POLSKO-JAPOŃSKA AKADEMIA TECHNIK KOMPUTEROWYCH

Wydział Informatyki

Katedra Sieci Komputerowych

Specjalizacja: Programowanie Systemowe i Sieciowe

Robert Kilar

Nr albumu 8359

Implementacja metodyki pracy zdalnej dla wirtualnego call center opartej na rozwiązaniach chmurowych

Praca magisterska pod kierunkiem dr inż. Mariusza Trzaski

Warszawa, wrzesień, 2015

Streszczenie

Niniejsza praca skupia się na zjawisku zastępowania przestarzałych architektur sprzętowych rozwiązaniami programowymi i towarzyszącymi temu procesowi skutkami. Autor przedstawia implikacje tych zmian w postaci obniżenia kosztów, większej elastyczności systemu, wsparcia dla telepracy, zmniejszeniu ryzyka awarii i ograniczeniu jej następstw na przykładzie implementacji rozwiązania programowego dla centrum telefonicznego. Zaproponowany w tej pracy prototyp został wykonany przy użyciu technologii ASP.NET-MVC5 i może być użytkowany niezależnie od systemu operacyjnego konsumenta. Dzięki interfejsowi programistycznemu aplikacji funkcjonalność wytworzonego systemu jest stosunkowo łatwo rozszerzalna, także przy wykorzystaniu innych niż ASP.NET-MVC technologii.

Spis treści

1.Wstęp

W dzisiejszych czasach pracodawcy coraz częściej wybierają model pracy zdalnej, czyli takiej w której pracownik wykonuje co najmniej znaczną część swojej pracy poza siedzibą firmy go zatrudniającej. Jest to stosunkowo młoda forma organizacji pracy pozwalająca między innymi na znaczne ograniczenie kosztów ponoszonych przez pracodawcę jak i pracownika. Pracodawca obniża koszty z tytułu wynajmowanej powierzchni biurowej, mediów, ograniczonej floty samochodowej, miejsc parkingowych. Z kolei pracownik nie ponosi kosztów dojazdu do pracy, nie spędza również czasu w drodze do pracy przyczyniając się do spadku natężenia ruchu drogowego w godzinach szczytu.

Narzędziami telepracy mogą być łatwo dostępne urządzenia takie jak: komputer osobisty, tablet lub telefon komórkowy wspierający szerokopasmowy dostęp do internetu. Uwzględnić należy również również usługi mobilne, telefonię komórkową, Voice over Internet Protocol, komunikatory internetowe(Skype, GG) oraz pocztę elektroniczną. Instrumentami pracy zdalnej są także wyrafinowane systemy implementujące daną metodykę pracy w firmie. Przykładem takiego rozwiązania może być system Customer Relationship Management(CRM) zaprojektowany pod kątem pracy na odległość.

W wielu centrach telefonicznych średniego rozmiaru w Polsce do zarządzania połączeniami wykorzystywana jest architektura oparta na rozwiązaniach sprzętowych. Przykładowo przy zastosowaniu w/w podejścia dla małego biura w którym pracuje 10 konsultantów do kosztów jego wynajęcia należy dodać koszt niezbędnego wyposażenia. W przypadku call center wykonującego tylko rozmowy wychodzące są to:

- centrala abonencka (Private Branch Exchange, PBX) – umożliwia stworzenie wewnętrznej sieci telefonicznej w obrębie przedsiębiorstwa. Pozwala na nieobciążone opłatą ze strony zewnętrznego operatora prowadzenie rozmów przez pracowników biura. PBX pozwala również na obsługę ruchu telefonicznego wychodzącego lub przychodzącego spoza instytucji.

- moduł nagrywający rozmowy – umożliwia przeprowadzanie kontroli jakości pracy konsultantów poprzez nadzorców grupy odsłuchujących wykonane rozmowy. Wybrane rozmowy są również jednym z podstawowych materiałów szkoleniowych dla nowych pracowników. Element systemu niezbędny na wypadek roszczeń reklamacyjnych ze strony klientów przedsiębiorstwa. Moduł ten pełni również rolę w oszacowaniu ile czasu pracownik rzeczywiście poświęca na powierzoną mu pracę.

