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. 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. Jarosław Karcewicz

1. **Łukasz Lis** – (1%)
   Wkład pracy lisa

2. **Sebastian Nalepka** – (3%)
   Wkład pracy seby

3. **Mateusz Ogiermann** – (0%)
   Wkład pracy ogi

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:112233, ... (podpis:) .....

Gliwice, dnia .....

Spis treści

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 spotkania 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.

Twierdzenie 1.1. *Twierdzenie Twierdzenie Twierdzenie Twierdzenie Twierdzenie*

Definicja 1.0.1. *Twierdzenie Twierdzenie Twierdzenie Twierdzenie Twierdzenie*

2. Analiza przedwdrożeniowa

3. Istniejące rozwiązania konkurencyjne

4. Zastosowane rozwiązania

4.1. Wykorzystana technologia

4.1.1. .NET/C#

4.1.2. WPF

4.1.3. XAML

XAML jest to oparty na XML[2] 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.

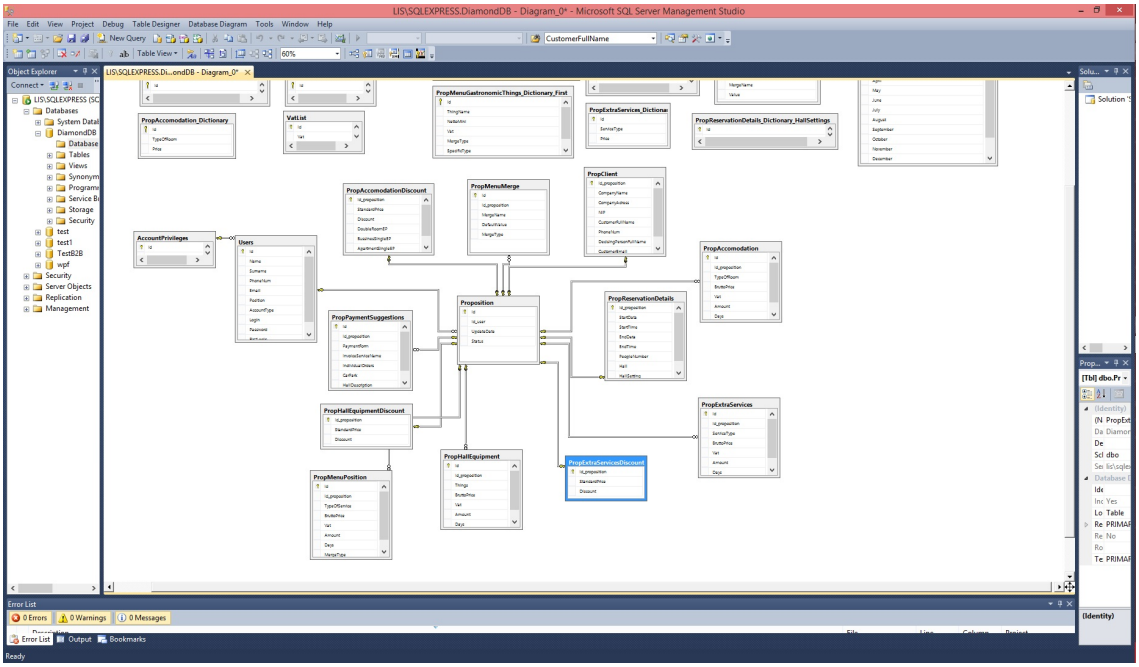
4.2. Wykorzystane narzędzia

4.2.1. Visual Studio 2013

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 [15] wynika to z istniejącej infrastruktury opartej na rozwiązaniach Microsoftu, co ułatwiło nam 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 zrzut ekranu. Umożliwia to diagnozowanie problemu braku połączeń niektórych tabel z resztą struktury.



