnościami i zdarzeniami. Własności te umożliwiają określenie dokładnej formy użytej kontrolki dotyczącej jej koloru, wielkości, położenia, wyrównania, nazwy oraz widoczności. Zdarzenia kontrolki z kolei odpowiadają za jej reakcje w czasie interakcji z użytkownikiem oprogramowania w którym została zastosowana. Zdarzenia te dotyczą dla przykładu kliknięcia w kontrolkę, przesunięcia kursora myszki nad nią oraz jej przesunięcia.

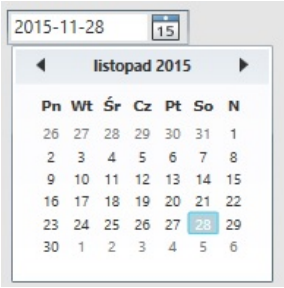
Liczba domyślnych kontrolek dla zaimplementowania niektórych funkcjonalności nie jest jednak wystarczająca. Odwołując się do przytoczonego przykładu wpisania daty przez użytkownika, WPF wymaga od programisty dużego wkładu pracy do prawidłowego wykonania tej funkcjonalności. Z pomocą przychodzą jednak dedykowane biblioteki rozszerzające, z których jedną jest Extended WPF Toolkit.

Extended WPF Toolkit jest to darmowa biblioteka przeznaczona dla technologii WPF, która rozpowszechniana jest na zasadach Microsoft Public Licence [42]. Biblioteka ta w znacznym zakresie rozszerza liczbę domyślnych kontrolek umożliwiając szybką i prostą implementację licznych funkcjonalności, których nie umożliwiały kontrolki domyślne.

Jednym z elementów, dla którego przedstawiona biblioteka znalazła zastosowanie w projekcie jest wpisywanie przez użytkownika wartości liczbowej do ustalonego pola. Kontrolka IntegerUpDown umożliwia programiście zapisanie w jej własnościach zakresu przyjmowanych liczb, które niwelują konieczność implementacji dodatkowej walidacji. Dodając do tego domyślne funkcje powyższej kontrolki odnoszące się do wewnętrznego sprawdzania typu wpisanych przez użytkownika znaków, podkreślania i nieprzyjmowania błędnych danych oraz dwóch graficznych przycisków, które umożliwiają po ich naciśnięciu zmiany wpisanej wartości, kontrolka ta doskonale nadaje się do przedstawionej funkcjonalności odciążając przy tym w znaczny sposób

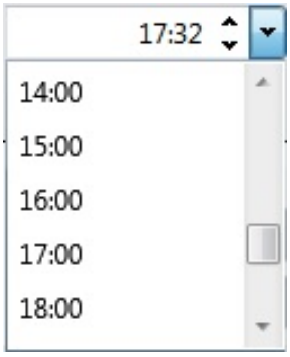
programistę.

Kolejnym miejscem zastosowania biblioteki Extended WPF Toolkit w projekcie jest ustalanie z poziomu programu daty oraz godziny. Do tych celów posłużono się dwiema dodatkowymi kontrolkami DatePicker oraz TimePicker. Kontrolka DatePicker, której wygląd przedstawiony został na rysunku 11, umożliwia użytkownikowi wybór daty w ustandaryzowanym formacie. Po operacji naciśnięcia kursorem myszki w przedstawianą kontrolkę wyświetlany jest przejrzysty kalendarz, w którym proces wyboru daty polega na wybraniu przez użytkownika odpowiedniego roku, miesiąca, a następnie kliknięcia w dzień odpowiadający jego potrzebom. Istnieje także możliwość wpisania ręcznie daty, z tą różnicą, iż musi być ona podana w formacie RRRR-MM-DD. Zapis daty w innych formatach zostanie automatycznie wykryty przez mechanizmy wewnętrzne kontrolki jako niepoprawny, a wpisana wartość usunięta.



Rysunek 11: Kontrolka DatePicker, Źródło: Opracowanie własne.

Walidacja godziny, analogicznie jak daty, także może doprowadzić do wielu problemów na tle jednolitego formatu danych. Zastosowana kontrolka TimePicker widoczna na rysunku 12 niweluje ten problem poprzez wprowadzenie mechanizmów czuwających nad prawidłowym formatem wpisanej godziny i w przypadku błędnego jej wprowadzenia, podobnie jak DatePicker, usuwa ją. Użytkownik samodzielnie może wpisać godzinę w formacie GG:MM lub za pomocą przycisków odpowiednio ustalić jej wartości.



Rysunek 12: Kontrolka TimePicker, Źródło: Opracowanie własne.

4.3.2. PDFsharp & MigraDoc Foundation

PDF (ang. Portable Document Format) czyli tzw. przenośny format dokumentów, został zaprojektowany i wdrożony przez firmę Adobe Systems w 1992 r. Dokumenty przechowywane w tym formacie mogą zawierać proste dane tekstowe, jak również dźwięki, grafiki czy tabele. Jedną z zalet formatu jest jego uniwersalność, wygląd pliku jest spójny niezależnie od systemu operacyjnego użytkownika [43]. Darmowe narzędzia, takie jak Adobe Acrobat Reader, Foxit Reader umożliwiają podgląd plików PDF. Większość pakietów biurowych takich jak, Microsoft Office czy Apache OpenOffice umożliwia zapis plików do formatu PDF. Popularność PDF stale wzrasta [44]. Wszelkie rachunki, faktury, prace naukowe są archiwizowane w postaci plików PDF.

Jednym z celów projektu, jest przedstawienie szczegółów transakcji pomiędzy klientem, a sprzedawcą w formie lekkiego i przejrzystego dokumentu. Format PDF spełnia założone wymagania. Do implementacji modułu tworzącego dokument wykorzystano bibliotekę PDFSharp & Migradoc Foundation dla języka C#.

PDFSharp & Migradoc Foundation jest darmową biblioteką umożliwiającą tworzenie dokumentów PDF w językach programowania z rodziny .NET. Oferowane funkcjonalności to m. in. tworzenie paragrafów, tabel, formatowanie tekstu, załączanie oraz skalowanie grafiki. Projekt posiada dedykowaną stronę internetową zawierającą dokumentację i przykładowe programy wraz z kodami źródłowymi [45]. Instalacja w Microsoft Visual Studio jest możliwa za pomocą menedżera pakietów Nuget.

5. Implementacja

5.1. Architektura aplikacji

W inżynierii oprogramowania analogicznie jak w innych dziedzinach, w których przeprowadzana jest operacja budowania zadanego przedmiotu, należy starannie zaplanować jego proces. Prezentowany rozdział zawiera opis metodologii, która umożliwia w uporządkowany sposób przeprowadzenie procesu budowania aplikacji wraz z zastosowaną w prezentowanym projekcie jego odmianą. Rozdział 5.1.3 przedstawia również efekt zastosowanego rozwiązania widoczny z poziomu projektu programistycznego.

5.1.1. Wzorce architektoniczne oprogramowania

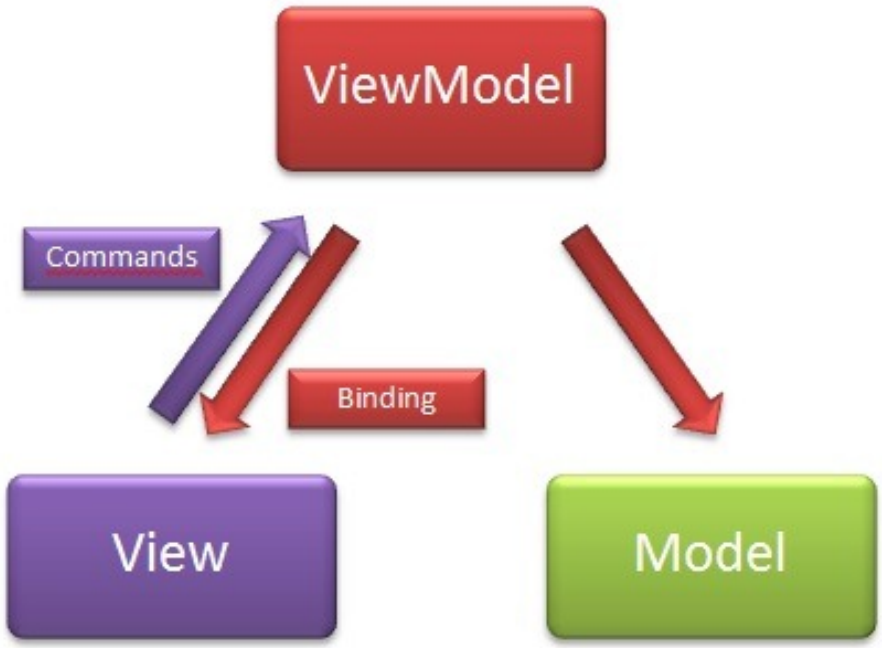
W celu przeprowadzenia procesu budowy aplikacji w sposób umożliwiający jego efektywny rozwój oraz dokonywanie zmian we funkcjonalnościach należy kierować się przyjętymi fazami produkcji oprogramowania [46]. Fazy te mówią o tym, iż należy dokładnie określić wymagania, które powinno budowane oprogramowanie spełniać, a następnie ustalić jego ogólną architekturę. Punkt ten jest niebywale ważny z racji na problematyczność zmiany architektury zbudowanego już systemu. W celu zobrazowania tego zagadnienia można przywołać operację budowy budynku. Wszelkie zmiany dotyczące jego konstrukcji są ciężkie do zrealizowania w sytuacji, w której jest on już fizycznie gotowy. Kolejne fazy polegają na realizacji opisanej wcześniej architektury poprzez implementację wszystkich komponentów oprogramowania wraz z ich wzajemnymi połączeniami, przetestowaniem całego systemu oraz jego finalnemu uruchomieniu wraz z usuwaniem wykrytych podczas jego działania błędów.

Odnosząc się do planowania architektury systemu bardzo pomocne okazują się dostępne wzorce architektoniczne. Wzorce te są to sprawdzone oraz zaakceptowane sposoby rozwiązania danego problemu z dziedziny architektury oprogramowania, określające ogólną strukturę systemu informatycznego, zasady komunikacji pomiędzy komponentami oraz elementy wchodzące w jego skład wraz z opisem ich funkcji. Wybór odpowiedniego wzorca w dużej mierze zależny jest od wykorzystanej w projekcie technologii. W przypadku WPF powszechnie uznanym rozwiązaniem jest

MVVM (Model View ViewModel) [47], który jest zmodyfikowaną wersją wzorca MVC, zawierającą specjalizację modelu prezentacji.

5.1.2. Zastosowany wzorzec architektoniczny – MVVM

MVVM jest to wzorzec, który dzięki funkcjonalności odseparowania warstwy prezentacji od warstwy logiki biznesowej, umożliwia napisanie łatwo testowalnej oraz prostej w rozbudowanie aplikacji, której fragment kodu może zostać ponownie użyty w innych projektach. Opisywany wzorzec cieszy się dużą popularnością w gronie developerów WPF z powodu możliwości wykorzystania największych atutów tej technologii, takich jak komendy (command), wiązania (binding) oraz zachowania (behavior). Kod aplikacji o architekturze Model-View-ViewModel podzielony jest, zgodnie z jego nazwą na trzy oddzielne warstwy Model, View oraz ViewModel, z których każda przechowuje dedykowane dla siebie dane i spełnia określone funkcje. Z graficznym podziałem wraz z kierunkiem przepływu danych można zapoznać się pod rysunkiem 13.



Rysunek 13: Przepływ danych w modelu MVVM, Źródło: www.tomaszmalesza.pl

Model to warstwa, która odpowiedzialna jest w omawianym wzorcu za logikę biznesową aplikacji [48]. W przypadku przedstawianego projektu inżynierskiego

warstwa ta reprezentowana jest przez przez klasy, które utworzone zostały za pomocą biblioteki Entity Framework, służącej do odwzorowania relacyjnej bazy danych na obiekty dostępne z poziomu kodu. Wszelkie klasy zawierające dane, które maja przekazane być z poziomu modelu do warstwy widoku, z którego użytkownik może owe informacje odczytać zobowiązane są do implementacji interfejsu INotifyPropertyChanged lub INotifyCollectionChanged, który współpracuje z bindingiem [49] wykorzystywanym w WPF.

ViewModel zajmuje się reprezentacją danych, które wysyłane są do widoku, nie mając przy tym żadnej referencji do niego. Widok odnosi się do elementów ViewModelu za pośrednictwem komend oraz wspomnianych operacji bindowania, co zapewnia pełną separację warstwy ViewModelu i umożliwia przetestowanie jej odrębnie od modelu i części prezentującej.

Wastwa widoku (View) odpowiedzialna jest za wyświetlanie danych i pełni rolę wyłącznie prezentacyjną. Jej komunikacja z warstwą modelu przebiega za pośrednictwem warstwy pośredniczącej ViewModel, do której jest przyłączona za pomocą właściwości DataContext. Właściwość ta wstrzykuje zależności pomiędzy View oraz ViewModel, który ją kontroluje.

5.1.3. Model aplikacji

Pytanie do promotora: Opisywać podział klas, strukturę folderów aplikacji ? Jest to bardzo techniczna kwestia, nie wiem, czy potrzebna

TODO: after refactor. - opis warstw (model == baza danych) - podzial view/viewmodeli - ilustracja podzialu foldery+klasy

5.2. Architektura bazy danych

5.2.1. Definicja bazy danych

Bazy danych można porównać do archiwum firmy gdzie są przechowywane wszelakie projekty i dane osobowe. Tabelą w kontekście tego przykładu jest pojedynczy segregator zawierający dane o identycznej tematyce. Tabela składa się z wierszy dalej zwanych rekordami i kolumn. Rekord tworzy spójną całość tak jak dane jednej osoby, a kolumny to są poszczególne dane typu imię nazwisko pesel itp.. Tak jak pesel jest unikatowy dla osoby w firmie tak klucz główny w tabeli też musi być unikatowy. Połączenie dwóch segregatorów poprzez na przykład pesel danej osoby

to w drugim segregatorze nazywa się kluczem obcym, a samo połączenie relacją. Rozróżniamy kilka relacji:

1. Jeden do, wielu kiedy jedna koszulka w segregatorze łączy się z wieloma koszulkami w drugim.

2. Jeden do jednego występuje wtedy, kiedy jedna teczka łączy się tylko z teczką inną w pojedynczy sposób.

3. Wiele do wielu występuje wtedy, kiedy jedna koszulka z jednego segregatora odpowiada wielu w drugim i na odwrót.

W projekcie została wykorzystana baza relacyjna, która działa na powyższym przykładzie. Ważnym z punktu widzenia projektowania aplikacji bazodanowej jest odpowiednie zaprojektowanie struktury tabel i połączeń miedzy nim. Nie można lekceważyć tego procesu. Gdy się źle zaprojektuje bazę mogą wyniknąć w późniejszym czasie problemy z jej dalszym rozwojem. Istnieje jeszcze gorszy scenariusz, gdy w trakcie dalszych prac baza danych nie będzie spełniała swojego zadania. Efektem źle zaprojektowanej bazy danych może być długość wykonywanych zapytać jak i bardzo duża złożoność. Wiąże się to z koniecznością przeprojektowania od początku całej bazy lub starania się w programie ukryć te niedomagania.

5.2.2. MS-SQL – Zastosowany system zarządzania bazą danych

Z racji wykorzystania w projekcie baz danych istniała konieczność uruchomienia serwera bazodanowego. W sieci hotelowej wykorzystane są technologie Microsoftu. Produktem tej firmy związanym z powyższą tematyką jest serwer z rodziny MS-SQL[50]. Serwer ten umożliwia tworzenie baz danych dwóch typów relacyjny jak i nierelacyjnych. Tworzenie baz danych może się odbywać w dwojaki sposób poprzez:

1. SQL Server Management Studio, który jest nakładką graficzną

2. T-SQL.

Dzięki wbudowanym typom danych zabezpiecza to przed wpisaniem nieprawidłowej wartości w danym miejscu. Wbudowane mechanizmy kopii zapasowej plików i transakcji pozwalają na zapis ich na innym klastrze. W wypadku kiedy główny klaster zostanie uszkodzony zadanie przejmuje drugi klaster. Istnieje również możliwość

tworzenia testów rozwojowych bazy danych. Wykorzystując dziennik jest możliwe diagnozowanie problemów z działaniem bazy danych obciążenie.

W tym silniku bazodanowym można wyszczególnić kilka komponentów:

- *Interfejs T-SQL*

  Komponent ten odpowiada za interpretacje zapytania lub innych instrukcji napisanych w języku T-SQL i wysłanie wyników do użytkownika.

- *Podsystem bezpieczeństwa*

  Moduł ten odpowiada za dostęp od bazy danych można go konfigurować w dwojaki sposób wykorzystując SQL Server Management Studio lub T-SQL. Moduł ten umożliwia szyfrowanie całej bazy lub tylko wybranych kolumn.