- telefony systemowe – wymagane są kosztowne telefony przystosowane do współpracy z centralą PBX

- okablowanie w postaci linii telefonicznych

- wykwalifikowany personel do wsparcia technicznego – na wypadek awarii oraz do codziennej konserwacji systemu

- kupno oferty u tradycyjnego operatora telefonii

Dodatkowo w przypadku obsługi rozmów przychodzących uwzględnić:

- Automatyczny system odpowiedzi głosowej (Interactive Voice Response, IVR) – umożliwia interaktywną obsługę osoby dzwoniącej, na przykład poprzez wybór tonowy DTMF opcji po wysłuchaniu nagranego menu głosowego

- Automatyczny system dystrybucji połączeń (Automatic Call Distributor, ACD) – bierze udział przyporządkowaniu rozmowy do właściwego konsultanta jak i umożliwia oczekiwanie na linii w przypadku gdy wszyscy konsultanci prowadzą rozmowy

Kwota niezbędna na zakup wymienionego wyposażenia rozpoczyna się od 20 tysięcy złotych. Do podanej sumy należy doliczyć koszt instalacji systemu oraz pozostałe koszty okresowe takie jak wynajem powierzchni biurowej dla kilkunastu osób i pensje dla pracowników. Przy obecnych cenach rynkowych w/w sprzętu koszt otwarcia małego call center wydaje się być kwotą zaporową.

Sprzętowe podejście poza wysoką ceną niesie ze sobą również wiele ograniczeń. W przypadku pojawienia się nowych potrzeb, na przykład w sytuacji powiększenia personelu poza zwiększeniem wynajmowanej powierzchni biurowej niezbędna jest przebudowa architektury całego systemu, który musi podołać nowym wyzwaniom, co może skutkować koniecznością wymiany niemal wszystkich elementów instalacji. Kolejnym utrudnieniem jest brak możliwości prowadzenia pracy zdalnej. Pracownik w przypadku niemożności przybycia do biura z powodu kontuzji, awarii środka transportu, zatoru drogowego lub innych przyczyn nie może podjąć obowiązków służbowych poza zakładem pracy. Należy również nadmienić o braku dywersyfikacji ryzyka w przypadku awarii. W sytuacji gdy jedyną możliwością wykonywania pracy jest wykorzystanie instalacji znajdującej się w biurze, jej usterka powoduje całkowity paraliż firmy na czas jej usunięcia, co prowadzi do powstania dużych strat.

Z powyższych przesłanek wynika wniosek wskazujący, że podejście sprzętowe ogranicza możliwości rozwoju przedsiębiorstwa, a wysoki koszt zakupu i instalacji systemu nie jest rekompensowany przez widoczne zalety.

Rozwinięciem czysto sprzętowego podejścia jest wzbogacenie go o moduł integracji komputer-telefon (Computer Telephony Integration, CTI). Rozwiązanie CTI sprowadza się do aplikacji pozwalającej na sprawniejsze zarządzanie danymi, planowanie połączeń. W zależności od stopnia zaawansowania może umożliwiać w miernym stopniu pracę zdalną, jednakże zazwyczaj wyłącznie na komputerach

klasy PC, to jest z wyłączeniem urządzeń mobilnych oraz po uprzedniej instalacji i konfiguracji przez zaawansowanego technika zatrudnionego przez firmę call center. Obsługa urządzeń mobilnych jest jednakże kluczowa w przypadku agentów terenowych mających w bezpośredni kontakt z klientem.

Technologie CTI jednak nie rozwiązują problemów braku alternatywnych rozwiązań w sytuacji awarii sprzętu, rozwoju architektury w przypadku rozrostu firmy, ponieważ cały system call center nadal opiera się na własnościowym zapleczu sprzętowym.

Kolejnym zagadnieniem są nadmienione ograniczenia w aspekcie prowadzenia telepracy. Jej liczne zalety zostały wymienione w bieżącym rozdziale, jednakże w kontekście call center uwypuklone zostają dodatkowe korzyści. Po stronie pracodawcy oprócz spadku kosztów wynajmu powierzchni biurowej, rachunków za media pojawia się również możliwość przeniesienia części kosztów za sprzęt na rzecz pracownika. W zależności od polityki przedsiębiorstwa, w przypadku posiadania przez pracownika: telefonu komórkowego, tabletu, innych urządzeń mobilnych, a w szczególności notebooka lub komputera klasy PC; może on być wykorzystywany do celów służbowych w przypadku przedsiębiorstwa wymagającego od pracowników prowadzenia jednoosobowej działalności gospodarczej, co jest warunkiem koniecznym do uniknięcia odpowiedzialności z tytułu Art. 67. § 1 Kodeksu Pracy[7]. Taki rodzaj stosunków pomiędzy pracodawcą a pracownikiem, który formalnie jest relacją pomiędzy kontrahentami jest szczególnie popularny w marketingu wielopoziomowym oraz wśród firm ubezpieczeniowych, gdzie sprawne systemy CRM są istotnym składnikiem sukcesu.