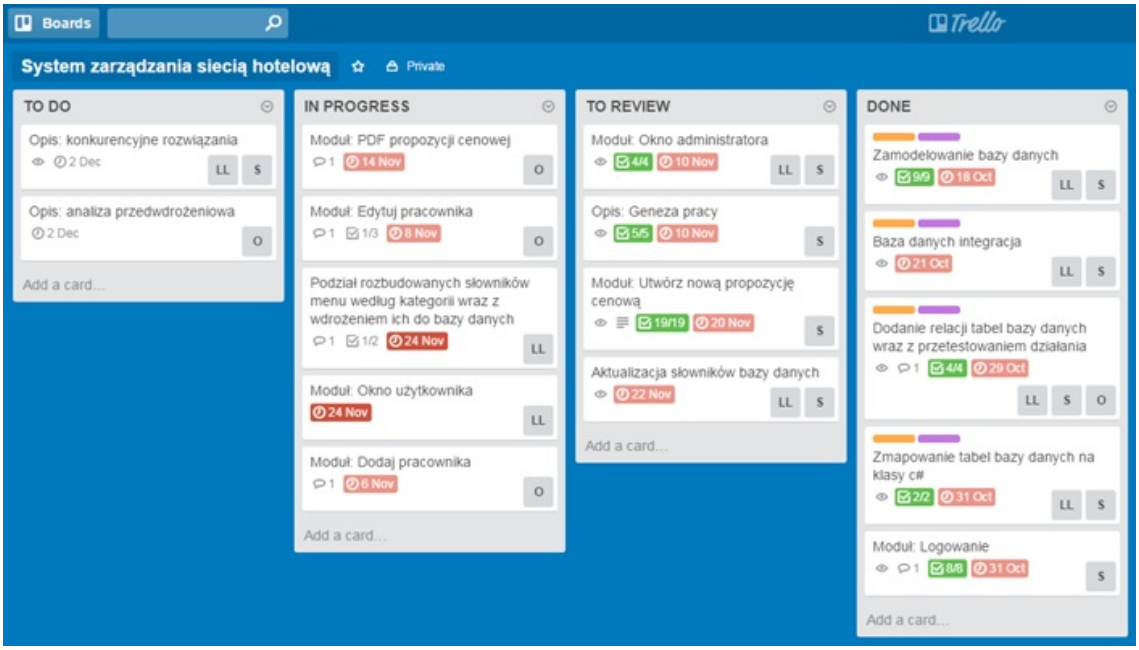
Rysunek 1: Interfejs SQL Server Management Studio Express, Źródło: Opracowanie własne.

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 naszego programu w trakcie pracy. Sprawdzić można czy wszystkie elementy prawidłowo działają i czy w poprawny sposób jest zapisywana dana pozycja.

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

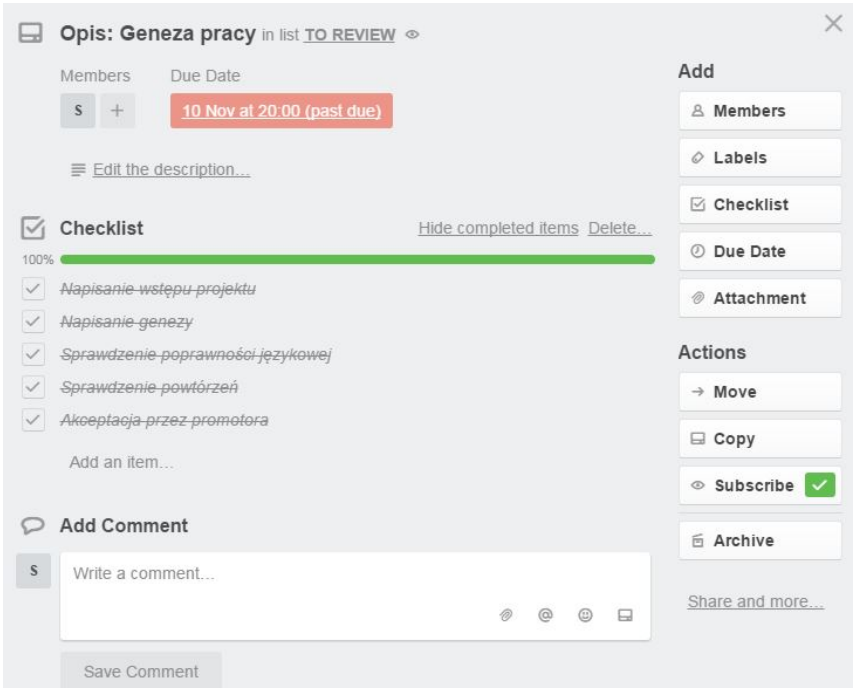
narzędziami, które umożliwiają prostą, szybką i skuteczną komunikację wspierającą pracę wielu osób nad jednym projektem. Trello jest to darmowa aplikacja, której przeznaczeniem jest wspieranie organizacji pracy grupowej opierając się na metodyce Kanban [3], która zaadoptowana została na potrzeby inżynierii oprogramowania. Potencjalny wzrost wydajności pracy przy użyciu Trello ma nastąpić dzięki scentralizowanemu systemowi zarządzania projektem, który opiera się na wirtualnych tablicach, na których ulokowane są listy zadań.