- *SQL agent*

  SQL agent odpowiada za automatyzację części rzeczy między, innymi za tworzenie kopii zapasowej o określonej godzinie. Dzięki takiemu planowaniu jest możliwe równomierne rozłożenie obciążenia serwera. Pozwala wysyłanie wiadomości w przypadku jakiegoś krytycznego błędu, czy się wykonała jakiejś zaplanowana czynność danego dnia.

- *Replikant*

  Odpowiada on za rozpowszechnianie danych pomiędzy różnymi serwerami na przykład FTP lub też urządzeniami. Głównie ten moduł wykorzystuje się w przypadku serwer-serwer do tworzenie kopii zapasowej w tej sytuacji jeden serwer jest serwerem centralny a drugi kopią zapasową.

5.2.3. Budowa bazy danych użytej w projekcie

Bazy danych można porównać do archiwum firmy gdzie są przechowywane wszelakie projekty i dane osobowe. Tabelą w kontekście tego przykładu jest pojedynczy segregator zawierający dane o identycznej tematyce. Tabela składa się z wierszy dalej zwanych rekordami i kolumn. Rekord tworzy spójną całość tak jak dane jednej osoby, a kolumny to są poszczególne dane typu imię nazwisko pesel itp. Tak jak pesel jest unikatowy dla osoby w firmie tak klucz główny w tabeli też musi być unikatowy. Połączenie dwóch segregatorów poprzez na przykład pesel danej osoby to w drugim

segregatorze nazywa się kluczem obcym, a samo połączenie relacją. Rozróżniamy kilka relacji:

1. Jeden do, wielu kiedy jedna koszulka w segregatorze łączy się z wieloma koszulkami w drugim.

2. Jeden do jednego występuje wtedy, kiedy jedna teczka łączy się tylko z teczką inną w pojedynczy sposób.

3. Wiele do wielu występuje wtedy, kiedy jedna koszulka z jednego segregatora odpowiada wielu w drugim i na odwrót.

W projekcie została wykorzystana baza relacyjna, która działa na powyższym przykładzie. Ważnym z punktu widzenia projektowania aplikacji bazodanowej jest odpowiednie zaprojektowanie struktury tabel i połączeń miedzy nim. Nie można lekceważyć tego procesu. Gdy się źle zaprojektuje bazę mogą wyniknąć w późniejszym czasie problemy z jej dalszym rozwojem. Istnieje jeszcze gorszy scenariusz, gdy w trakcie dalszych prac baza danych nie będzie spełniała swojego zadania. Efektem źle zaprojektowanej bazy danych może być długość wykonywanych zapytać jak i bardzo duża złożoność. Wiąże się to z koniecznością przeprojektowania od początku całej bazy lub starania się w programie ukryć te niedomagania.

Bazę, która została wykorzystana w projekcie można podzielić na dwie główne części: części zawierającą domyślne wartość w dalszej części zwane słownikami i część powiązana z tworzoną propozycją cenową.

- *Część słownikowa*

  Tabele słownikowe przez wzgląd na funkcje, którą pełnią nie są powiązane z innymi tabelami Taki podział tabel na dwa osobne zbiory został spowodowany bezpieczeństwem. Jeśli istniałaby relacja łącząca te dwie tabele to zmiana wartość w tabeli słownikowej powodowałaby zmianę cen produktów w starszych propozycja. Efektem uniknięcia tego typu problemów jest rozdzielenie na dwie odseparowane części. Gdyby tak nie było wtedy możliwość odtworzenia starej propozycji cenowej byłaby niemożliwa, co jest efektem niepożądanym z punktu widzenia działania aplikacji.

- *Część Propozycji*

Drugi człon bazy danych stanowią tabele związane z tworzoną propozycją cenową. Są one podzielone w logiczny sposób uwzględniając podział przykładowej propozycji cenowej przedstawione przez sieć hotelową: produkty gastronomiczne, wyposażenie sali, szczegóły rezerwacji, szczegóły klienta. Jest to też najbardziej rozbudowana część bazy danych wiążą się też z nią konta użytkowników kto, którą propozycje cenową wystawił. Wiąże ze sobą konieczność nadawania uprawnień użytkownikom w systemie. Aktualnie istnieją tylko dwa typy sprzedawca i administrator. Do każdej stworzonej propozycji cenowej są też dołączane rabaty na poszczególne jej elementy: gastronomie, wyposażenie sali

Podział taki uwzględnia możliwość dalszego rozwoju aplikacji. Możliwe jest dodawanie nowych słowników która będą zwierać inne elementy związane z tworzoną propozycją cenowa. Dodanie nowej tabeli do słowników, która będzie wiązała się z nową grupy produktów.

5.3. Komunikacja bazy danych z projektem programistycznym

5.3.1. Mapowanie obiektowo-relacyjne

Pojęcie mapowania obiektowo-relacyjnego (ORM) odnosi się do programistycznego terminu dotyczącego współpracy z bazą danych, wykorzystując idee programowania obiektowego. Konkretnie rzecz ujmując chodzi o zamianę danych, przechowywanych w postaci tabelarycznej w relacyjnej bazie danych na postać obiektową dostępną z poziomu używanego języka programowania lub w drugą stronę.

Przedstawiona translacja danych okazała się bardzo praktyczna przez wzgląd na zniwelowanie konieczności zagłębiania się w struktury bazy danych przez programistę, a także w przypadku użycia dodatkowych frameworków, brak wymogu dotyczącego jego znajomości zapytań języka SQL przez fakt pracy wyłącznie na obiektach, które odwzorowują strukturę tabel.

Aplikacje komputerowe, których funkcjonalności zorientowane są na wielu użytkowników lub wymuszają konieczność ciągłego przechowywania i przetwarzania dużych ilości danych bardzo często korzystają z systemów bazodanowych. Systemy te umożliwiają nieprzerwane magazynowanie dużej ilości informacji, a także szybkie ich wyszukiwanie, sortowanie, dodawanie, edycję oraz usuwanie. Przedstawione aspekty

są bardzo trudne do zaimplementowania wewnątrz standardowych aplikacji.

W przypadku technologii .NET, w której prowadzony jest prezentowany projekt, standardowa komunikacja aplikacji z bazą danych odbywa się za pośrednictwem protokołu ADO.NET. Przy jego użyciu programista ma obowiązek każdorazowo nawiązać połączenie z bazą, ręcznie wprowadzić zapytanie SQL, odebrać wynik i zapisać go w odpowiednim obiekcie oraz zamknąć połączenie. Schemat ten w znacznym stopniu utrudnia utrzymanie prawidłowego działania aplikacji w przypadku jej rozbudowy lub modyfikacji bazy danych. Napisane zapytanie SQL w żadnym stopniu nie jest sprawdzane ze względu na jego poprawną formę, a w sytuacji zmiany przykładowo nazwy kolumny w wykorzystywanej bazie danych, programista w celu utrzymania prawidłowej pracy aplikacji ma obowiązek aktualizacji wszystkich zapytań, które dotyczyły zmienionej kolumny. Przedstawione zagadnienie wraz z porównaniem analogicznej operacji w technologii mapowania obiektowo-relacyjnej przedstawione zostało z poziomu kodu w kodzie źródłowym 1.

W celu przeciwdziałania przedstawionym problemom zdecydowano się zastosować dedykowany framework, który upraszcza developerowi przeprowadzanie wszelkich bazodanowych operacji.

```
//ADO.NET
//wymagane manualne dodanie pliku konfiguracji połączenia
SqlConnection connection =
    new SqlConnection(connectionString);
SqlCommand command =
    new SqlCommand(@"INSERT INTO [TestDb].[Users]
    VALUES (@Login, @Password)", connection);
command.Parameters.Add(
    new SqlParameter("Login", "TestLogin"));
command.Parameters.Add(
    new SqlParameter("Password","TestPa$$"));
connection.Open();
command.ExecuteNonQuery();
connection.Close();

//Entity Framework
```

```
// EF automatycznie odczyta plik konfiguracji połączenia
MyEntities dataContext = new MyEntities();
TestUser newUser = new TestUser
{ Login = "TestLogin", Password = "TestPa$$word" };
dataContext.Users.Add(newUser);
dataContext.SaveChanges();
```

Kod źródłowy 1: Porównanie operacji dodania użytkownika w ADO.NET oraz Entity Framework

5.3.2. Zastosowane narzędzie ORM – Entity Framework

Entity Framework jest to dedykowane dla platformy .NET narzędzie mapowania obiektowo-relacyjnego, które wspiera budowę trójwarstwowych aplikacji bazodanowych. Budowa obiektowego modelu bazy danych w prezentowanym frameworku może przebiegać w zależności od potrzeb trzema ścieżkami.

Pierwszą opcją jest podejście Database First, które okazuje się przydatne w sytuacji, gdy obecna jest działająca fizycznie baza danych. W tym przypadku, korzystając z wbudowanego we framework kreatora, istnieje możliwość określenia lokalizacji bazy oraz automatycznego jej mapowania, w wyniku którego wygenerowane zostają potrzebne klasy obiektowego modelu bazy danych.

Kolejną metodą jest Code First umożliwiające utworzenie fizycznego modelu bazy danych na podstawie własnoręcznie napisanych klas w języku C# wraz z jej odpowiednimi adnotacjami [51].

Ostatnie podejście to Model First, które polega na zbudowaniu fizycznego modelu bazy danych posługując się wbudowanemu we framework Designerowi, który w znacznym stopniu ułatwia tworzenie encji oraz właściwości bazy danych. Ścieżka ta jest wykorzystywana w sytuacji posiadania wyłącznie schematu bazy danych. Ze zbudowanego modelu generowany jest fizyczny model bazy danych jak i również klasy, które reprezentują model obiektowy.

5.3.3. Stosowanie poleceń bazodanowych ze strony projektu C# za pomocą Linq to Entities

Zastosowane narzędzie Entity Framework nie tylko w znacznym stopniu upraszcza połączenie bazy danych z projektem programistycznym ale także umożliwia w nim proste i intuicyjne wykonywanie poleceń bazodanowych. W projekcie w celu

zaimplementowania założonych funkcjonalności stosowano trzy podstawowe polecenia modyfikujące zawartość bazy: Insert, Update oraz Delete. Entity Framework domyślnie zawiera moduł *LINQ to Entities*, który umożliwia wykorzystanie składni technologii LINQ [52] do operowania na bazodanowym źródle danych.

W celu wykonania polecenia SELECT zwracającego rekordy z bazy danych należy za pomocą przedstawionego w kodzie źródłowym 2 schematu wybrać nazwę tabeli z obiektu kontekstu wygenerowanego przez Entity Framework, a następnie użyć polecenia *where* definiującego wyszukiwaną zależność. Finalnie poleceniem *select* następuje ustalenie zdefiniowanej wcześniej zwracanej zmiennej, w której przechowywane są wyszukane elementy. Tym sposobem dane przechowywane w bazie danych poddane zostają translacji na poziom obiektów, z których możliwe jest korzystanie z poziomu C# w sposób standardowy.

```
var exampleQuerry = (from records in _context.TableName
                    where table.ColumnName == _valueToFind
                    select records);
```

Kod źródłowy 2: Opis polecenia select

Edycja danych znajdujących się w bazie danych z poziomu projektu programistycznego realizowana jest poprzez modyfikację opisanego polecenia *SELECT*. Modyfikacja ta polega na przekształceniu typu otrzymanego obiektu na typ klasy modelu Entity, który reprezentuje edytowaną bazodanową tabelę. W przypadku operowania na jednym rekordzie, przekształcenie te należy dokonać posługując się metodą *SingleOrDefault*, natomiast w przypadku wielu rekordów metodą *ToList*, która dokona konwersji na listę obiektów pożądanego typu. Do takiego utworzonego obiektu, programista ma możliwość odwoływania się i dokonywania zmian w standardowy dla języka C# sposób. Po operacji modyfikacji, wszystkie zmiany należy potwierdzić metodą *SaveChanges* na obiekcie modelu Entity, która uaktualnia zmodyfikowane elementy na poziomie bazy danych.

Dodanie rekordu do bazy danych jest operacją, które nie wymaga znajomości składni LINQ to Entities. Operacja ta polega na utworzeniu obiektu klasy wygenerowanej przez Entity Framework, która reprezentuje bazodanową tabelę, do której nowy rekord ma być dodany, a następnie uzupełnieniu jego elementów danymi. Tak zbudowany obiekt należy dodać do obiektu kontekstu, odwołując się do jego właściwości o nazwie interesującej nas tabeli bazy danych, a następnie w parametrze

metody *Add* umieścić wcześniej zbudowany obiekt. Dokonane zmiany, analogicznie jak w przypadku edycji danych należy potwierdzić na obiekcie kontekstu. Przedstawienie opisanej operacji z poziomu kodu, dostępne jest również do wglądu w drugiej części kodu źródłowego 1.

6. Automatyzacja wydawania kolejnych wersji aplikacji

6.1. Continuous Integration/Continuous Delivery/Continuous Deployment

Ciągła integracja (ang. Continuous Integration, CI) jest praktyką stosowaną podczas procesu wytwarzania oprogramowania. Członkowie zespołów programistycznych synchronizują zmiany w kodzie źródłowym przynajmniej raz dziennie, dążąc do wielokrotnej intagracji jednego dnia. Każda integracja weryfikowana jest przez narzędzia do automatyzacji (ang. build automation), takie jak Jenkins. Dla każdej zmiany kodu źródłowego w repozytorium kompilacja, czy testy jednostkowe mogą zostać wykonane automatycznie, co prowadzi do wcześniejszego wykrywania błędów. Stosowanie tej praktyki umożliwia zmniejszenie ilości pracy podczas łączenia zmian projektów programistycznych [53].

Ciągła dostawa (ang. Continuous Delivery) jest podejściem w inżynierii oprogramowania którego podstawowym założeniem jest zapewnienie wysokiej jakości oraz poprawności działania cyklicznie wytwarzanego oprogramowania. Wynikiem procesu jest stabilna wersja oprogramowania, która może zostać dostarczona do klienta w dowolnym czasie [54]. Rozwinięciem praktyki jest ciągłe wdrażanie (ang. Continuous Deployment), które umożliwia automatyczne wydawanie aplikacji do lini produkcyjnej [55].

6.2. Programy i wtyczki użyte w Jenkins

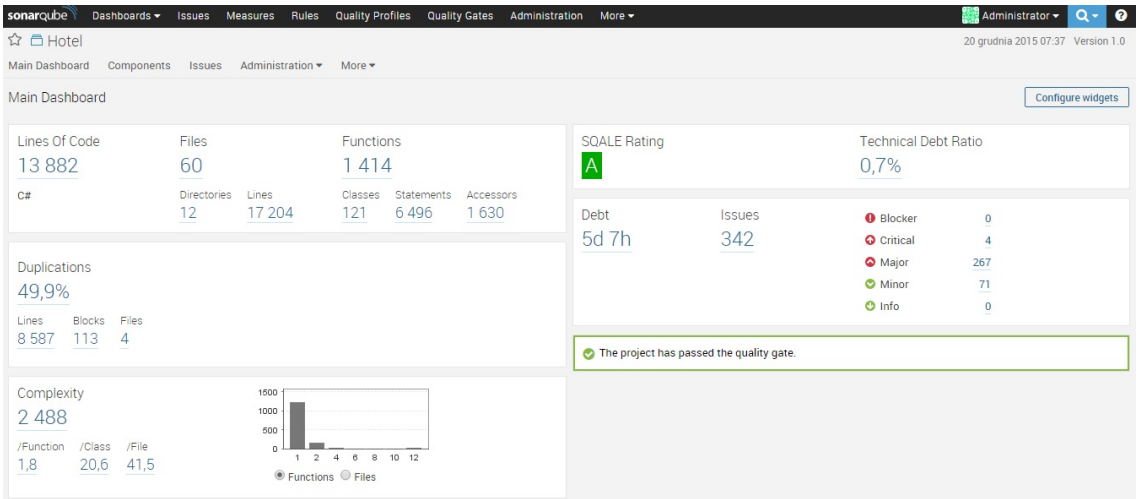
6.2.1. SonarQube

SonarQube jest darmową, wieloplatformową aplikacją służącą do zarządzania jakością kodu źródłowego. Została napisaną w językach Java i Ruby. Domyślnie działa na porcie 9000. Posiada szereg wtyczek zapewniających wsparcie dla popularnych języków programowania takich jak Java, C++, Objective-C, C# czy PHP [56], które umożliwiają statyczną analizę kodu. W przypadku niniejszego projektu użyto

wtyczkę SonarQube Scanner for MSBuild, która zawiera zbiór zasad określających poprawność kodu w języku C#. Wsparcie SonarQube w Jenkins'ie, umożliwia wykorzystywanie narzędzia w planach ciągłej integracji [57]. Podstawowe parametry wyliczane w analizie są następujące:

- *Liczba linii kodu źródłowego*
- *Liczba plików i katalogów*
- *Liczba funkcji, klas*
- *Liczba powtórzeń bloków kodu*
- *Liczba potencjalnych błędów i przybliżony czas potrzebny do ich rozwiązania*
- *Ocena jakości w skali SQALE*

Przykładowy raport analizy kodu niniejszego projektu został przedstawiony na rysunku 14.