Zalety telepracy zostały wymienione już we wstępie, jednakże pominięte zostały kwestie będące wyzwaniem postawionym przed projektantem metodologii pracy zdalnej w przedsiębiorstwie. Do tych problemów należą obawy pracodawcy: kontrola nad wydajnością pracy, oszacowanie realnego czasu poświęconego przez pracownika na pracę a co za tym idzie wyliczenie należnej pensji, inspekcja sprawdzająca czy pracownik stosuje się do obowiązujących w przedsiębiorstwie procedur, miejsce stanowiska pracy, kontakt przełożonego z pracownikiem; innymi słowy uniknięcie konsekwencji wynikających z braku przebywania pracownika w placówce przedsiębiorstwa w postaci utraty nad nim kontroli.

Powyższa charakterystyka klasycznych rozwiązań stawia przed autorem problem stworzenia modelu elastycznego rozwiązania informatycznego i towarzyszącej mu metodologii pracy pozwalającej na znaczne obniżenie kosztów rozpoczęcia działalności na rynku call center poprzez wysoki stopień uniezależnienia od fizycznych urządzeń telekomunikacyjnych w siedzibie przedsiębiorstwa oraz umożliwiający prowadzenie telepracy w kontrolowanych przez pracodawcę warunkach.

2. Analiza dostępnych na rynku rozwiązań wspierających pracę w call center

W tym rozdziale autor precyzuje pojęcia telepracy oraz call center w celu doprecyzowania dziedziny problemowej. Autor również zestawia ze sobą dwa rozwiązania funkcjonujące na rynku, będące w wielu aspektach swoimi przeciwieństwami. Z pierwszym rozwiązaniem opisanym w rozdziale 2.3 autor zetknął się podczas wykonywania pracy w call center. Jest ono wykorzystywane po dziś dzień[1] w omawianej korporacji. Z powodu wysokich kosztów jest w niskim stopniu dostępne publicznie, zatem mało prawdopodobna jest styczność z nim poza środowiskiem pracy. Z kolei drugie rozwiązanie jest przystępne cenowo i łatwo dostępne dla każdego zainteresowanego.

2.1. Precyzyjna definicja telepracy

Praca zdalna zwana również telepracą jest stosunkowo młodym zjawiskiem na polskim rynku. Polski Kodeks Pracy reguluje stosunek pracodawcy z pracownikiem oraz warunki wykonywania obowiązków dopiero od zmiany Dz.U. 2007 nr 181 poz. 1288. Na wstępie autor przytacza artykuły z obowiązującego Kodeksu Pracy[7].

Art 675 § 1

„Praca może być wykonywana regularnie poza zakładem pracy, z wykorzystaniem środków komunikacji elektronicznej w rozumieniu przepisów o świadczeniu usług drogą elektroniczną (telepraca).”

Art 675 § 2

„Telepracownikiem jest pracownik, który wykonuje pracę w warunkach określonych w § 1 i przekazuje pracodawcy wyniki pracy, w szczególności za pośrednictwem środków komunikacji elektronicznej.”

oraz definicję telepracy sformułowaną poprzez Komisję Europejską:

„Telepraca jest to metoda organizowania i wykonywania pracy, w której pracownik pracuje poza miejscem pracy pracodawcy przez znaczną część swojego czasu pracy, dostarczając do pracodawcy wyniki [rezultaty] pracy przy wykorzystaniu technologii informacyjnych oraz technologii przekazywania danych, zwłaszcza Internetu”

[1] czerwiec, 2015

Z powyższych przepisów można wywnioskować, że wyniki pracy są niematerialne i nie wymagają bezpośredniego kontaktu fizycznego z osobami trzecimi. Zatem telepraca jest idealnym modelem dla osób wykonujących zawód konsultanta telefonicznego w call center. Konsultant analizuje, przetwarza informacje i w większości przypadków nie nawiązuje fizycznego kontaktu z klientem.

2.2. Call center

Centrum telefoniczne, jest to „całość elementów sprzętowych, programowych służących do obsługi masowych kontaktów z klientami”[1] przy użyciu medium komunikacyjnego. Jako obsługę kontaktów rozumie się obsługę połączeń przychodzących od klientów oraz wychodzących inicjowanych przez pracowników centrum telefonicznego, częstokroć na zlecenie firmy zewnętrznej. Kontaktom z klientami towarzyszą czynności biznesowe takie jak ustalanie warunków oraz sprzedaż towarów i usług. W ostatnim czasie pojawił również nowy termin „contact center” będący rozszerzeniem call center o metody komunikacji przez internet takie jak komunikatory internetowe, czaty, czy rzadziej wideorozmowy poprzez Skype.