Rysunek 2: Interfejs aplikacji webowej Trello, Źródło: Opracowanie własne.

Listy te umożliwiają konstruowanie zadań w postaci tak zwanych kafelek, które w bardzo intuicyjny sposób można przenosić pomiędzy listami za pomocą techniki drag-and-drop. Każda kafelka reprezentująca zadanie w Trello posiada szereg dodatkowych funkcjonalności. 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, które w sposób automatyczny przekazywane są na skrzynkę elektroniczną lub ja-



Rysunek 3: Szczegóły zadania na pojedynczej kafelce., Źródło: Opracowanie własne.

ko 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 [4], 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 imlementacyjne budowanego systemu.

4.2.4. SourceTree + GitHub

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 [5] 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 ¡przyklad¿, ostrzeżeniem programisty przed kodem, który potencjalnie może doprowadzić do nieprawidłowego działania całego oprogramowania lub informacją o fragmencie, który jest kodem martwym [6].

Kolejnym bardzo istotnym modułem jest funkcja 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. Przeniesie 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 przy rozbudowanym systemie jest wyjątkowo przydatna i znacznie skraca czas wykonywanej czynności.

Przedstawione funkcjonalności ReSharpera wraz z wieloma innymi dostępnymi do zapoznania się pod [7] 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 wdrożonych rozwiązań.

4.2.6. Jenkins

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 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.

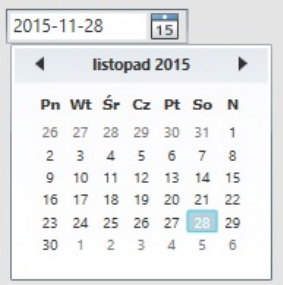
Liczba domyślnych kontrolek dla zaimplementowania niektórych funkcjonalności nie jest jednak wystarczająca. Odwołując się do przytoczonego przykładu wpisania daty przez użytkownika, WPF wymaga od programisty dużego wkładu pracy do prawidłowego wykonania tej funkcjonalności. Z pomocą przychodzą jednak 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 [8]. 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 których 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.

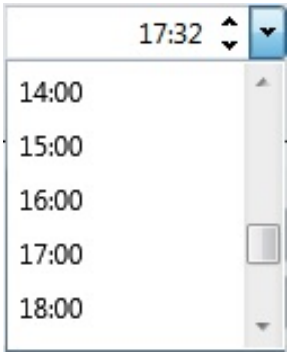
Kolejnym miejscem zastosowania biblioteki Extended WPF Toolkit w projekcie jest ustalanie z poziomu programu daty oraz godziny. Do tych celów posłużono się dwiema dodatkowymi kontrolkami: DatePicker oraz TimePicker. Kontrolka DatePicker umożliwia użytkownikowi wyboru daty w ustandaryzowanym formacie. 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 4: Kontrolka DatePicker, Źródło: Opracowanie własne.

Walidacja godziny, analogicznie jak daty, także może doprowadzić do wielu problemów na tle jednolitego formatu danych. Zastosowana kontrolka TimePicker 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.

4.3.2. PDFsharp & MigraDoc



Rysunek 5: Kontrolka TimePicker, Źródło: Opracowanie własne.