Rysunek 14: Raport statycznej analizy kodu źródłowego, Źródło: Opracowanie własne.

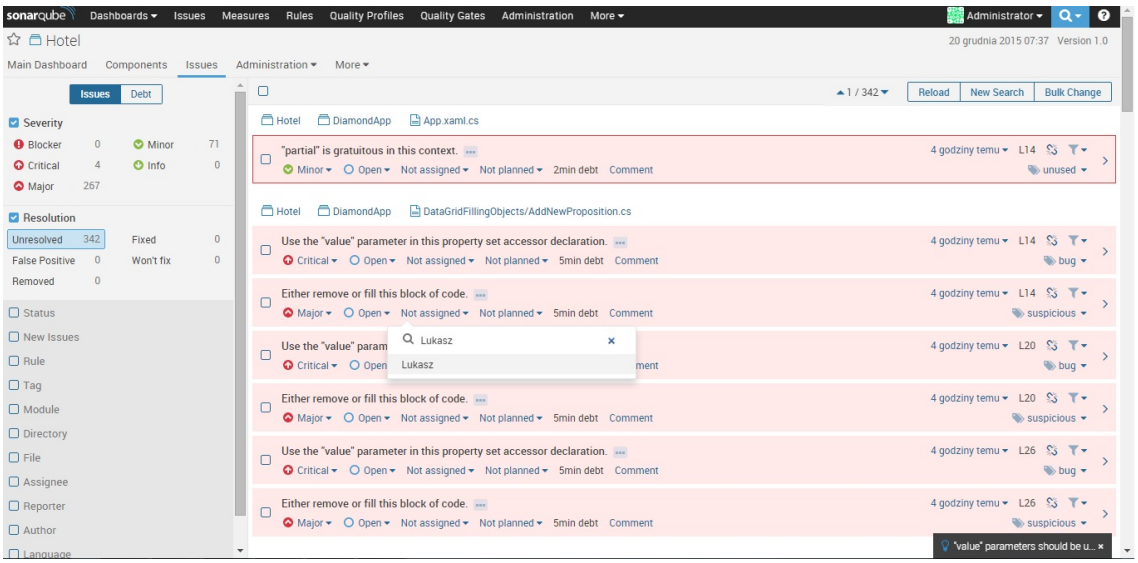
SonarQube na podstawie raportu analizy dokonuje oceny jakości kodu w skali SQALE (ang. Software Quality Assessment based on Lifecycle Expectations). Ocena jest wyliczana ze wzoru [58]:

$$\frac{\text{Liczba dni potrzebnych do naprawienia błędów}}{\text{Liczba dni potrzebnych do naprawy błędu z jednej linii kodu * Liczba linii kodu źródłowego}}$$

Wynik działania jest przedstawiany w procentach. Im mniejsza wartość, tym jakość kodu jest wyższa. Oceny odpowiadające wartościom wyliczonym ze wzoru są następujące:

- *A - mniej niż 10%*
- *B - od 11% do 20%*
- *C - od 21% do 50%*
- *D - od 51% do 100%*
- *E - powyżej 100%*

Potencjalne błędy zapisywane są na liście zadań, pokazanej na rysunku 15.



Rysunek 15: Lista zadań w SonarQube, Źródło: Opracowanie własne.

6.3. Schemat budowania projektu

Kolejne wersje niniejszego projektu są wydawane zgodnie z zasadami Continuous Delivery. Utworzono plan w Jenkins'ie który wykonuje następujące zadania:

- *Pobranie kodu źródłowego*

  Pliki zawierające kod źródłowy są pobierane z repozytorium do obszaru roboczego.

- *Rozpoczęcie statycznej analizy kodu*

  Za pomocą wtyczki SonarQube Scanner for MSBuild, SonarQube Scanner rozpoczyna statyczną analizę kodu źródłowego.

- *Aktualizacja bibliotek*

  W projekcie zostały użyte zewnętrzne biblioteki takie jak PDFSharp & Migradoc Foundation. Kod zostanie skompilowany poprawnie po przywróceniu (ang. restore) bibliotek przy użyciu NuGet Package Manager.

- *Kompilacja kodu źródłowego*

  Kod źródłowy zostaje skompilowany przy użyciu MSBuild, który jest kompilatorem zintegrowanego środowiska programistycznego Visual Studio.

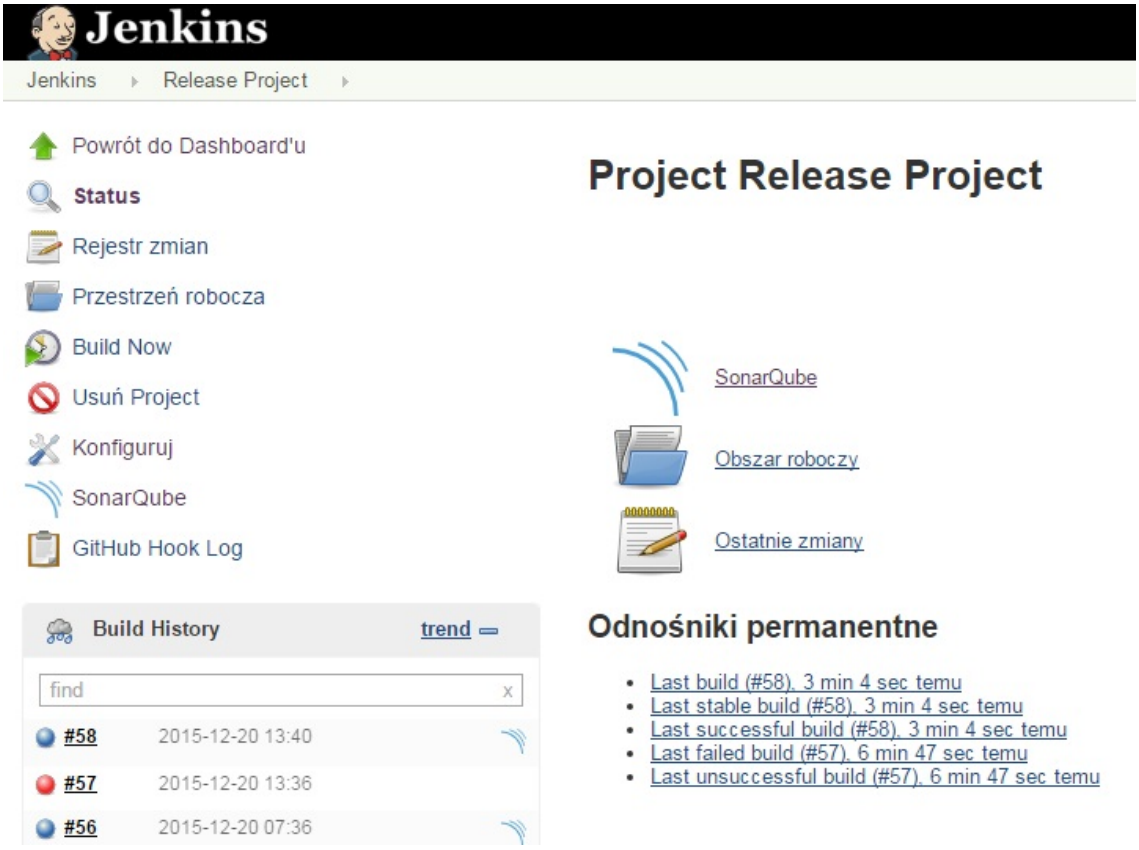
- *Raport analizy*

  Raport statycznej analizy kodu zostaje opublikowany w SonarQube.

- *Utworzenie paczki zip*

  Jeżeli kompilacja kodu się powiodła, zostaje utworzone archiwum w formacie zip. Zawiera plik wykonywalny, w formacie exe oraz niezbędne do poprawnego działania aplikacji pliki. Paczka zawierająca kolejną wersję aplikacji, może zostać dostarczona do potencjalnego klienta w dowolnym czasie.

Rysunek 16 przedstawia stronę wygenerowaną dla planu.



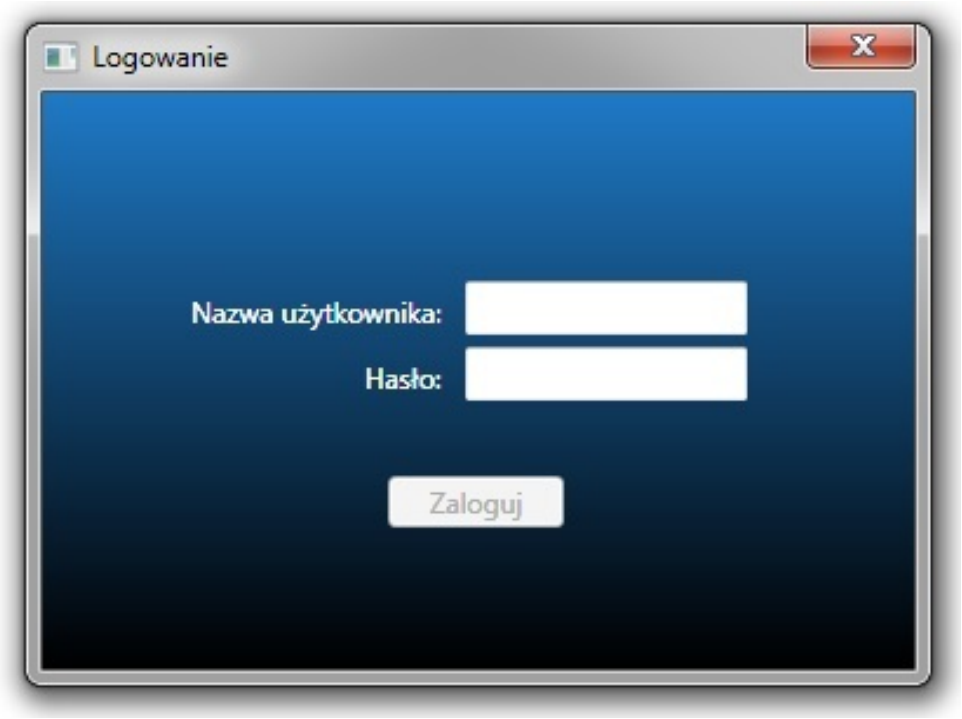
Rysunek 16: Plan ciągłej integracji w Jenkins'ie, Źródło: Opracowanie własne.

7. Testowanie aplikacji

7.1. Przeprowadzenie operacji logowania użytkownika

Podstawowym modułem każdej aplikacji wspierającej pracę wielu użytkowników na ich dedykowanych kontach jest moduł logowania. Po uruchomieniu aplikacji, która jest przedmiotem projektu, oczom użytkownika ukazuje się widocznie na rysunku 17 okno, którego zadaniem jest sprawdzenie jego tożsamości oraz w sytuacji jej potwierdzenia, przełączenie do odpowiedniego okna głównego aplikacji, w zależności od posiadanych uprawnień.

Proces logowania od strony użytkownika polega na wprowadzeniu z poziomu klawiatury swojej nazwy konta składającej się z imienia oraz nazwiska oddzielonych

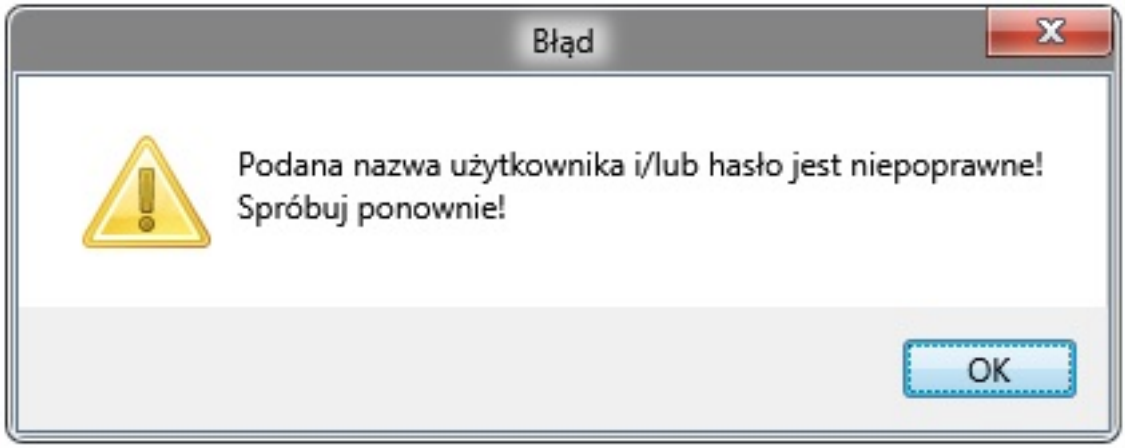


Rysunek 17: Okno procesu logowania, Źródło: Opracowanie własne.

kropką oraz hasła, które wybrano do zabezpieczenia swojego konta. Aplikacja na podstawie tych informacji przeprowadza operację ich weryfikacji, komunikując się z bazą danych i wyszukując w odpowiednich tabelach rekordy, które pasują do otrzymanego schematu. W zależności od rezultatu tej akcji, użytkownikowi zwracana jest odpowiednia odpowiedź.

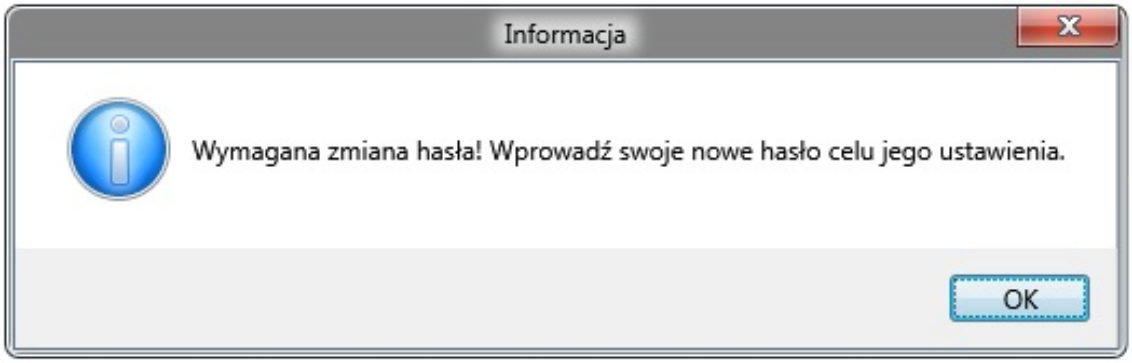
W sytuacji podania poprawnego loginu oraz hasła, aplikacja ukrywa okno logowania oraz uruchamia odpowiednie dla posiadanych uprawnień okno główne. Istnieją jednak inne schematy przeprowadzenia operacji logowania, takie jak wpisanie błędnego hasła lub loginu, nie uzupełnienie nazwy użytkownika lub pierwsze logowanie na konto, które zgodnie z ustalonymi przez zespół założeniami projektowymi ma umożliwiać ustalenie hasła przez logującego się użytkownika.

Wpisanie błędnego hasła lub nazwy konta wychwytywane jest przez mechanizm modułu logowania, który zwraca informację ukazaną na rysunku 18 o problemie wraz z jego opisem. W celu zwiększenia bezpieczeństwa wypisywana jest ogólna informacja o otrzymaniu błędnych danych, a nie wyszczególniona, która informacja została podana błędnie.

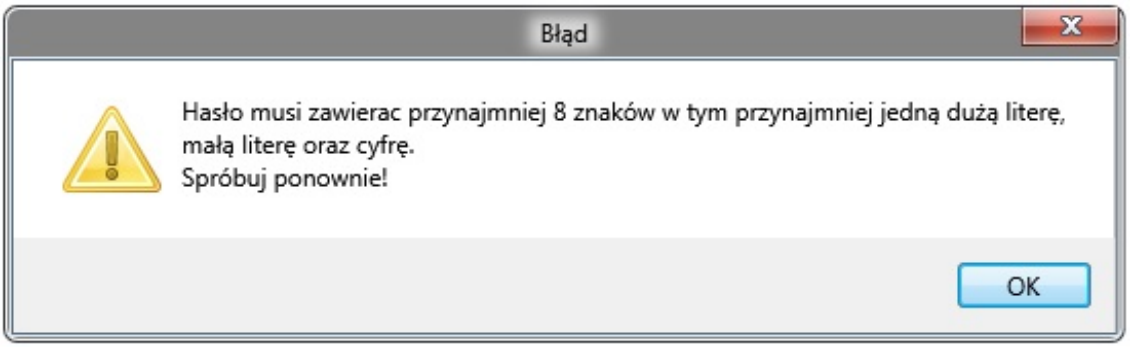


Rysunek 18: Informacja otrzymania błędnych danych podczas operacji logowania, Źródło: Opracowanie własne.