2.3. Rozwiązanie wdrożone w firmie Linea Directa Communications

Firma działająca w sektorze telesprzedaży. Wykorzystuje klasyczne rozwiązanie sprzętowe opierające się o centralę PBX, moduł IVR i ACD. Integrację systemu telefonicznego z komputerowym zapewnia program Abraxas Cat Pro. Jest to szablonowy przykład przestarzałego rozwiązania stosowanego w polskich firmach. Firma wynajmuje ponad tysiąc metrową powierzchnię biurową dla personelu pracującego na trzy zmiany korzystającego z biurowej stołówki i kuchni.

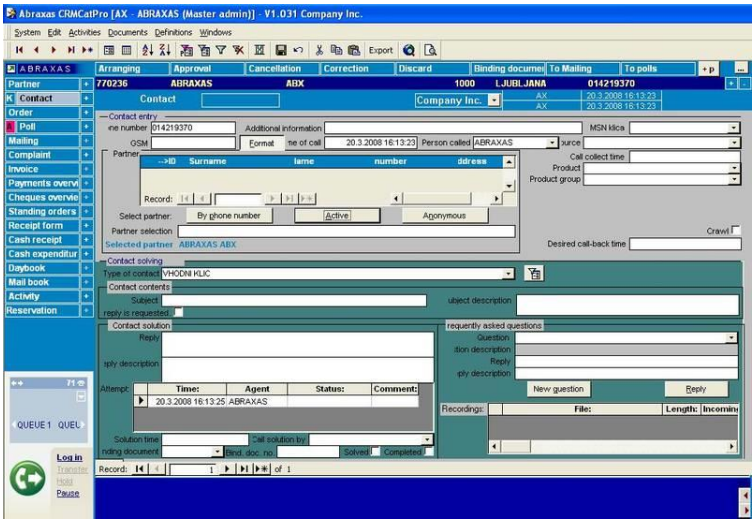
Strona informatyczna architektury korzysta z oprogramowania typu gruby klient CTI Abraxas Cat Pro. Jest to dojrzałe oprogramowanie, którego najważniejszym zadaniami są:

- udostępnianie danych klientów
- wylosowanie klienta z którym telemarketer ma się połączyć
- zarządzanie rozmową i przypisanie jej statusu informującego o powodzeniu lub jego braku
- ustalić termin kolejnej rozmowy z danym klientem, po którego upłynięciu dane klienta są automatycznie aportowane
- sporządzić zamówienie dla danego klienta
- zliczanie wykonanych rozmówców

- umożliwić pracownikowi sporządzenie raportu na koniec pracy ile czasu spośród dnia przeznaczył na dane czynności

Największymi wadami programu są:

- archaiczny, nieczytelny oraz nieergonomiczny interfejs użytkownika powodujący przyspieszone zmęczenie u użytkownika, jest on widoczny na rysunku 1.
- nagminne zawieszanie się systemu
- nabycie umiejętności posługiwania się Abraxas Cat Pro w stopniu co najmniej umiarkowanym przez osobę bez wykształcenia technicznego wymaga odbycia szkolenia przez profesjonalistę
- brak jakiegokolwiek wsparcia dla telepracy, zainstalowanie i skonfigurowanie systemu poza infrastrukturą przedsiębiorstwa jest niemożliwe
- ze względu na niestabilność systemu, nad funkcjonowaniem programu w miejscu pracy musi czuwać technik
- brak wbudowanej ochrony danych osobowych uniemożliwiającej kopiowanie ich poza okno programu
- brak wbudowanej ochrony danych osobowych przeciwdziałającej edycji przez nieupoważnione osoby, agent może edytować/niszczyć dane dowolnego klienta
- brak możliwości porozumiewania się z klientem drogą pisemną: e-mail, czat, komunikator
- aplikacja nie przewiduje API umożliwiającego rozwinięcie systemu o dodatkowe moduły przez samego przedsiębiorcę, bez udziału producenta



Rysunek 1. Graficzny interfejs użytkownika programu Abraxas Cat Pro

Opisany model ma za zadanie przybliżyć czytelnikowi jak wygląda w praktyce architektura rozwiązań stosowanych w średnich call center w Polsce, w szczególności założonych przed erą tzw. cloud computingu.

2.4. INCC

Dostępny od 2012 roku system do zarządzania relacjami z klientem (client relationship management, CRM). W odróżnieniu od wcześniej opisanego jest to rozwiązanie w pełni programowe. Składa się z dwóch modułów:

- usługi inCC CRM Call Center świadczonej metodą SaaS(System as a Service, system jako usługa)
- aplikacji inCall typu softphone

Aplikacja CRM jest pozbawiona zdecydowanej większości wad programu Abraxas Cat Pro. Jej interfejs graficzny jest dość ubogi estetycznie, jednakże przejrzysty i funkcjonalny. Program Call Center funkcjonuje na zewnętrznym serwerze i jest dostępny poprzez dowolną przeglądarkę internetową w formie strony internetowej, jest zatem wieloplatformowy i nie wymaga żadnej instalacji jak i konfiguracji ze strony szeregowego pracownika.