5. Implementacja

5.1. Architektura aplikacji

5.1.1. Wzorce architektoniczne oprogramowania

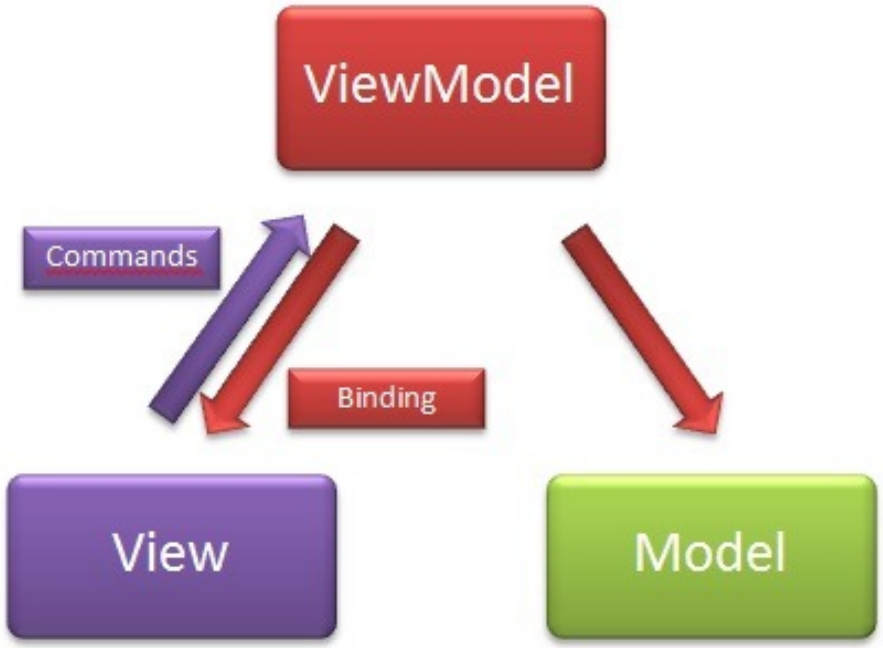
W inżynierii oprogramowania analogicznie jak w innych dziedzinach, w których przeprowadzana jest operacja budowania zadanego przedmiotu, należy starannie zaplanować jego proces. W tym calu należy kierować się przyjętymi fazami produkcji oprogramowania [9]. Fazy te mówią o tym, iż należy dokładnie określić 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, przetestowanie całego systemu oraz jego finalne uruchomienie 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 określonego 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) [10], 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.



Rysunek 6: Przepływ danych w modelu MVVM, Źródło: www.tomaszmalesza.pl

Model to warstwa, która odpowiedzialna jest w omawianym wzorcu za logikę biznesową aplikacji [11]. 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 [12] 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 zapew-

nia 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 ViewModelem, który ją kontroluje.

5.1.3. Model aplikacji

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

5.2.2. MS-SQL – Zastosowany system zarządzania bazą danych

5.2.3. Budowa bazy danych użytej w projekcie

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 baz danych. 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.

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

Listing 1: Dodanie rekordu w ADO.NET oraz Entity Framework

```
//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
DiamondDBEntities dataContext = new DiamondDBEntities();

TestUser newUser = new TestUser
{
    Login = "TestLogin", Password = "TestPa$$word",
};
dataContext.Users.Add(newUser);
dataContext.SaveChanges();
```

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 jest obecna działająca fizycznie baza danych. W tym przypadku, korzystając z wbudowanego we framework Designera, 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 [13].

Ostatnie podejście to Model First, które polega na zbudowaniu fizycznego modelu bazy danych posługując się kreatorem, który z znacznym stopniu ułatwia tworzenie encji oraz właściwości bazy danych. Ścieżka ta jest wykorzystywana w sytuacji

posiadania wyłącznie schematu bazy danych. Ze zbudowanego modelu generowana 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 [przypis] do operowania na bazodanowym źródle danych.

W celu wykonania polecenia SELECT zwracającego rekordy z bazy danych należy za pomocą przedstawionego 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 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.

Listing 2: Opis polecenia select

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

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, która reprezentuje interesującą nas 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 o interesującym nas typie. Do takiego utworzonego obiektu, programista ma możliwość odwoływania się i dokonywania

zmian w standardowy dla języka C# sposób. Po wszelkich modyfikacjach, wszystkie zmiany należy potwierdzić metodą *SaveChanges* na obiekcie modelu Entity, która uaktualnia zmodyfikowane elementy na poziomie bazy danych.

Listing 3: Potwierdzenie dokonanych zmian

```
_context.SaveChanges()
```

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.

Listing 4: Dodanie rekordu do tabeli

```
_context.TableName.Add(tableObject);
```

5.4. Automatyzacja wydawania kolejnych wersji programu

5.4.1. Continuous Integration – definicja

5.4.2. Continuous Deployment – definicja

5.4.3. Continuous Delivery – definicja

5.4.4. Schemat budowania programu

6. Testowanie aplikacji

6.1. 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 okno, którego zadaniem jest sprawdzenie jego tożsamości oraz w sytuacji jej potwierdzenia, przełączenie do odpowiedniego okna głównego aplikacji, w zależności od posiadanych uprawnień.

SCREEN OKNA LOGOWANIA (wrzucić po poprawkach wizualnych programu)

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

TODO: SCREEN BŁĘDNEGO WPISANIA DANYCH (wrzucić po poprawkach wizualnych programu)

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, która instruuje go o dalszych krokach.

TODO: /odnośnik do zrzutu ekranu ilustrującą opisaną sytuację/ (wrzucić po poprawkach wizualnych programu)

TODO: SCREEN PIERWSZEGO LOGOWANIA (wrzucić po poprawkach wizualnych programu)

6.2. Zarządzanie kontem sprzedawcy oraz administratora (obsługa)

6.3. 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 podczas spotkania 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 spotkania 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.