Pierwsze logowanie jest specyficzną formą logowania, w którym użytkownik posiada obowiązek wpisania hasła, z którego będzie korzystał w pracy z aplikacją. W chwili wpisania nazwy użytkownika, program w tle przeprowadza operację walidacji statusu konta i w sytuacji wykrycia, iż użytkownik loguje się pierwszy raz, wyświetla stosowną informację przedstawioną na rysunku 19, która instruuje go o dalszych krokach. Ustalone przez użytkownika hasło ma obowiązek spełniać określone jego zasady bezpieczeństwa dotyczące posiadania przynajmniej ośmiu znaków, co najmniej jednej dużej oraz małej litery, a także cyfry. Użytkownik w sytuacji wpisania hasła, nie spełniającej powyższej polityki bezpieczeństwa poinformowany zostanie stosowną informacją, która przedstawiona została na rysunku 20.



Rysunek 19: Okno informacji podczas pierwszego logowania, Źródło: Opracowanie własne.



Rysunek 20: Okno informujące o polityce bezpieczeństwa haseł, Źródło: Opracowanie własne.

7.2. Tworzenie nowej propozycji cenowej

Moduł tworzenia nowej propozycji cenowej umożliwia użytkownikowi aplikacji przeprowadzenie procesu utworzenia schematu wymaganych przez klienta usług wraz z ich zapisem do bazy danych. Tym sposobem wszelkie ustalenia dokonane z potencjalnym klientem zapisywane są na centralnym serwerze, z którego w razie potrzeby można wybraną propozycję pobrać i wygenerować jej plik w formacie PDF lub dokonać jej edycji.

W czasie kontaktu sprzedawcy z klientem, chcąc przedstawić ofertę cenową dotyczącą organizowanego przez klienta wydarzenia, sprzedawca posiada w menu głównym aplikacji opcję stworzenia nowej propozycji cenowej. W sytuacji wybrania przedstawionej opcji, oczom sprzedawcy ukazuje się forma przeznaczona do wpisania wymaganych danych. Forma ta przez wzgląd na kilka typów usług podzielona została na pięć zakładek, z których każda zawiera odpowiednia pola, umożliwiające ustalenie wszelkich elementów konkretnej usługi.

7.2.1. Opis klienta

Pierwsza zakładka, której okno graficzne dostępnej jest do wglądu przy rysunku 21 zawiera wszelkie informacje dotyczące danych osobowych sprzedawcy, klienta oraz ogólnych danych o organizowanym wydarzeniu. Dane osobowe sprzedawcy oraz informacje o firmie, którą reprezentuje, pobierane są automatycznie z bazy danych. Operacja ta jest możliwa do wykonania dzięki przechowywaniu w programie informacji o tym, na jakie konto użytkownik został zalogowany. Wszelkie pozostałe dane

sprzedawca ma obowiązek uzupełnić samodzielnie bazując na potrzebach klienta.

- *NAZWA FIRMY*

  Przez wzgląd na możliwość obsługi klientów biznesowych wprowadzone zostało pole umożliwiające wpisanie nazwy firmy, którą klient reprezentuje. W celu jednoznacznej identyfikacji każdej propozycji cenowej, uznano iż pole to będzie jedną z kolumn listy propozycji okna głównego.

- *ADRES KLIENTA*

  Pole to przeznaczone jest do wpisania adresu klienta zarówno w przypadku osoby indywidualnej jak i firmy. Po analizie wszelkich założeń projektowych nie dostrzeżono konieczności rozdzielenia danych adresowych - potrzebne są one tylko i wyłącznie do opisu klienta na wygenerowanym dokumencie PDF.

- *NIP*

  NIP jest kolejnym polem opcjonalnym, które przeznaczone jest do wpisania numeru identyfikacji podatkowej. Jego wypełnienie konieczne jest wyłącznie w sytuacji obsługi klienta reprezentującego firmę. W przypadku klienta prywatnego pole te należy pozostawić puste.

- *TERMIN WAŻNOŚCI PROPOZYCJI*

  Utworzona dla klienta propozycja cenowa posiada ustalony okres ważności, który domyślnie wykosi cztery dni od sporządzenia i zapisania propozycji cenowej przez sprzedawcę. Spowodowane jest to częstymi zmianami cenników i chęcią uniknięcia sytuacji, w której klient po długim okresie czasu będzie chciał z posiadanej propozycji cenowej skorzystać. Sprzedawca dzięki modułowi edycji propozycji posiada możliwość aktualizacji terminu ważności propozycji, dzięki czemu utworzona propozycja cenowa nie musi być tworzona ponownie od podstaw.

- *KONTAKT ZE STRONY ZAMAWIAJĄCEGO*

  Jest to podkategoria zawierająca cztery pola tekstowe, które przeznaczone są do wpisania imienia i nazwiska klienta, jego numeru telefonu oraz adresu e-mail. Ostatnie pole tekstowe (*Osoba decyzyjna*) umożliwia wpisanie osoby, która posiadać będzie uprawnienie do dokupywania usług nieuwzględnionych

w propozycji cenowej w czasie trwania wydarzenia. Zdarza się, iż podczas jego trwania, organizator chce dobrać dla gości dodatkowe usługi, które nie są uwzględnione w ofercie, na bazie której wydarzenie jest przeprowadzane. W tej sytuacji w celu uniknięcia późniejszych problemów dotyczących zwiększenia kwoty końcowej wydarzenia, wybierana jest osoba, która decyduje jakie elementy mogą być dokupione. Ilustrując tę sytuację można posłużyć się przykładem imprezy weselnej do której błędnie przewidziano liczbę dostępnych napoi. Osoba decyzyjna może tę wartość zwiększyć z pełną odpowiedzialnością zwiększenia kosztów. Zawartość pola tekstowego osoby decyzyjnej automatycznie wypełniana jest zawartością pola tekstowego *Imię i nazwisko*. Istnieje jednak możliwość, aby sprzedawca zmienił tą wartość na inną ponieważ nie zawsze osoba klienta jest osobą decyzyjną w czasie trwania wydarzenia.

- *TERMIN (od - do)*

  Pola te określają datę rozpoczęcia oraz datę zakończenia organizowanego wydarzenia. Data rozpoczęcia odgrywa istotną rolę w tworzonej propozycji przez fakt, iż ceny sal uzależnione są właśnie od miesiąca, w którym dane wydarzenie będzie organizowane. Po wyborze daty, w polu *Miesiąc* ukazuje się nazwa miesiąca na podstawie którego dobierana będzie cena wybranej w późniejszym etapie tworzenia propozycji cenowej sali.

- *CZAS TRWANIA (od - do)*

  Czas trwania definiuje planowane godziny, w czasie których przeprowadzone ma być wydarzenie. Bezpośrednio odwołują się one do wpisanej uprzednio daty rozpoczęcia oraz zakończenia. Na tej podstawie możliwe jest dokładne określenie okresu trwania wydarzenia.

- *ILOŚĆ OSÓB*

  Pole to określa planowaną liczbę osób, które mają w danym wydarzeniu uczestniczyć. W celu weryfikacji, czy dana liczba osób fizycznie zmieści się na wybranej sali, sprzedawca ma możliwość posłużyć się dostępną poniżej prezentowanych pól tabelą pomocniczą.

- *STATUS PROPOZYCJI*

Pole to jest polem informacyjnym, który opisuje status tworzonej propozycji. Domyślnym statusem podczas stworzenia nowej propozycji jest *"Nowa"*, którą sprzedawca w sytuacji oczekiwania na decyzję klienta ma obowiązek zmienić na *"W trakcie realizacji"*, natomiast po sfinalizowaniu transakcji oznaczyć jako *"Zrealizowana"*. Status propozycji cenowej wyświetlany jest na liście wszystkich propozycji w oknie głównym.

- *TABELA SAL*

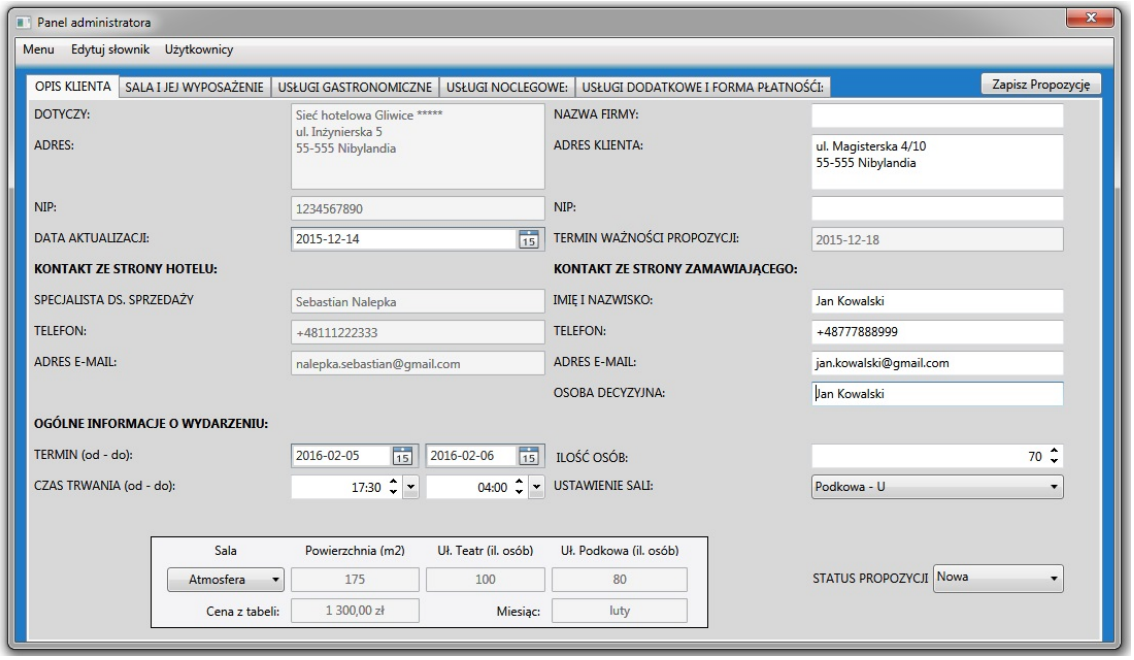
  W zależności od wybranej z listy rozwijanej sali oraz uprzednio wybranej dacie rozpoczęcia wydarzenia, sprzedawca ma możliwość otrzymania informacji o cenie sali, jej powierzchni oraz liczbie dostępnych miejsc. Informacje te w czasie ustalania szczegółów transakcji umożliwiają wybór optymalnego dla potrzeb klienta pomieszczenia, w którym przez wzgląd na liczbę dostępnych miejsc będzie można rozlokować wszystkich uczestników. Wybrana z listy rozwijanej sala wraz z datą rozpoczęcia jednoznacznie identyfikuje cenę oraz nazwę pomieszczenia, której rekord w sposób automatyczny dodawany jest jako pierwszy w kolejnej zakładce opisującej szczegóły rezerwacji.

7.2.2. Sala i jej wyposażenie

Druga zakładka modułu tworzenia nowej propozycji cenowej widoczna pod rysunkiem 22, dotyczy szczegółów rezerwacji dotyczących sali, jej dodatkowego wyposażenia oraz cen poszczególnych elementów. Na tym etapie negocjowany jest również rabat dotyczący finalnej ceny wybranej przez klienta sali.

- *WYBÓR WYPOSAŻENIA*

  Dodanie elementów wyposażenia dodatkowego sali umożliwiają rozwijane listy, które zawierają dostępne dla hotelu elementy, z których klient może skorzystać. Po jego wybraniu, sprzedawca zobowiązany jest na drodze negocjacji ustalić z klientem *cenę jednostkową brutto*, podać jego *ilość* oraz *liczbę dni*, przez które będzie używane. Tak wpisane informacje zostają przeliczane, a ich wynik wyświetlany zostaje w polu *wartość brutto*. *Ceny jednostkowe netto* oraz całkowite *wartości netto* są również obliczane automatycznie na podstawie podanych cen brutto oraz ustalonej również z poziomu rozwijanej listy stawki VAT.



Rysunek 21: Okno tworzenia propozycji cenowej. Zakładka kontakt ze strony zamawiającego, Źródło: Opracowanie własne.

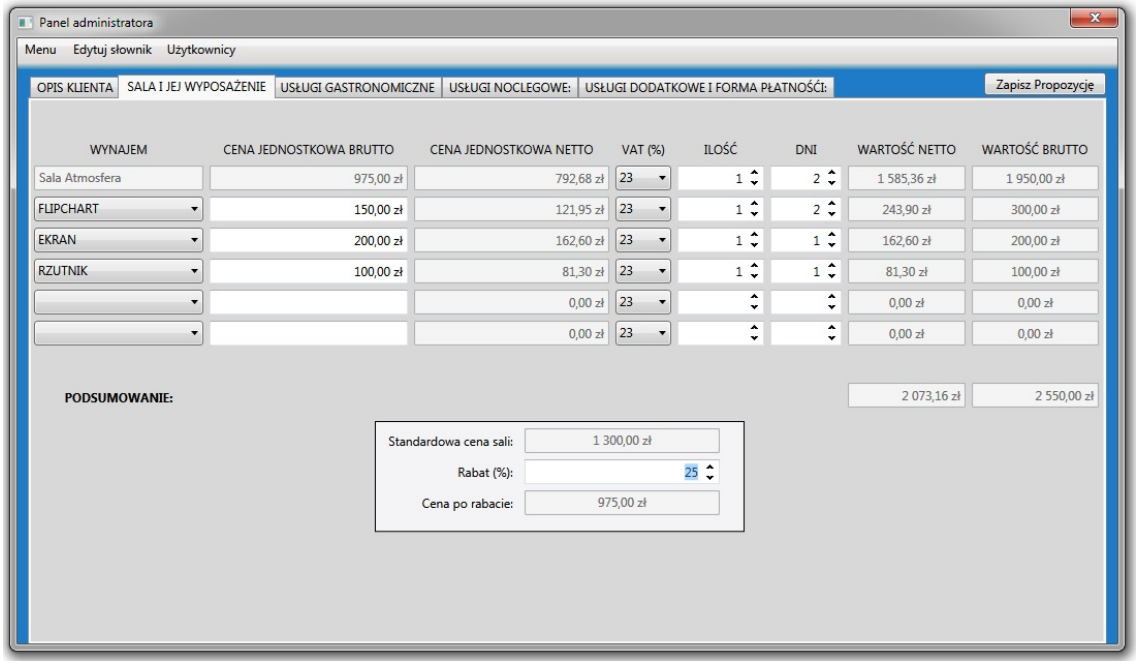
- *RABATY*

  Ceny tabelaryczne wynajmowanych sal są cenami umownymi, narzuconymi przez menadżera odgórnie. Sprzedawca ma jednak prawo ich obniżania na drodze negocjacji w celu przedstawienia klientowi korzystnej oferty na tle konkurencji. Po operacji wyboru sali, w polu *"Standardowa cena sali"* widoczna jest jej cena, którą za pomocą znajdującego się poniżej pola można redukować. W chwili zmiany wartości rabatu, wszelkie wartości cenowe jego dotyczące są w czasie rzeczywistym odświeżane, co ułatwia ich kontrolę przez sprzedawcę.

7.2.3. Usługi gastronomiczne

Kolejnym krokiem w czasie procesu tworzenia propozycji cenowej jest wybór usług związanych z gastronomią. Zasada działania usług gastronomicznych, których okno widoczne jest pod rysunkiem 23, podobna jest do poprzedniej zakładki, z tą różnicą iż wewnętrzne mechanizmy sprawdzające powiązania są znacznie bardziej skomplikowane.