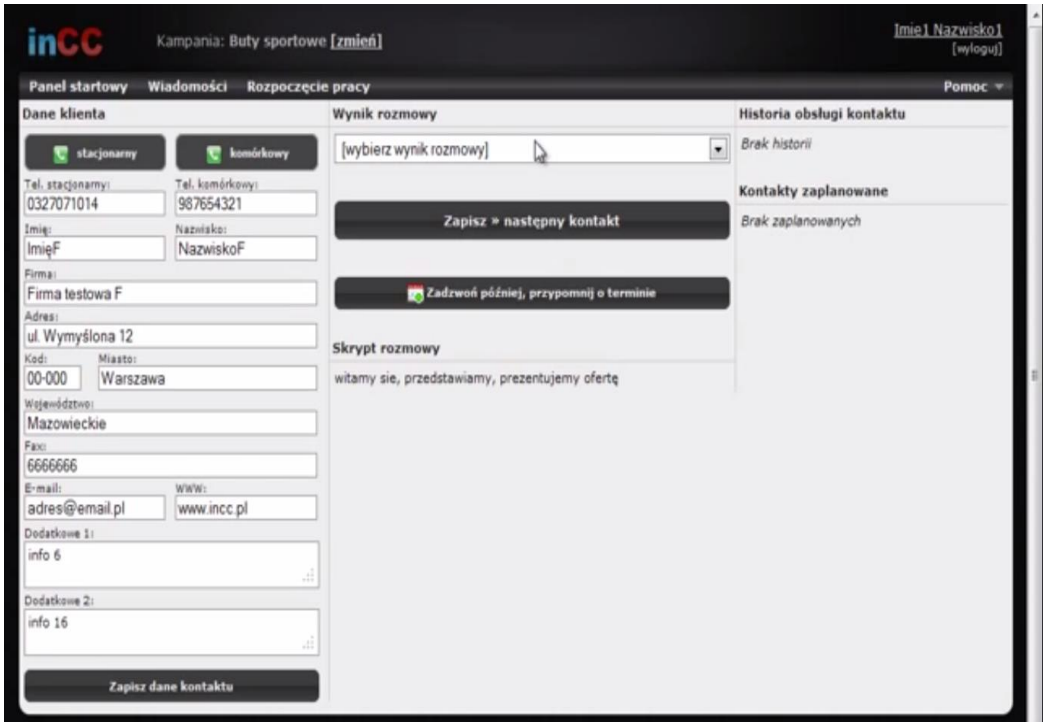
Aplikacja umożliwia:

- tworzenie kont użytkowników z podziałem na role do których przypisane są konkretne uprawnienia
- tworzenie grup użytkowników i przypisywanie ich do konkretnych kampanii
- dodawanie/edycję/usuwanie kontaktów
- zarządzenie skryptem rozmowy przypisywanym do danej kampanii
- określanie wyniku rozmowy w formie klasyfikatora i sporządzanie na ich podstawie statystyk rozmów zakończonych danym statusem
- przysyłanie wiadomości do pracowników
- import kontaktów z plików arkuszy kalkulacyjnych
- wyświetlanie statystyk dla danego agenta, kampanii
- odsłuchiwanie rozmów danego pracownika

Graficzny interfejs użytkownika aplikacji inCC CRM umieszczony na rysunku 2. nie jest jednak responsywny co utrudnia korzystanie z niego na urządzeniach przenośnych. Poprzez responsywność autor rozumie automatyczne dostosowywanie interfejsu użytkownika do rozdzielczości ekranu poprzez zmianę rozkładu okna lub stylu wyświetlanych elementów bez konieczności ponownego ładо-

wania strony internetowej. Program nie umożliwia również inicjowania połączeń bezpośrednio poprzez stronę internetową, lecz wymaga zainstalowania dodatkowej aplikacji inCall dostępnej wyłącznie pod system Windows co ogranicza wygodę korzystania z produktu na urządzeniach mobilnych.

Projektanci aplikacji nie przewidzieli również API dla swojej aplikacji uniemożliwiając tym samym współpracę pomiędzy nią a innymi aplikacji wytworzonymi przez/na potrzeby przedsiębiorstwa.