6.3.1. Opis klienta

IN PROGRESS

Pierwsza zakładka 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 dzięki przechowywaniu w programie informacji o tym, na jakie konto użytkownik został zalogowany. Wszelkie pozostałe

dane sprzedawca ma obowiązek uzupełnić samodzielnie, uzyskując informacje od 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 wystąpiła konieczność rozdzielenia danych adresowych - potrzebne są one tylko i wyłącznie do opisu klienta na wygenerowanym dokumencie PDF.

- *NIP*

  Kolejne pole opcjonalne, które w przypadku klientów biznesowych przeznaczone jest do wpisania numeru identyfikacji podatkowej.

- *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 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 posiada uprawnienie do dokupywania usług nieuwzględnionych w propozycji cenowej w czasie trwania wydarzenia. Zdarza się, iż podczas wydarzenia,

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 może być 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 pomieszczenie oraz jej cenę, której rekord w sposób automatyczny dodawany jest jako pierwszy w kolejnej zakładce opisującej szczegóły rezerwacji.

- ...

TODO: po edycji graficznej wrzucić tab1 testuje jaki jest margines testuje jaki jest margines testuje jaki jest margines testuje jaki jest margines

6.3.2. Szczegóły rezerwacji

TODO: po edycji graficznej wrzucić tab2

6.3.3. Usługi gastronomiczne

TODO: po edycji graficznej wrzucić tab3

6.3.4. Usługi noclegowe

TODO: po edycji graficznej wrzucić tab4

6.3.5. Usługi dodatkowe i forma płatności

TODO: po edycji graficznej wrzucić tab5

6.4. Edycja istniejącej propozycji cenowej

6.5. Tworzenie pliku PDF istniejącej propozycji cenowej

6.6. Modyfikacja kont użytkowników/sprzedawców

6.6.1. Dodawanie konta

6.6.2. Edycja konta

6.6.3. Usuwanie konta

6.7. Modyfikacja słowników cenowych

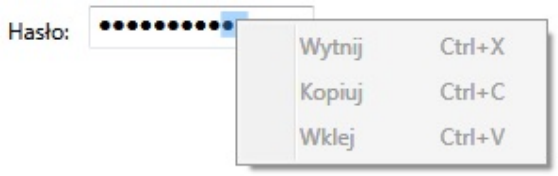
7. Specyfikacja wewnętrzna

7.1. Przeprowadzenie operacji logowania użytkownika

7.1.1. 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. 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 7: Kontrolka PasswordBox z przestawionymi aspektami 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, hasło przekształcane jest za pomocą algorytmu haszującego SHA-256 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.

TODO: zrzut rekordu w bazie danych (hashe hasel) + kod porownujacy hasla: wrzucić gdy włączę haszowanie w aplikacji

7.1.2. Szyfrowanie haseł - funkcja skrótu SHA2

potrzebne ????

7.2. Konto administratora i użytkownika – ListView

7.3. Tworzenie propozycji cenowej

7.4. Edycja słowników oraz kont użytkowników

7.4.1. DataGridView a dynamiczna komunikacja z Bazą danych

7.5. Tworzenie szablonu PDF propozycji cenowej

8. Perspektywy rozwoju oprogramowania

- wersja offline - pierwsze logowanie: ponowne wpisanie hasła dla potwierdzenia zgodności - dynamicznie rozwijane listy podczas tworzenia propozycji cenowej (ilość rzeczy do wyboru w każdym tabie)

9. 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.

Rysunki

CACHE

Załączniki

CACHE

Tu programy

```
//ADO.NET
    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$$word"));

    connection.Open();
    command.ExecuteNonQuery();
    connection.Close();

    //Entity Framework
    DiamondDBEntities dataContext = new DiamondDBEntities();
    // EF automatycznie odczyta plik konfiguracyjny aplikacji

    TestUser newUser = new TestUser
    {
        Login = "TestLogin", Password = "TestPa$$word",
    };
    dataContext.Users.Add(newUser);
    dataContext.SaveChanges();
```

Oraz

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

Twierdzenie 9.0.1. *Twierdzenie Twierdzenie Twierdzenie Twierdzenie Twierdzenie*

Literatura

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

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

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

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

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

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

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

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

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

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

[11] 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

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

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

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

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

[16] Jakaś pozycja literatury

Spis rysunków