- *WYBÓR ELEMENTÓW GASTRONOMICZNYCH*



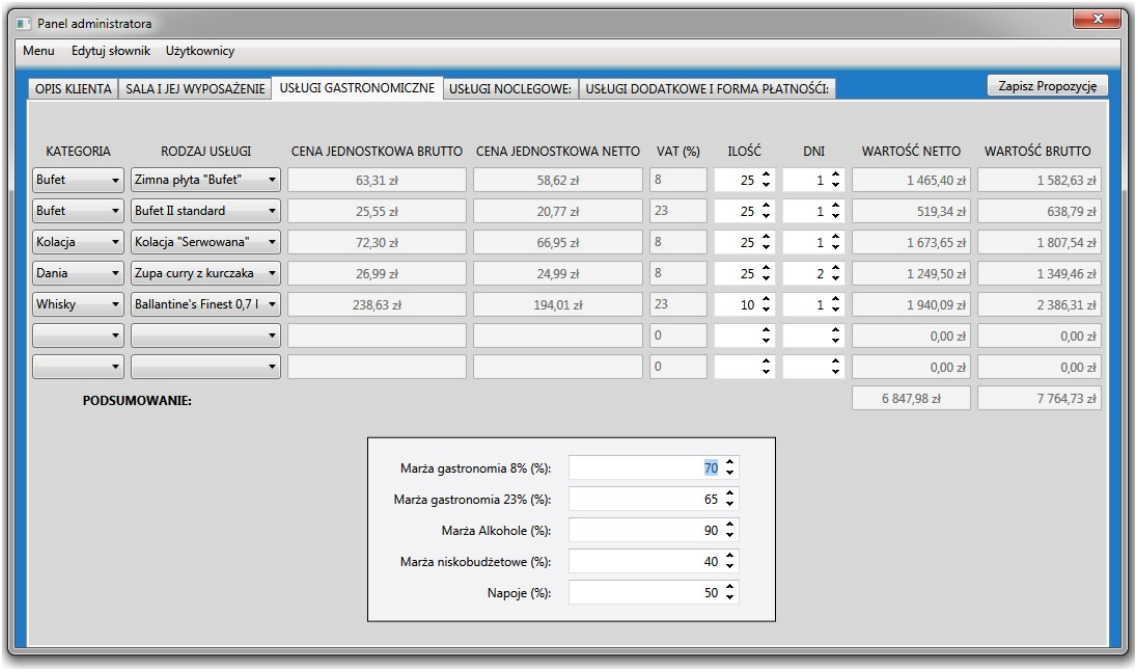
Rysunek 22: Okno tworzenia propozycji cenowej. Zakładka sali i jej wyposażenia, Źródło: Opracowanie własne.

Wszelkie elementy gastronomiczne przez wzgląd na ich ilość podzielone zostały na kategorie, które umożliwiają proste wyszukanie interesującego elementu. W celu dodania elementu gastronomicznego należy z pierwszej rozwijanej listy wybrać interesującą kategorię oraz z drugiej listy wybrać konkretny dla tej kategorii element. Po wykonaniu przedstawionych operacji, aktualne ceny jednostkowe wraz z podatkiem VAT zostają automatycznie uzupełniane, co usuwa ze sprzedawcy konieczność ich manualnego sprawdzania. Po wyborze ilości oraz liczby dni przez które produkt ma być dostępny, z poziomu tabeli wyświetlana zostaje jego zsumowana wartość.

- *MARŻE*

  Pobierane z bazy danych ceny produktów są cenami przyjętymi jako minimalne ceny po których produkt może zostać sprzedany. W celu wygenerowania większego przychodu sprzedawca ma możliwość podwyższenia ceny produktu poprzez ustalenie wielkości jego marży, których domyślne wartości ustalone zostały przez menadżera sprzedaży. Przedstawiane marże podzielone zostały na pięć kategorii przez wzgląd na ich specyfikę - dla przykładu marża na na-

poje alkoholowe jest z definicji znacznie większa niż na potrawy bufetowe lub produkty niskobudżetowe.



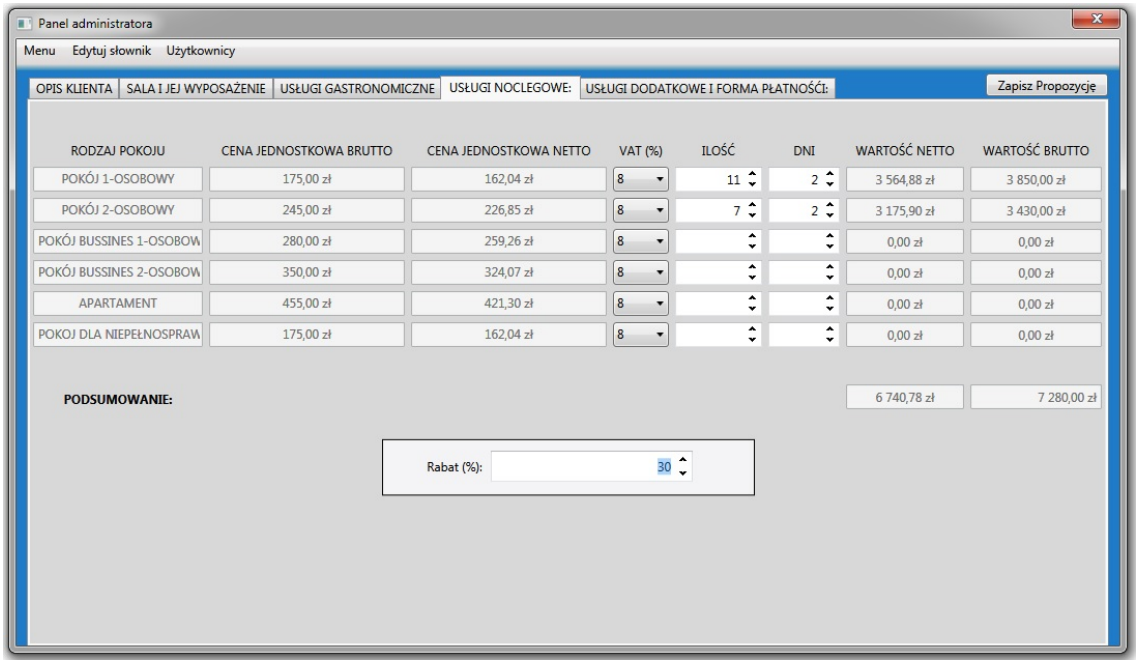
Rysunek 23: Okno tworzenia propozycji cenowej. Zakładka elementów gastronomicznych, Źródło: Opracowanie własne.

7.2.4. Usługi noclegowe

Zakładka usług noclegowych widoczna pod rysunkiem 24 opisuje liczbę oraz rodzaj wybranych pokoi hotelowych. Lista dostępnych pokoi wraz z wyszczególnionymi cenami automatycznie dostarczana jest do okna z bazy danych. Rolą sprzedawcy jest wybór odpowiedniej ich ilości oraz liczby dni. Jedyną możliwością obniżenia ustalonych przez menadżera cen jest ustalenie rabatu, który jednakowo obowiązuje wszystkie rodzaje pokoi. Po wpisaniu odpowiednich wartości, wewnętrzne mechanizmy programu przeliczają wszystkie wartości umożliwiając podgląd ich zsumowanych wartości netto i brutto.

7.2.5. Usługi dodatkowe i forma płatności

Ostatnią zakładką procesu tworzenia nowej propozycji cenowej widoczną pod rysunkiem 25 jest zakładka umożliwiająca ustalenie finalnej formy w której realizowana będzie zapłata, dodanie indywidualnie ustalanych z klientem elementów, które



Rysunek 24: Okno tworzenia propozycji cenowej. Zakładka dotycząca usług noclegowych, Źródło: Opracowanie własne.

nie są dostępne w zakresie standardowych usług oferowanych przez sieć hotelową oraz rezerwację miejsc parkingowych dla uczestników organizowanego wydarzenia. Przez wzgląd, iż przedstawiana zakładka jest ostatnią częścią tworzenia propozycji cenowej, zawiera także pola zawierające całkowity koszt netto oraz brutto wszystkich wybranych wcześniej usług.

- *USŁUGI DODATKOWE*

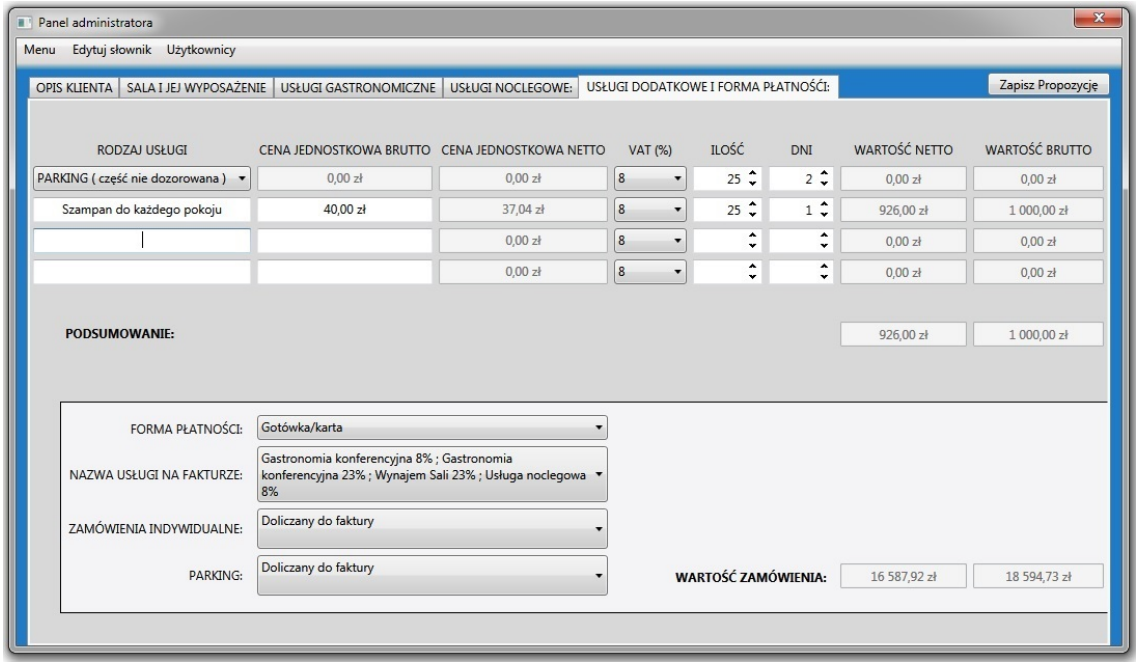
  W celu realizacji procesu dodania usługi niestandardowej, pracownik ma obowiązek uzyskania od przełożonego informacji dotyczącej fizycznej możliwości jej realizacji podczas organizowanego wydarzenia i w sytuacji akceptacji może zapisać jej nazwę w polu *Rodzaj usługi*, uzupełnić ustaloną cenę brutto wraz z wartością VAT oraz wybrać jej ilość oraz liczbę dni. Tak utworzony rekord usługi zostanie, analogicznie jak w przypadku poprzednich zakładek odpowiednio przeliczony i wyświetlony w polach dotyczących wartości.

  Rezerwacja parkingu jest elementem nieco prostszym. Obowiązkiem sprzedawcy jest ustalenie tylko i wyłącznie rodzaju wybranego parkingu, dostępnego z poziomu rozwijanej listy pierwszego wiersza okna oraz uzupełnienie jego

ilości oraz liczby dni.

- *FORMA ZAPŁATY*

  Podkategoria ta jest finalnym elementem tworzonej propozycji cenowej. Umożliwia ona sprzedawcy wybór formy w jakiej klient chce dokonać płatności, nazwy usługi zapisanej na fakturze oraz sposobu zapłaty za zamówienia indywidualne oraz parking. Opcje wszystkich przedstawionych elementów dostępne są z poziomu rozwijanych list, umożliwiając tym samym sprzedawcy bardzo szybką możliwość ich wyboru.



Rysunek 25: Okno tworzenia propozycji cenowej. Zakładka dotycząca usług dodatkowych oraz formę zapłaty za organizowane wydarzenie, Źródło: Opracowanie własne.

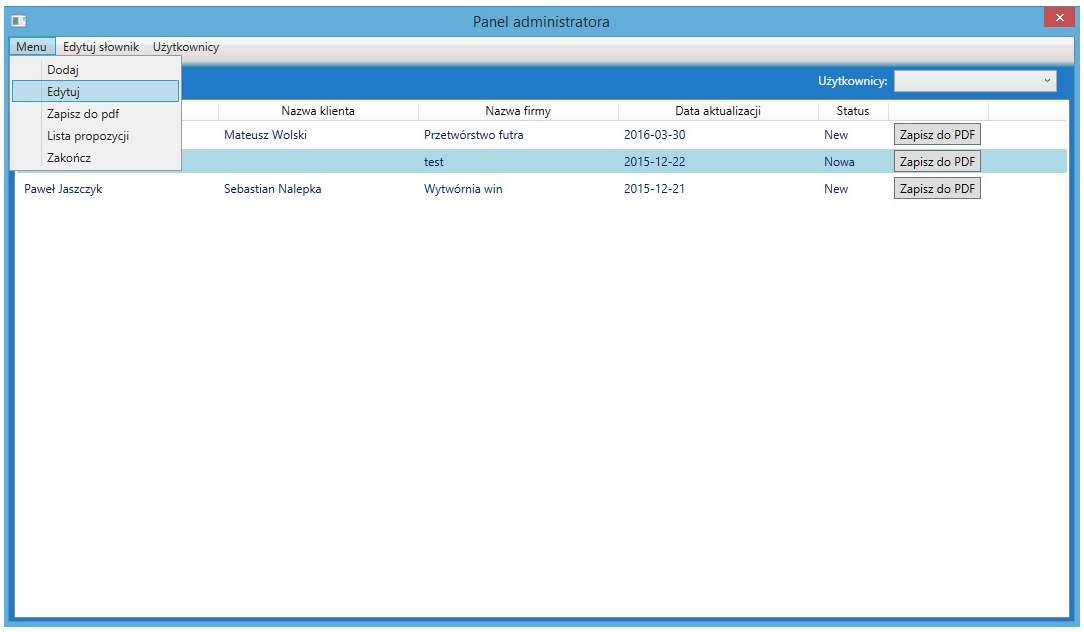
7.2.6. Zapis nowej propozycji cenowej

Operacja zapisu nowej propozycji cenowej możliwa jest do wykonania w każdej chwili tworzenia propozycji. Sprzedawca nie ma obowiązku tworzenia pełnego dokumentu, który w sposób natychmiastowy przekazywany będzie klientowi. Dostępna z poziomu menu opcja edycji istniejącej propozycji umożliwia jej finalne zakończenie w dowolnym czasie. W sytuacji chęci zapisu tworzonej propozycji, sprzedawca

zobowiązany jest nacisnąć przycisk *Zapisz propozycję*, który po odpowiedniej konwersji danych zapisze wszystkie wymagane do późniejszej edycji elementy do bazy danych. Lokalizacja przycisku umiejscowiona jest w prawym górnym rogu każdej zakładki tworzenia nowej propozycji cenowej. Jej wygląd dostępny jest do wglądu w przywołanym już rysunku 25.

7.3. Edycja istniejącej propozycji cenowej

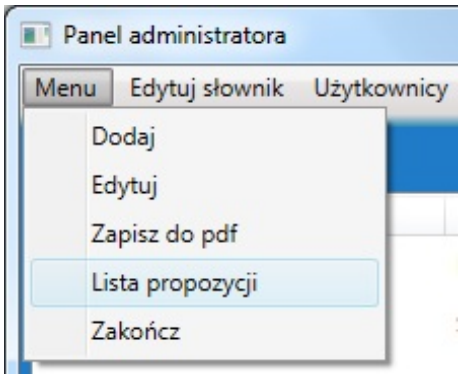
Edycja istniejącej propozycji cenowej odbywa się poprzez zaznaczenie w oknie główny programu wybranej propozycji cenowej, a następnie z menu wybrania "Edytuj" co można zaobserwować na rysunku 26. Użytkownik zostanie przeniesiony do okna tworzenia nowej propozycji, ale już z wypełnionymi wcześniej polami. Sprzedawca ma możliwość dodawania nowych produktów a usuwanie odbywa się poprzez wybranie pustego pola na liście rozwijalnej. Zapis edytowalnej propozycji jest możliwy w każdym momencie edycji.