Rysunek 2. Graficzny interfejs użytkownika programu inCC Call Center CRM

2.5. Podsumowanie

W tabeli 1. autor zestawił ze sobą dwa odmienne implementacje rozwiązań dla firm call center. Poniższa tabelka zestawia różnice w charakterystyce obu rozwiązań w kluczowych aspektach.

Tabela 1. Porównanie cech dostępnych na rynku rozwiązań

System Cecha	Abraxas Cat Pro	inCC
Uniezależnienie od telefonicznej architektury sprzętowej w miejscu pracy	Brak	Jest
Umożliwia inicjalizację połączenia w programie	Tak	Tak

Zarządzanie połączeniem w programie	Jest	Brak, softphone pod system Windows
Wieloplatformowy	Nie	Częściowo
Interfejs użytkownika	Skomplikowany, przeładowany, jaskrawy	Prosty, przjrzysty, o stonowanych barwach
Responsywny interfejsu użytkownika	Brak	Brak
Integracja z komunikatorami internetowymi	Brak	Brak
Wymagana instalacja systemu na komputerach/urządzeniach mobilnych	Tak	Tak
Konfiguracja systemu	Skomplikowana	Prosta
Czas opanowywania systemu przez agenta	Długi	Krótki
Ochrona danych osobowych przed kopiowaniem	Brak	Brak
Możliwość tworzenia zamówień	Jest	Brak
Możliwość generowania faktur	Jest	Brak
Przystępność cenowa dla przedsiębiorcy	Niska	Wysoka
Kontrola lokalizacji wykonywania pracy	Brak	Brak
Interfejs programistyczny API	Brak	Brak

Powyższa analiza uwypukla problemy i braki z którymi borykają się użytkownicy rozwiązań dla call center. Jest nim brak prawdziwej wieloplatformowości oprogramowania z powodu instalacji samego CRMu lub konieczności instalacji dodatkowego softphone'a, który jest dedykowany jednemu systemowi operacyjnemu, zamiast np. konkretnej przeglądarce jako wtyczka. Przeszkodą jest także zamknięcie systemu poprzez brak API.

Powyższe przesłanki skłaniają do rozwinięcia listy wymagań stawianych przed autorem i sformułowanych we wstępie do pracy o postulat wieloplatformowości oraz integracji obsługi połączeń z systemem wsparcia call center, a także otwarciu oferowanej metody pracy w call center poprzez udostępnienie API dla użytkowników.

3. Propozycja rozwiązania

Rozdział ten ma za zadanie nakreślić koncepcję rozwiązania wybraną przez twórcę pracy. Autor wymienia kolejno motywacje stojące za zaprojektowaniem nowego rozwiązania oraz wymagania funkcjonalne i niefunkcjonalne, a także sposób sprostania im.

3.1. Motywacja proponowanego rozwiązania

Na podstawie analizy rozwiązań zawartej rozdziale 2. można wywnioskować, że obecnie stosowane rozwiązania są dalekie od optymalnych. Stosują one przestarzałe metody oparte o architekturę sprzętową w placówce przedsiębiorstwa. Nie wykorzystują również w pełni potencjału leżącego w przeglądarkach internetowych, dostępnych nie tylko na komputerach stacjonarnych, lecz także urządzeniach mobilnych. Celem autora jest zaproponowanie rozwiązania wolnego od wad przedstawionych aplikacji.

3.2. Wymagania funkcjonalne

Po szczegółowej analizie dobrych i złych stron istniejących systemów oraz możliwości obecnie dostępnych technologii określono wymagania funkcjonalne kreowanego rozwiązania:

- przechowywanie oraz zarządzanie danymi osobowymi klientów
- umożliwienie wyszukiwania klientów poprzez daną frazę
- dostęp do historii rozmów z klientem
- dodawanie raportów do danego klienta
- przypisywanie klientów do danych użytkowników
- możliwość przeprowadzania ankiet i eksportu ich wyników
- powiadomienia o ogłoszeniach w systemie
- automatyczny przydział rozmówców do użytkownika systemu
- możliwość zarządzenia połączeniem telefonicznym wewnątrz aplikacji
- przeprowadzanie audytu pracownika poprzez:
    - odsłuchiwanie wykonanych połączeń
    - badanie wydajności w zadanych okresach
    - śledzenie lokalizacji pracownika pracującego zdalnie
    - odnotowywanie logowań do systemu
- możliwość korzystania z bazy danych aplikacji poprzez interfejs programistyczny(API)

3.3. Wymagania niefunkcjonalne

Autor określił następujące wymagania niefunkcjonalne dla opracowywanego rozwiązania:

- uniezależnienie od platformy sprzętowej, którą pracownik będzie wykorzystywał do wykonywania pracy
- wsparcie dla telepracy poprzez umożliwienie korzystania z systemu poza placówką przedsiębiorstwa
- niezależność systemu call center od sprzętowej architektury telefonicznej w siedzibie przedsiębiorstwa
- komfort użytku na urządzeniach mobilnych
- wysoka dostępność usługi
- autentykacja użytkowników
- podział uprawnień pomiędzy użytkowników oraz autoryzacja dostępu do wybranych obszarów aplikacji
- utrudnienie eksportu danych osobowych przez pracowników

3.4. Opis rozwiązania

W tym podrozdziale autor opisuje wybrany sposób rozwiązania wymagań wymienionych w podrozdziałach 3.2 i 3.3.

3.4.1. Przechowywanie informacji

Aby rozwiązać problem przechowywania dużej ilości informacji o klientach przedsiębiorstwa, należy skorzystać z bazy danych i odpowiedniego oprogramowania ORM umożliwiającego odwzorowanie obiektowej architektury systemu na relacyjną bazę danych. Autor rozwinął wątek ORM w rozdziale 4.14. Aplikacja ma także posiadać funkcjonalność wypełniania ankiet, których wyniki zapisane w bazie danych będzie można wyeksportować do arkusza kalkulacyjnego.

3.4.2. Wieloplatformowość rozwiązania

System informatyczny można w pełni uniezależnić od platformy poprzez stworzenie jego wersji dla każdej z nich lub zbudować jedną wersję wykorzystującą mechanizmy, które już zostały na nich zaimplementowane. Jeżeli wszystkie interesujące autora platformy dysponują wsparciem dla technologii o którą autor może oprzeć swoje rozwiązanie, drugie z wymienionych podejść znacznie przyspiesza proces powstawania użytecznego oprogramowania.

Cechą wspólną wszystkich popularnych platform stacjonarnych i mobilnych:

- Microsoft Windows
- Mac OSX
- Linux
- Android
- Windows Phone
- iOS

jest bezsprzecznie nacisk na dostęp do Internetu i tym samym wyposażenie w przeglądarki internetowe.

Wypływa stąd wniosek, że aby umożliwić użytkownie z systemu na wielu platformach i jednocześnie znacznie ograniczyć nakład pracy, należy je zaimplementować w formie aplikacji internetowej.

Wieloplatformowość nie ogranicza się jednakże wyłącznie do funkcjonowania aplikacji na określonym zbiorze systemów operacyjnych. Każdy z wyżej wymienionych systemów jest przeznaczony na użytek w klasycznych komputerach o dużych wyświetlaczach albo urządzeniach mobilnych o wyświetlaczach, których przekątna rzadko przekracza 10 cali. Korzystanie z aplikacji, której interfejs graficzny został przystosowany do dużych ekranów na urządzeniu mobilnym, takim jak telefon komórkowy, wiąże się ze znacznym obniżeniem komfortu i wydajności pracy. Aby tego uniknąć autor postanowił, że wygląd oraz układ komponentów graficznych aplikacji internetowej będzie się samodzielnie dopasowywać do parametrów urządzenia użytkownika.

3.4.3. Wsparcie dla telepracy i bezpieczeństwo danych

Naturalnie sama forma aplikacji internetowej ułatwia prowadzenie telepracy, jednakże wiąże się z tym wiele wyzwań związanych z bezpieczeństwem danych oraz kontrolą jakości i wydajności pracy konsultanta.

Podstawowym zagadnieniem jest uwierzytelnianie oraz odpowiedni przydział praw poszczególnym użytkownikom. Autor podzielił użytkowników systemu na 3 klasy:

- Marketer – ma bezpośredni kontakt z klientem, pracuje w terenie, nie posiada dostępu do pełnej bazy klientów, może przeglądać dane tylko tych klientów, którzy zostali przydzieleni mu przez telemarketera
- TeleMarketer – posiada dostęp do pełnej listy klientów którą może przeglądać jak i edytować, lecz nie usuwać, wykonuje połączenia telefoniczne z nimi, przydziela klientów poszczególnym marketerom

- Admin – dysponuje uprawnieniami telemarketera, jednakże dodatkowo posiada pełnię władzy nad klientami, użytkownikami, ma możliwość przeglądania ich wydajności i podjętych akcji, a także modyfikację aktualnie przeprowadzanych ankiet oraz eksport ich wyników do pliku .xls w formacie zgodnym z IBM SPSS.

Użytkownik w celu zalogowania będzie zobowiązany podać w formularzu adres mailowy, hasło oraz wyrazić zgodę na pobranie jego współrzędnych geograficznych.