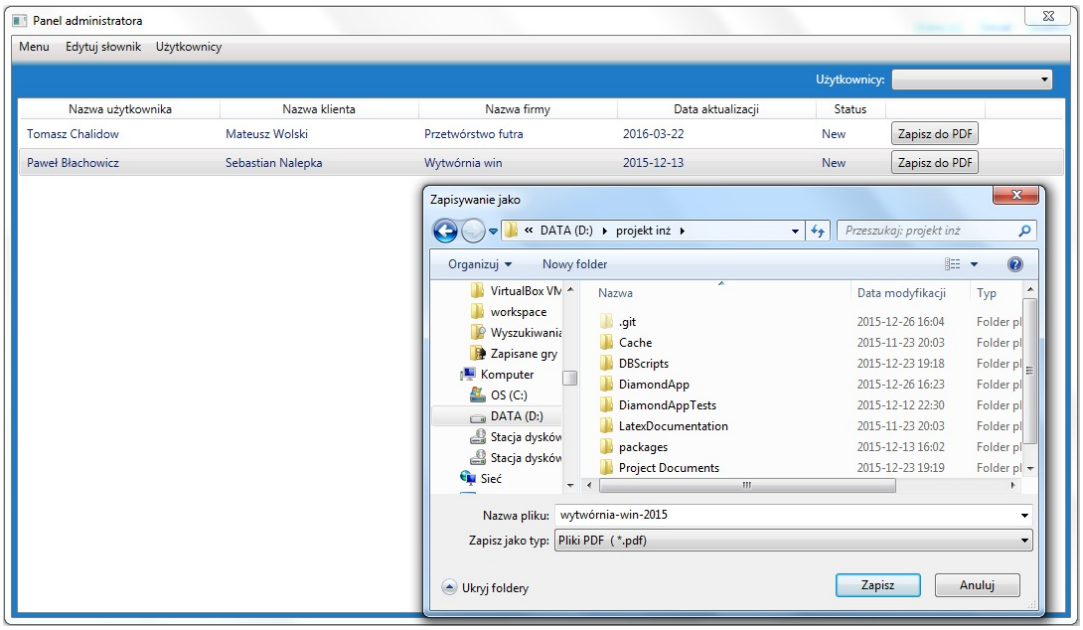
Rysunek 26: Edycja propozycji, Źródło: Opracowanie własne.

7.4. Tworzenie pliku PDF istniejącej propozycji cenowej

Jedną z najważniejszych funkcjonalności niniejszej aplikacji jest możliwość zapisu propozycji cenowych do pliku PDF. W menu rozwijanym znajduje się pozycja *Lista propozycji*, co zostało przedstawione na rysunku 27.

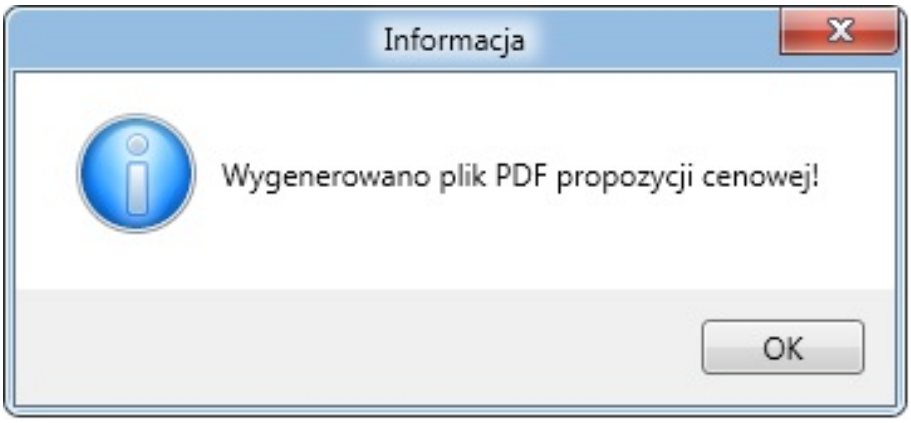


Rysunek 27: Wyświetlanie listy propozycji cenowych, Źródło: Opracowanie własne.



Rysunek 28: Zapis propozycji cenowej do pliku PDF, Źródło: Opracowanie własne.

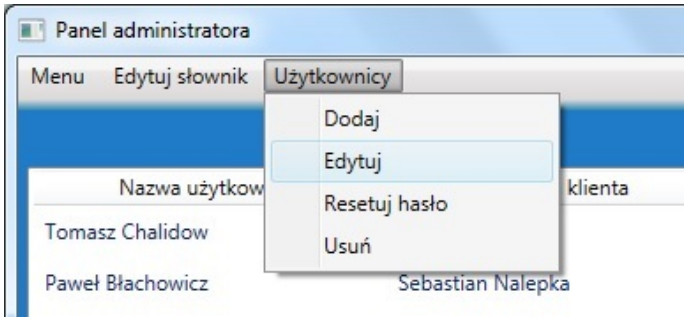
Po kliknięciu w opcję *Lista propozycji* zostaje wyświetlona lista istniejących propozycji cenowych. Oprócz podstawowych parametrów propozycji cenowych, w ostatniej kolumnie listy pojawia się przycisk *Zapisz do PDF* którego kliknięcie otwiera dialog wyboru pliku. Użytkownik wpisuje dowolną nazwę pliku, oraz wybiera lokalizację zapisu pliku co zostało pokazane na rysunku 28. Jeżeli proces generowania pliku PDF przebiegnie pomyślnie, zostaje wyświetlony komunikat z informacją o sukcesie, przedstawiony na rysunku 29. (dodać odnośnik do załącznika - wygenerowanego przez program PDF).



Rysunek 29: Poprawny zapis propozycji cenowej do pliku PDF, Źródło: Opracowanie własne.

7.5. Modyfikacja kont użytkowników/sprzedawców

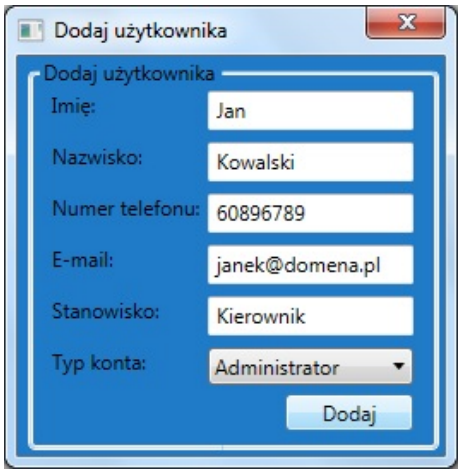
Użytkownicy z uprawnieniami administratora posiadają możliwość wykonywania podstawowych operacji związanych z zarządzaniem kontami pracowników. Rysunek 30 przedstawia menu *Użytkownicy* w którym znajdują się dostępne w tym module funkcjonalności. Po kliknięciu wybranej pozycji z menu, zostaje otwarte nowe okno w którym administrator może wykonać operacje danego typu.



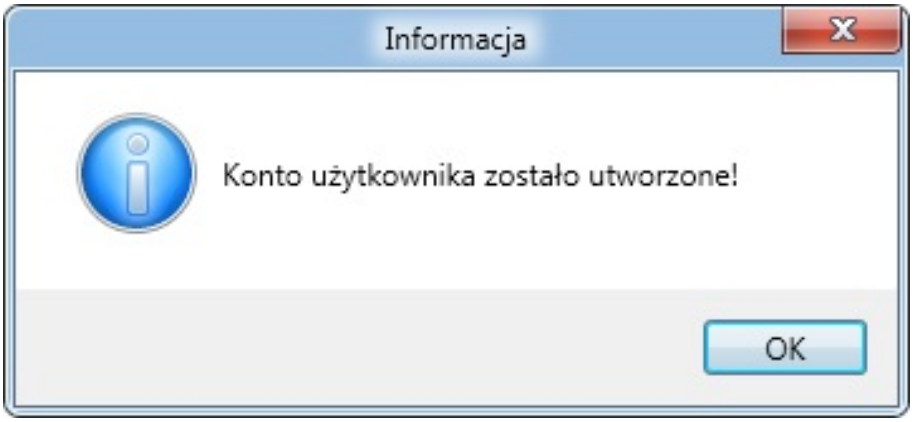
Rysunek 30: Menu rozwijane modułu służącego do zarządzania kontami pracowników, Źródło: Opracowanie własne.

7.5.1. Dodawanie konta

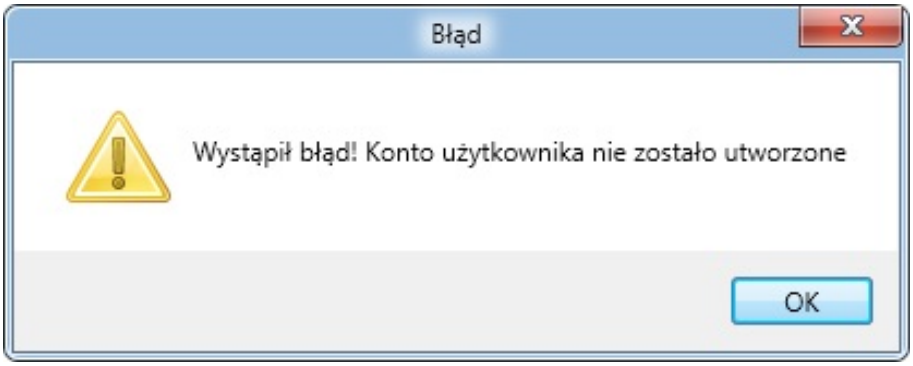
Pierwszą z funkcjonalności zaimplementowanych w module, jest możliwość utworzenia nowego konta. Rysunek 31 przedstawia okno zawierające pola tekstowe do których wpisywane są dane osobowe pracownika, oraz listę rozwijaną z której wybierany jest typ konta. Po kliknięciu przycisku *Dodaj* w przypadku poprawnego wykonania operacji dodawania konta, wyświetlana jest informacja przedstawiona na rysunku 32. W sytuacji niepowodzenia, zostaje wyświetlany komunikat o błędzie co można zaobserwować na rysunku 33.



Rysunek 31: Dodawanie nowego konta użytkownika, Źródło: Opracowanie własne.



Rysunek 32: Komunikat o poprawnym zapisie konta użytkownika w bazie danych, Źródło: Opracowanie własne.



Rysunek 33: Komunikat o błędzie podczas zapisu konta użytkownika do bazy danych, Źródło: Opracowanie własne.

7.5.2. Edycja konta

Kolejną funkcjonalnością jest możliwość edycji danych pracowników. Informacje o kontach wyświetlane są w formie tabeli. Edycja polega na zaznaczeniu wybranej komórki w tabeli i wpisaniu nowej wartości. Zapis nowej wartości do bazy danych potwierdzany jest kliknięciem przycisku *Enter*, bądź zaznaczeniem innej kolumny. Rysunek 34 zawiera wygenerowaną dla potrzeb projektu, listę użytkowników. Zbieżność osób i nazwisk jest przypadkowa.

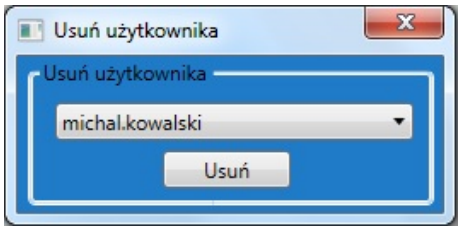
Edytuj użytkownika

Id	Imię	Nazwisko	Nr Telefonu	E-mail	Stanowisko	Typ konta	Login
1	Lidia	Kowalska	500400876	kowalska@domena.pl	Regionalny Kierownik Sprzedaży	Sprzedawca	liko
2	Agata	Kulesza	501501456	kulesza@domena.pl	Specjalista ds. sprzedaży	Sprzedawca	agku
3	Daria	Maksymiuk	546345995	maksymiuk@domena.pl	Specjalista ds. sprzedaży	Sprzedawca	dadr
4	Paweł	Błachowicz	500400234	blachowicz@domena.pl	Specjalista ds. sprzedaży	Sprzedawca	pabl
5	Michał	Wojciechowski	888456789	wojciech@domena.pl	Specjalista ds. sprzedaży	Administrator	miwi
6	Tomasz	Chalidow	678987677	chalidow@domena.pl	Specjalista ds. sprzedaży	Sprzedawca	toan
7	Małgorzata	Morawiec	555987677	morawiec@domena.pl	Contracting Manager	Sprzedawca	mami
8	Tomasz	Pruski	567897677	pruski@domena.pl	Specjalista ds. Sprzedaży	Sprzedawca	tosi
9	Anna	Jędrzejczak	555555555	jedrzejczak@domena.pl	Specjalista ds. Sprzedaży	Sprzedawca	anro
10	Maria	Borecka	505456777	borecka@domena.pl	Specjalista ds. Sprzedaży	Sprzedawca	mabo
11	Sylwia	Rusiecka	504101955	rusiecka@domena.pl	Specjalista ds. Klienta wypoczynkowego	Sprzedawca	sysl
12	Joanna	Majchrzak	603842838	majchrzak@domena.pl	Specjalista ds. Sprzedaży MICE	Sprzedawca	jogw
13	Monika	Morka	502973368	morka@domena.pl	Specjalista ds. Sprzedaży	Sprzedawca	mome
15	Daria	Krochmal	554987877	krochmal@domena.pl	Specjalista ds. sprzedaży	Sprzedawca	dadr
16	Justyna	Kaczmarek	578898777	kaczmarek@domena.pl	Specjalista ds. sprzedaży	Sprzedawca	juma
17	Magdalena	Puchalik	523550757	puchalik@domena.pl	Specjalista ds. sprzedaży	Sprzedawca	maje
18	Magdalena	Radtke-Kowalska	506101894	radtke@domena.pl	Specjalista ds. Sprzedaży	Sprzedawca	mara
19	Magdalena	Kozak	512971429	kozak@domena.pl	Specjalista ds. Sprzedaży	Sprzedawca	ma
20	Jan	Kowalski	500500500	jan@kowalski.pl	Sprzedawca	Sprzedawca	jan.kowalski
21	Michał	Kowalski	667895098	michal@domena.pl	Programista	Administrator	michal.kowalski
23	Jan	Kowalski	60896789	janek@domena.pl	Kierownik	Administrator	jan.kowalski

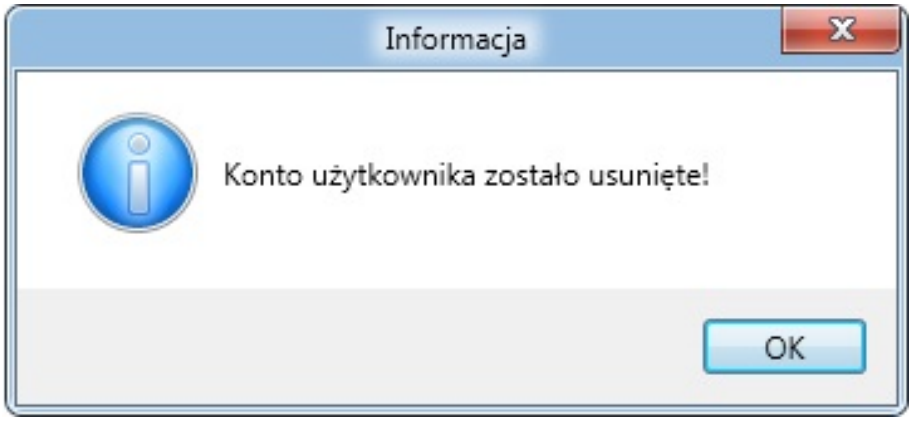
Rysunek 34: Tabela zawierająca dane użytkowników, Źródło: Opracowanie własne.

7.5.3. Usuwanie konta

W przypadku zmian kadrowych, administrator posiada możliwość usuwania kont użytkowników. Rysunek 35 przedstawia okno obsługujące funkcjonalność usuwania, które posiada listę rozwijaną zawierająca loginy użytkowników. Po wybraniu loginu użytkownika, którego konto ma zostać usunięte należy kliknąć przycisk *Usuń*. Pomyślne usunięcia konta z bazy danych jest potwierdzane komunikatem wyświetlanym na rysunku 36.



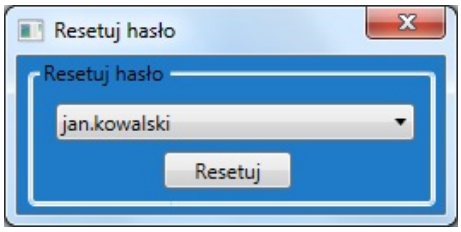
Rysunek 35: Okno usuwania kont użytkowników, Źródło: Opracowanie własne.



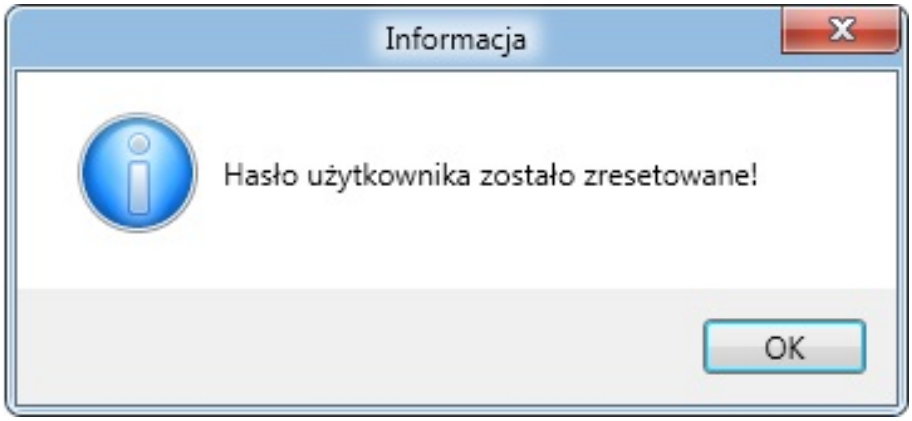
Rysunek 36: Komunikat o poprawnym usunięciu konta użytkownika z bazy danych, Źródło: Opracowanie własne.

7.5.4. Resetowanie haseł

Istnieje możliwość, iż użytkownik nie pamięta swojego hasła. W takim przypadku administrator korzysta z opcji resetowania hasła, co umożliwia użytkownikowi ustawienie nowego hasła podczas kolejnej próby logowania. Na rysunku 37 przedstawiono okno w którym znajduje się lista rozwijana, wypełniona loginami użytkowników tak samo jak w module służącym do usuwania kont. Po wybraniu loginu z listy należy kliknąć przycisk *Resetuj*. Komunikat o pomyślnym usunięciu konta z bazy danych znajduje się na rysunku 38.



Rysunek 37: Okno resetowania haseł użytkowników, Źródło: Opracowanie własne.



Rysunek 38: Komunikat o powodzeniu operacji usuwania konta użytkownika z bazy danych, Źródło: Opracowanie własne.

7.6. Modyfikacja słowników cenowych

Modyfikacja słowników cenowych odbywać się może tylko z poziomu administratora. Z menu należy wybrać „Edytuj słownik". Zostanie otwarte nowe okno, gdzie będzie można wybrać typ słownika, który będzie edytowany. Niestety nie wszystkie słoniki są edytowalne wynika to z faktu, braku cen produktów które są ustawiane przez sprzedawce. Pierwsze klikniecie w komórkę powoduje jej zaznaczenie, drugie wybranie, dopiero po trzecim kliknięciu jest możliwa edycja komórki. Edycja następuje poprzez wpisywanie wartości w poszczególnych komórkach tabeli w zależności, którą pozycję chce się edytować. Wciskając przycisk „ENTER" następuje zapis edytowanego wiersza do bazy danych. Istnieje również możliwość dodawania nowych pozycji. W ostatnim pustym wierszu tabeli należy wypełnić wszystkie komórki, by móc dodać do bazy. W trakcie edycja jak i dodawania istnieje kontrola typów danych jeśli w komórce ma być liczba a będziemy chcieli wpisać tekst komórka podświetli się na czerwono. Odświeżenie tabeli w przypadku dodania nowego produktu następuje po ponownym uruchomieniu okna edycji słowników.

- *Pokoje*

  Istnieje tylko możliwość edytowania ceny pokoi

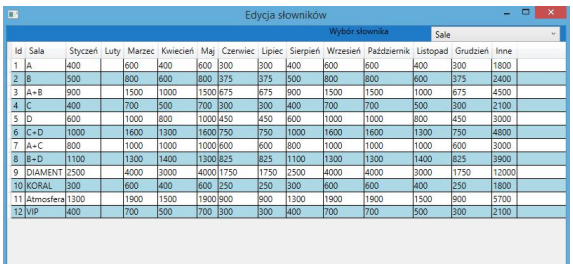
- *Gastronomia*

  W słowniku tym istnieje możliwość dodawania nowych pozycji. Ułatwieniem w przypadku tego słownika jest podział ze względu na kategorie produk-

tów. Tego typu udogodnienie ułatwia znajdowanie poszczególnych pozycji w tym słowniku, przyspiesza jego edycje.

- *Sale*

  Ceny poszczególnych sal są zależne on miesięcy, w których dane wydarzenie ma się odbyć. Nie ma możliwości dodawania nowych sal. Przedstawia to rysunku 39



Id	Sala	Styczeń	Luty	Marzec	Kwiecień	Maj	Czerwiec	Lipiec	Sierpień	Wrzesień	Październik	Listopad	Grudzień	Inne
1	A	400		600	400	600	300	300	400	600	600	400	300	1800
2	B	500		800	600	800	375	375	500	800	800	600	375	2400
3	A+B	900		1500	1000	1500	675	675	900	1500	1500	1000	675	4500
4	C	400		700	500	700	300	300	400	700	700	500	300	2100
5	D	600		1000	800	1000	450	450	600	1000	1000	800	450	3000
6	C+D	1000		1600	1300	1600	750	750	1000	1600	1600	1300	750	4800
7	A+C	800		1000	1000	1000	600	600	800	1000	1000	1000	600	3000
8	B+D	1100		1300	1400	1300	825	825	1100	1300	1300	1400	825	3900
9	DIAMENT	2500		4000	3000	4000	1750	1750	2500	4000	4000	3000	1750	12000
10	KORAL	300		600	400	600	250	250	300	600	600	400	250	1800
11	Atmosfera	1300		1900	1500	1900	900	900	1300	1900	1900	1500	900	5700
12	VIP	400		700	500	700	300	300	400	700	700	500	300	2100

Rysunek 39: Ceny sal w poszczególnych miesiącach, Źródło: Opracowanie własne.

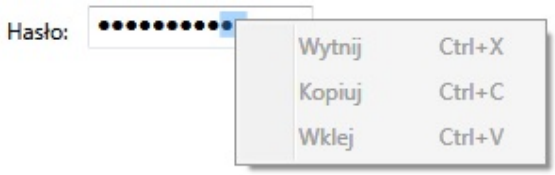
8. Specyfikacja wewnętrzna

8.1. Przeprowadzenie operacji logowania użytkownika

8.1.1. Bezpieczeństwo procesu logowania

Moduł logowania jest częścią systemu, do którego dostęp ma każdy użytkownik korzystający z prezentowanej aplikacji. Implikuje to konieczność jego zabezpieczenia przed niepowołanym użyciem dotyczącym dostępu do poufnych danych, takich jak dane osobiste oraz hasła, dla których dobrą praktyką odnoszącą się do bezpieczeństwa jest ich nieprzechowywanie w żadnym miejscu w postaci jawnej.

Pole wpisywania hasła przez użytkownika w oknie logowania stanowi dedykowana dla tego celu kontrolka PasswordBox, dostępna do wglądu na rysunku 40. Jest to zmodyfikowana kontrolka TextBox dostępna w technologi WPF, która wyświetla swoją zawartość w postaci ukrytej, uniemożliwiając jednocześnie wykonanie na jej zawartości operacji kopiowania oraz wycinania z poziomu klawiatury oraz myszy. Zabezpiecza to przed niechcianym podejrzeniem hasła oraz jego odzyskaniem przez osobę trzecią.



Rysunek 40: Kontrolka PasswordBox z przedstawieniem aspektów bezpieczeństwa, Źródło: Opracowanie własne.

Sama operacja sprawdzania poprawności wpisanego hasła została zaprojektowana w taki sposób, aby otrzymane z kontrolki hasło nie było nigdzie zapisywane i porównywane w postaci jawnej. W chwili wpisania hasła przez użytkownika oraz kliknięcia przycisku logowania, następuje sprawdzenie ustalonych zasad dotyczących złożoności hasła opisanych w rozdziale 8.1.2 oraz w sytuacji ich pozytywnego rozpatrzenia, hasło przekształcane jest za pomocą 256-bitowej odmiany algorytmu

haszującego SHA-2 na skrót, który porównywany jest ze skrótem hasła dopasowanemu wpisanej nazwie użytkownika użytkownika w bazie danych. Na tej podstawie przyznawany lub odmawiany jest dostęp do dalszej części programu.

8.1.2. Wymagania złożoności haseł

Ustalenie hasła przez użytkownika programu implikuje obowiązek walidacji wprowadzonego przez niego ciągu znaków w celu sprawdzenia jego poziomu bezpieczeństwa. Krótkie hasła są bardziej podatne próbę ich złamania, niż dłuższe ciągi, składające się zarówno z liter jak i cyfr. W tym celu, w czasie ustalania przez użytkownika nowego hasła, wprowadzona została metoda widoczna przy kodzie źródłowym 3, której zadaniem jest weryfikacja założonych z góry aspektów, które musi hasło spełniać w celu jego akceptacji. Wymogami tymi jest długość hasła, które składać się musi z przynajmniej ośmiu znaków, posiadanie minimum jednej dużej oraz małej litery, a także obecność w jego składzie cyfry. W przypadku niespełnienia chociaż jednej z przedstawionych zasad, hasło jest automatycznie odrzucane, a użytkownik informowany jest o zaistniałym problemie.

```
public static bool ValidatePassword(string pass)
{
    // minimalna wymagana długość hasła
    const int minLength = 8;
    // sprawdzenie czy hasło jest puste
    if (pass == null)
        return false;
    // sprawdzenie poprawności długości hasła
    bool isPassLengthOk = pass.Length >= minLength;
    bool hasUpperCaseChar = false;
    bool hasLowerCaseChar = false;
    bool hasDecimalDigit = false;
    if (isPassLengthOk)
    {
        // sprawdzanie czy hasło zawiera dużą literę, małą
           literę oraz cyfrę
        foreach (char letter in pass)
        {
```

```
            if (char.IsUpper(letter))
                hasUpperCaseChar = true;
            else if (char.IsLower(letter))
                hasLowerCaseChar = true;
            else if (char.IsDigit(letter))
                hasDecimalDigit = true;
        }
    }
    bool isValid = (isPassLengthOk && hasUpperCaseChar
                    && hasLowerCaseChar && hasDecimalDigit);
    // true == hasło poprawne, false == hasło niepoprawne
    return isValid;
}
```

Kod źródłowy 3: Metoda weryfikująca poprawność wpisanego przez użytkownika hasła

8.2. Konto administratora i użytkownika – ListView

Użytkownik może oglądać swoje propozycje ich ilość jak i z jak firmami związana jest dana propozycja cenowa. Do tego celu nie można w bezpośredni sposób wykorzystać bazy danych lecz została stworzona warstwa pośrednia w postaci klasy AdminProposition. Klasa ta zawiera pola wymagane do wyświetlenia propozycji cenowej w postaci imię nazwisko osoby odpowiedzialnej, status, jeśli dana propozycja cenowa jest dla firmy to firmę. Wykorzystując polecenia LINQ są pobierane dane z bazy danych i wykorzystując wbudowane mechanizmy, jest tworzona lista propozycji z wymaganymi polami. Przedstawia to kod źródłowy 20.

```
var myProposition = (from prop in _ctx.Proposition
                     from user in _ctx.Users
                     where user.Id == _userId
                     where prop.Id_user == _userId
            // szuka propozycji użytkownika zalogowanego
            select new AdminProposition
            {
             PropositionId = prop.Id,
```

```
                UserFirstName = user.Name,
                UserSurname = user.Surname,
                CustomerFullName = prop.PropClient
                                        .CustomerFullName,
                CompanyName = prop.PropClient.CompanyName,
                UpdateDate = prop.UpdateDate,
                Status = prop.Status
               }).ToList();
               PropositionsList = myProposition;
//przypisanie do listy propozycji
//w prawidłowej formie do wyświetlenia
```

Kod źródłowy 4: Polecenie wyciągające wszystkie propozycje użytkownika

Administrator ma również dodatkową możliwość wybierania propozycji danego użytkownika. Wykorzystując możliwości języka LINQ tworzona jest lista wszystkich użytkowników mających jakiekolwiek propozycje cenowe i przypisanie ich do ComboBox. Proces przypisywania prezentuje kod źródłowy 9 Wraz z wyborem użytkownika z tej listy następuje odświeżenie ListView z propozycjami.

```
var userlist = (from s in _ctx.Users
                    where s.Proposition.Any()
                    select s).ToList();
           userlist.Add(new Users());
//dodanie pustego użytkownika by
//móc wyświetlić wszystkie propozycje
           UsersList = userlist;
```

Kod źródłowy 5: Polecenie wyciągające wszystkie propozycje użytkownika

Edycja propozycji w obydwu przypadkach odbywa się w ten sam sposób poprzez wybranie propozycji z ListView następuje przypisanie identyfikatora z bazy danych do zmiennej i wyszukanie wszystkich elementów danej propozycji. Z kolei wypełniane są wszystkie pola, które zostały zaznaczone.

8.3. Tworzenie propozycji cenowej

Moduł tworzenia propozycji cenowej wraz z modułem jej edycji jest najbardziej rozbudowanym modułem prezentowanej aplikacji, który przez fakt aktywnej

komunikacji z bazą danych w celu pobrania potrzebnych słownikowych oraz zapisu wybranych przez użytkownika elementów posiada skomplikowaną logikę działania. Najważniejszym elementem koniecznym do prawidłowej pracy prezentowanego modułu była kontrola przesyłanych pomiędzy bazą a aplikacją danych. Pola obiektów, które zawierały puste wartości wymagały szczegółowej kontroli i konwersji typu NULL na typ obsługiwany przez odbiorcę. Wymóg ten był konieczny przez wzgląd na możliwość wystąpienia wyjątku, który przerywał pracę całej aplikacji. Zgodnie z założeniami projektowymi, użytkownik w czasie tworzenia propozycji cenowej ma możliwość wypełniania tylko jej części. Dodając do tego fakt, iż użytkownik przypadkowo może wypełnić dla przykładu rekord zawierający element gastronomiczny bez wpisania jego dni lub ilości, szczegółowa kontrola zawartości obiektów była priorytetowa. W sytuacji nie wpisania wszystkich kolumn rekordu odnoszącego się do elementu zakładki opisywanej w punktach 7.2.2, 7.2.3, 7.2.4 oraz 7.2.5, przy zapisie propozycji do bazy danych, rekord ten jest wychwytywany przez mechanizmy walidujące i usuwany.

8.4. Edycja słowników oraz kont użytkowników

8.4.1. DataGridView a dynamiczna komunikacja z Bazą danych

Aby umożliwić edycję danych z poziomu DataGridView, zaimplementowano funckję obsługującą zdarzenie *DataGridRowEditEndingEventArgs*. Kiedy edycja danej komórki zostaje zakończona, wywoływana jest funkcja której zadaniem jest dynamiczne przechwytywanie nowych wartości oraz aktualizacja bazy danych. Użycie zmiennej dynamic, podyktowane jest faktem, iż jest ona sprawdzana podczas wywołania, a nie kompilacji [60]. Umożliwia to odwołanie do nowych wartości zapisanych w komórkach. Funkcja jest wykorzystywana w module służącym do edycji danych pracowników. Implementację funkcji przedstawia kod źródłowy 6:

```
//pobranie identyfikatora użytkownika którego wiersz
//jest zaznaczony w DataGridView oraz zapis do zmiennej typu
   int
object item = UserList.SelectedItem;
string ID =
    (UserList.SelectedCells[0]
             .Column
```

```
            .GetCellContent(item) as TextBlock).Text;

int selected = Int32.Parse(ID);

//zapis danych z poszczególnych kolumn do zmiennej
   dynamicznej
dynamic userRow = UserList.SelectedItem;

//konwersja rodzaju konta z liczby na tekst
int tmp = 0;
if (userRow.UserAccountType == "Administrator")
        tmp = 1;
else
    tmp = 2;

//deklaracja obiektu bazy danych
DiamondDBEntities _ctx = new DiamondDBEntities();
Users userUpdate = (from user in _ctx.Users
                    where user.Id == selected
                    select user).First();

        //przypisanie nowych wartości z komórek do
           poszczególnych
        //kolumn tabeli Users
        userUpdate.Name = userRow.UserName;
        userUpdate.Surname = userRow.UserSurname;
        userUpdate.PhoneNum = userRow.UserPhoneNumber;
        userUpdate.Email = userRow.UserEmail;
        userUpdate.Position = userRow.UserPosition;
        userUpdate.AccountType = tmp;
        userUpdate.Login = userRow.UserLogin;

        //zapis danych do bazy
```

```
        _ctx.SaveChanges();
```

Kod źródłowy 6: Implementacja i opis funkcji aktualizującej bazę danych z poziomu DataGridView

8.5. Tworzenie szablonu PDF propozycji cenowej

Za pomocą mechanizmów biblioteki MigraDoc, zaimplementowano funckje tworzące tabele propozycji cenowej. Kod źródłowy 7 przedstawia prostą funkcję tworząca tabelę z jedną kolumną i wierszem.

```
//inicjalizacja nowej tabeli, ustawienie czcionki i jej
   rozmiaru
Table table = document.LastSection.AddTable();
table.Borders.Visible = true;
table.Format.Font.Size = 10;
table.Format.Font.Name = "Calibri";

//dodawanie nowej kolumny
Column column = table.AddColumn("3.750cm");
column.Format.Alignment = ParagraphAlignment.Left;

//dodawanie nowego wiersza, z przykładowymi atrybutami
Row row = table.AddRow();
row.Cells[0].Shading.Color = Colors.LightGray;
row.Cells[0].AddParagraph("WYNAJEM");
\end{verbatim}

Kod źródłowy \ref{listing:SavePdf} przedstawia implementację
   funkcji zapisującej wygenerowane przez program tabele
   propozycji cenowej do pliku PDF.

\begin{lstlisting}[caption= {Implementacja i opis funkcji
   zapisującej propozycję cenową do pliku PDF},
   label={listing:SavePdf}]
// deklaracja obiektu klasy PdfDocumentRenderer
PdfDocumentRenderer renderDocument =
```

```
new
   PdfDocumentRenderer(true,PdfSharp.Pdf.PdfFontEmbedding.Always);
//renderowanie dokumentu
renderDocument.Document = document;
renderDocument.RenderDocument();
//zapis do pliku
renderDocument.Save(path);
```

Kod źródłowy 7: Implementacja i opis funkcji tworzącej prostą tabelę przy użyciu biblioteki PDFSharp & MigraDoc

9. Podsumowanie i perspektywy rozwoju oprogramowania

Celem prezentowanego projektu było wykonanie aplikacji bazodanowej wspierającej pracę wybranej sieci hotelowej, która umożliwia zautomatyzowanie procesu tworzenia propozycji cenowej, uproszczenie sposobu aktualizacji wszelkich cenników usług dostępnych w hotelu oraz możliwość kontroli pracowników przez kierownictwo. Przedstawione cele zostały w pełni zrealizowane, czego efektem jest działająca aplikacja bazodanowa posiadająca szereg modułów umożliwiających ich osiągnięcie.

Dzięki zastosowaniu środowiska bazodanowego, pracownicy mają pewność, iż dane dotyczące cen usług na których pracują są danymi aktualnymi, które aktualizowane są na bieżąco z poziomu konta menadżera sprzedaży. Aspekt ten usprawnienia proces aktualizacji danych, niwelując potrzebę korzystania ze skrzynek elektronicznych. Menadżer sprzedaży w sytuacji aktualizacji danej ceny ma obowiązek edycji odpowiedniej tabeli z poziomu swojego uprzywilejowanego konta, a połączenie aplikacji z bazą danych automatycznie zapewni wprowadzenie zmiany na kontach wszystkich pracowników. Baza danych oraz skategoryzowanie wszystkich produktów gastronomicznych, zwalnia natomiast z użytkownika konieczność manualnego wyszukiwania określonego elementu w czasie tworzenia propozycji cenowej. Wprowadzona funkcja kategorii, umożliwia wybór określonego typu produktu oraz kolejno wybranie go z listy która pozbawiona jest wszystkich produktów, które do wybranej kategorii nie zostały przypisane. Korzyścią płynącą z przedstawionej funkcji jest oszczędność czasu pracownika, która umożliwia przyspieszenie procesu tworzenia nowej propozycji cenowej lub edycji już istniejącej. Każda propozycja cenowa, dzięki przypisaniu do osoby, która ją tworzyła możliwa jest również do wglądu z poziomu uprzywilejowanego konta. Dzięki temu kierownictwo ma możliwość przeglądania przygotowywanych przez pracowników ofert, a co za tym idzie wzgląd we wszelkie ustalone z klientami ceny oraz zniżki.

W fazie implementacji założonych modułów prezentowanej aplikacji napotkano na wiele problemów związanych ze zmianą wymagań projektowych i sposobem realizacji ustalonych funkcjonalności. Wraz z biegiem czasu powstawały nowe pomysły

dotyczące usprawnienia istniejących rozwiązań, które niejednokrotnie zwiększyłyby ich wydajność oraz efektywność. Rozwiązania te nie zawsze mogły być jednak wprowadzone przez pryzmat czasu którego to wymagały w celu ich implementacji, przetestowania oraz połączenia z istniejącym już systemem. Proponowane zmiany wiązały się także z modyfikacją istniejącej architektury systemu, co w znaczący sposób zmieniałoby zaplanowany wcześniej harmonogram prac i wydłużałoby finalne ukończenie projektu.

Pierwszym pomysłem rozbudowy prezentowanego systemu jest wprowadzenie funkcji tworzenia propozycji offline. Zgodnie z ustalonymi wymaganiami projektowymi, osoba korzystająca z aplikacji zobowiązana jest do posiadania dostępu do sieci Internet. W przypadku jego braku, praca z programem przez wzgląd na konieczność weryfikacji tożsamości na podstawie bazy danych jest niemożliwa. Wersja offline wprowadziłaby możliwość stworzenia nowej propozycji cenowej w sytuacji braku połączenia internetowego oraz zapis wszystkich jego elementów w postaci pliku na lokalnym dysku twardym. Implikuje to jednak szereg problemów do rozwiązania na tle synchronizacji z bazą danych oraz szablonu tworzonej propozycji. W sytuacji utworzenia nowej propozycji offline oraz późniejszym uzyskaniu dostępu do sieci, wszystkie jej elementy synchronizowane być powinny z bazą danych oraz lokalnie usuwane. Komplikacja na tle schematu tworzenia propozycji dotyczyłaby z kolei sposobu wypełnienia pól, których zawartość zależna jest od bazy danych.

Następną modyfikacją możliwą do wprowadzenia w przypadku rozwoju projektu jest modyfikacja obecnie zaimplementowanego szablonu wyboru elementów dostępnych usług. W chwili obecnej liczba możliwych do wyboru elementów każdej usługi jest z góry ustalona. Modyfikacja wprowadzałaby dynamicznie rozwijane rekordy, których liczba zależna byłaby od liczby wybranych elementów. Zmiana ta głęboko ingerowałaby w obecną architekturę modułu tworzenia propozycji i wymagałaby jej gruntownej przebudowy dotyczącej zmiany stosowanych kontrolek graficznych, implementacji ich zachowań oraz walidacji wprowadzanych danych.

10. Tytuł drugiego rozdziału. Bardzo długi tytuł. Jego formatowanie jest trudniejsze

CACHE

Tu jest wnętrze rozdziału drugiego.

Dodatek

Mój specjalny dodatek

CACHE

Tu treść dodatku. Zwróćmy uwagę na sposób numerowania dodatku, możliwa jest zmiana numerowania, patrz wyjaśnienia.

Załączniki

CACHE zalaczniki

```
<?php
␣␣␣echo␣"test=$test";
?>
```

Twierdzenie 10.0.1. *Twierdzenie Twierdzenie Twierdzenie Twierdzenie Twierdzenie*

Bibliografia

[1] http://www.wykresy.net/liniowe/liczba-hoteli-w-polsce-ostatnie-12-lat.html

[2] http://www.ranking.pl/pl/rankings/operating-systems.html

[3] http://db-engines.com/en/ranking

[4] http://stat.gov.pl/obszary-tematyczne/podmioty-gospodarcze-wyniki-finansowe/przedsiebiorstwa-niefinansowe/male-i-srednie-przedsiebiorstwa-niefinansowe-w-polsce-w-latach-2009-2013,22,1.html

[5] Jerzy Kisielnicki, Małgorzata Pańkowska, Henryk Sroka.:*Zintegrowane systemy informatyczne. Dobre praktyki wdrożeń systemów klasy ERP*, Wydawnictwo Naukowe PWN, Warszawa, 2012

[6] http://www.profitcrm.pl/crm

[7] http://meteoryt.pl/

[8] https://www.insert.com.pl

[9] https://msdn.microsoft.com/pl-pl/library/w0x726c2(v=vs.110).aspx

[10] https://msdn.microsoft.com/pl-pl/vstudio/cc511286

[11] http://www.thebestcsharpprogrammerintheworld.com/fundamentals/Base-Class-Library.aspx

[12] http://www.etechpulse.com/2013/04/difference-between-bcl-and-fcl-in-net.html

[13] https://msdn.microsoft.com/en-us/library/zcx1eb1e(v=vs.110).aspx

[14] https://msdn.microsoft.com/library/12a7a7h3(v=vs.100).aspx

[15] http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html

[16] http://antyweb.pl/hello-world-czyli-krotka-historia-jezykow-programowania/

[17] http://urriellu.net/en/articles-software/csharp-advantages.html

[18] Adam Nathan *WPF 4.5. Księga eksperta*, Wydawnictwo Helion, Gliwice , 2015

[19] http://www.w3schools.com/xml/xml_whatis.asp

[20] Itzik Ben-Gan *Microsoft SQL Server 2012. Podstawy języka T-SQL* Wydawnictwo Helion, Gliwice , 2012

[21] Jeffrey D. Ullman, Jennifer Widom *Podstawowy kurs systemów baz danych. Wydanie III* Wydawnictwo Helion, Gliwice , 2011

[22] https://www.visualstudio.com/pl-pl/downloads/download-visual-studio-vs.aspx

[23] https://www.visualstudio.com/support/legal/dn877550

[24] http://ms.polsl.pl/aktualnosci.php?news=149wid=22

[25] https://visualstudiogallery.msdn.microsoft.com

[26] https://www.microsoft.com/en-us/download/details.aspx?id=29062

[27] Krasiński M.: *Nauki o zarządzaniu*, Wydawnictwo Uniwersytetu Ekonomicznego we Wrocławiu, Wrocław, 2013, str.24-32

[28] https://trello.com/platforms

[29] http://www.gajdaw.pl/varia/subversion-system-kontroli-wersji-tutorial/p1.html

[30] http://blog.undicom.pl/systemy-kontroli-wersji-ktore-wybrac/

[31] https://git-scm.com/book/en/v2/Getting-Started-A-Short-History-of-Git

[32] https://git-scm.com/

[33] http://leniwy.eu/news,18,Git-poradnik-poczatkujacego.html

[34] https://answers.atlassian.com/questions/303753/sourcetree-free-license

[35] https://www.atlassian.com/software/sourcetree/overview

[36] https://github.com/blog/40-we-launched

[37] Chappell D.: *Understanding .NET (2nd Edition)*, Wydawnictwo Addison-Wesley Professional, Boston 2006

[38] Martin R.C.: *Czysty kod. Podręcznik dobrego programisty*, Wydawnictwo Helion, Gliwice 2014

[39] https://www.jetbrains.com/resharper/features/

[40] https://wiki.jenkins-ci.org/display/JENKINS/Meet+Jenkins

[41] http://techwriter.pl/jenkins-opis-narzedzia/

[42] https://opensource.org/licenses/MS-PL

[43] https://acrobat.adobe.com/pl/pl/products/about-adobe-pdf.html

[44] http://www.download.net.pl/jak-zapisac-dowolny-dokument-do-pliku-pdf/n/3662/

[45] http://www.pdfsharp.net/

[46] Jaszkiewicz A. *Inżynieria oprogramowania*, Wydawnictwo Helion, 1997

[47] Petzold C. *Windows 8. Programowanie aplikacji z wykorzystaniem C# i XAML*, Wydawnistwo Helion, Gliwice 2013

[48] Borycki D., Matulewski J., Pakulski M., Grabek M.: *ASP.NET MVC. Kompletny przewodnik dla programistów interaktywnych aplikacji internetowych w Visual Studio*, Wydawnictwo Helion, Gliwice 2014

[49] MacDonald M., *Pro WPF 4.5 in VB: Windows Presentation Foundation in .NET 4.5*, Wydawnictwo Apress, 2012

[50] LeBlanc Patrick *Microsoft SQL Server 2012. Krok po kroku*,Wydawnictwo Helion, Gliwice 2012

[51] https://msdn.microsoft.com/en-us/data/jj591583

[52] Magennis T., *LINQ to Objects w C# 4.0*, Wydawnictwo Helion, Gliwice 2012

[53] http://www.martinfowler.com/articles/continuousIntegration.html

[54] http://www.infoq.com/articles/cd-benefits-challenges

[55] https://puppetlabs.com/blog/continuous-delivery-vs-continuous-deployment-whats-diff

[56] http://docs.sonarqube.org/display/PLUG/Plugin+Library

[57] http://docs.sonarqube.org/display/SONARNEXT/Installing+and+Configuring+SonarQube+Scann

[58] http://www.sonarsource.com/products/plugins/governance/sqale/installation-and-usage/#understandingSqaleRatings

[59] Bruce Schneier, Schneier on Security: Cryptanalysis of SHA-1, 2005

[60] http://www.pzielinski.com/?p=831

Spis rysunków

1 Sposób dodawania towaru w ofercie, Źródło: Opracowanie własne. . . 18
2 Interfejs programu Asysten CRM, Źródło: Opracowanie własne. . . . 19
3 Interfejs programu Gestor GT, Źródło: Opracowanie własne. 20
4 Interfejs SQL Server Management Studio Express, Źródło: Opracowanie własne. 29
5 Interfejs aplikacji webowej Trello, Źródło: Opracowanie własne. 30
6 Szczegóły zadania na pojedynczej kafelce., Źródło: Opracowanie własne. 31
7 Atlassian SourceTree - okno główne, Źródło: Opracowanie własne. . . 34
8 Interfejs repozytorium w witrynie GiHub.com, Źródło: Opracowanie własne. 35
9 Statystyka wkładu pracy z dnia 13 grudnia 2015 r., Źródło: Opracowanie własne. 35
10 Informacja Resharpera dotycząca potencjalnej możliwości wystąpienia wyjątku NullReferenceException, Źródło: Opracowanie własne. . . 36
11 Kontrolka DatePicker, Źródło: Opracowanie własne. 39
12 Kontrolka TimePicker, Źródło: Opracowanie własne. 40
13 Przepływ danych w modelu MVVM, Źródło: www.tomaszmalesza.pl . 42
14 Raport statycznej analizy kodu źródłowego, Źródło: Opracowanie własne. 54
15 Lista zadań w SonarQube, Źródło: Opracowanie własne. 55
16 Plan ciągłej integracji w Jenkins'ie, Źródło: Opracowanie własne. . . 57
17 Okno procesu logowania, Źródło: Opracowanie własne. 58
18 Informacja otrzymania błędnych danych podczas operacji logowania, Źródło: Opracowanie własne. 59
19 Okno informacji podczas pierwszego logowania, Źródło: Opracowanie własne. 59
20 Okno informujące o polityce bezpieczeństwa haseł, Źródło: Opracowanie własne. 60
21 Okno tworzenia propozycji cenowej. Zakładka kontakt ze strony zamawiającego, Źródło: Opracowanie własne. 64

22 Okno tworzenia propozycji cenowej. Zakładka sali i jej wyposażenia, Źródło: Opracowanie własne. 65
23 Okno tworzenia propozycji cenowej. Zakładka elementów gastronomicznych, Źródło: Opracowanie własne. 66
24 Okno tworzenia propozycji cenowej. Zakładka dotycząca usług noclegowych, Źródło: Opracowanie własne. 67
25 Okno tworzenia propozycji cenowej. Zakładka dotycząca usług dodatkowych oraz formę zapłaty za organizowane wydarzenie, Źródło: Opracowanie własne. 68
26 Edycja propozycji, Źródło: Opracowanie własne. 69
27 Wyświetlanie listy propozycji cenowych, Źródło: Opracowanie własne. 70
28 Zapis propozycji cenowej do pliku PDF, Źródło: Opracowanie własne. 70
29 Poprawny zapis propozycji cenowej do pliku PDF, Źródło: Opracowanie własne. 71
30 Menu rozwijane modułu służącego do zarządzania kontami pracowników, Źródło: Opracowanie własne. 72
31 Dodawanie nowego konta użytkownika, Źródło: Opracowanie własne. 72
32 Komunikat o poprawnym zapisie konta użytkownika w bazie danych, Źródło: Opracowanie własne. 73
33 Komunikat o błędzie podczas zapisu konta użytkownika do bazy danych, Źródło: Opracowanie własne. 73
34 Tabela zawierająca dane użytkowników, Źródło: Opracowanie własne. 74
35 Okno usuwania kont użytkowników, Źródło: Opracowanie własne. . . 74
36 Komunikat o poprawnym usunięciu konta użytkownika z bazy danych, Źródło: Opracowanie własne. 75
37 Okno resetowania haseł użytkowników, Źródło: Opracowanie własne. 75
38 Komunikat o powodzeniu operacji usuwania konta użytkownika z bazy danych, Źródło: Opracowanie własne. 76
39 Ceny sal w poszczególnych miesiącach, Źródło: Opracowanie własne. . 77
40 Kontrolka PasswordBox z przedstawieniem aspektów bezpieczeństwa, Źródło: Opracowanie własne. 79

Kody źródłowe

1 Porównanie operacji dodania użytkownika w ADO.NET oraz Entity Framework 48
2 Opis polecenia select 50
3 Metoda weryfikująca poprawność wpisanego przez użytkownika hasła 80
4 Polecenie wyciągające wszystkie propozycje użytkownika 81
5 Polecenie wyciągające wszystkie propozycje użytkownika 82
6 Implementacja i opis funkcji aktualizującej bazę danych z poziomu DataGridView 83
7 Implementacja i opis funkcji tworzącej prostą tabelę przy użyciu biblioteki PDFSharp & MigraDoc 85