Aby ustrzec dane przed przechwyceniem przez osoby niepowołane, działania programisty nie mogą ograniczyć się jedynie do skutecznego mechanizmu autentykacji. Należy również uodpornić aplikację na mniej wyrafinowane ataki ze strony samych pracowników, jak choćby eksport danych poprzez kopiowanie danych widocznych na ekranie komputera pracownika. Autor podjął zatem decyzję o uniemożliwieniu zaznaczenia nieedytowalnej treści w interfejsie graficznym aplikacji.

Kontrola nad wydajnością pracy pracownika może być realizowana poprzez prezentację administratorowi wykresu zależności pomiędzy przedziałem czasu a liczbą wykonanych w nim rozmów telefonicznych.

Z kolei jednym z parametrów rzetelności pracownika może być miejsce w którym realizuje on swoje obowiązki. Przykładowo pracownik logujący się do systemu na wyjeździe wakacyjnym, może w mniej skuteczny sposób wykonywać powierzone mu obowiązki. Autor uznał, że najbardziej dogodnym zobrazowaniem miejsc logowania podwładnego jest mapa.

3.4.4 Uniezależnienie od architektury sprzętowej placówki przedsiębiorstwa

Jednym z kluczowych problemów leżących u podstaw niniejszej pracy jest uniezależnienie systemu od sprzętowej architektury telefonicznej. W tym celu należy skorzystać z operatora telefonii internetowej, na którym to spoczywa obowiązek dbania o dostępność usługi. Usługa ta jest dostępna wszędzie tam gdzie użytkownik posiada połączenie z internetem. Proponowane rozwiązanie musi jednakże udostępniać w/w usługę poprzez przeglądarkę internetową aby w pełni wesprzeć telepracę, a jej interfejs graficzny musi być responsywny w celu komfortowego użytkowania na urządzeniach o mniejszym ekranie. Interfejs ma także udostępniać przyciski kontrolujące stan połączenia.

Skorzystanie z zewnętrznej usługi telefonii odciąża serwery na których działa aplikacja dla call center, tym samym, zwiększając wydajność systemu przy utrzymaniu niższych kosztów, a także przyspiesza powrót to pełnej gotowości w przypadku awarii.

3.4.5. Audyt pracownika

Wraz z rozpoczęciem wdrażania HTML5 uzyskanie współrzędnych geograficznych użytkownika zostało znacznie uproszczone. Odnalezienie pozycji jest możliwe poprzez szereg dostępnych usług lokalizacyjnych świadczonych np. przez firmę Google, co zostało rozwinięte w rozdziale 4.15. Dane te wraz z adresem IP użytkownika i czasem logowania zostaną umieszczone w bazie danych w chwili pomyślnej autentykacji użytkownika. Autor postanowił przedstawić zebrane dane lokalizacyjne za pomocą interaktywnej mapy z możliwością powiększania i pomniejszania, co jest stosunkowo proste do osiągnięcia dzięki popularyzacji takich usług jak Google Maps API lub Mapy Bing

Autor zdecydował o mierzeniu wydajności pracownika, miarą ilości połączeń w jednogodzinnych przedziałów czasu i prezentować je w formie wykresu słupkowego.

Rejestracją rozmów autor postanowił obciążyć usługodawcę telefonii internetowej, co znacząco odciąża serwery na których funkcjonuje tworzony system, a także urządzenie oraz sieć komputerową użytkownika systemu, która nie jest zmuszona o przesyłanie dodatkowych danych audio.

3.4.6. Interfejs programistyczny

Ważnym aspektem każdego systemu jest możliwość jego rozwoju poprzez użytkowników, którzy stają się jednocześnie deweloperami. Może to skutkować zwiększeniem jego popularności i przyczyniać się do powstawania kolejnych udoskonalonych wersji, wykorzystujących kreatywność i sugestie odbiorców[8]. Jest to uniwersalna prawda, którą można zaobserwować nie tylko na przykładzie aplikacji biznesowych, lecz również gier komputerowych, czy systemów operacyjnych(Linux). Przygotowanie takiego rozwiązania zwiększy jego konkurencyjność.

Autor zdecydował o wytworzeniu API umożliwiającego korzystanie z bazy danych prototypu poprzez odpowiedni język zapytań. Aplikacja ma za zadanie zwracać wyspecyfikowany przez użytkownika zbiór rekordów w formie tekstowej przy pomocy wybranego protokołu internetowego opisanego w rozdziale 4.10.

4. Wykorzystane technologie

Autor wprowadza przedstawia główne technologie i koncepcje, okazały się istotnymi składnikami w prezentowanej propozycji rozwiązania przedstawionego problemu.

4.1. Platforma .NET i język C#

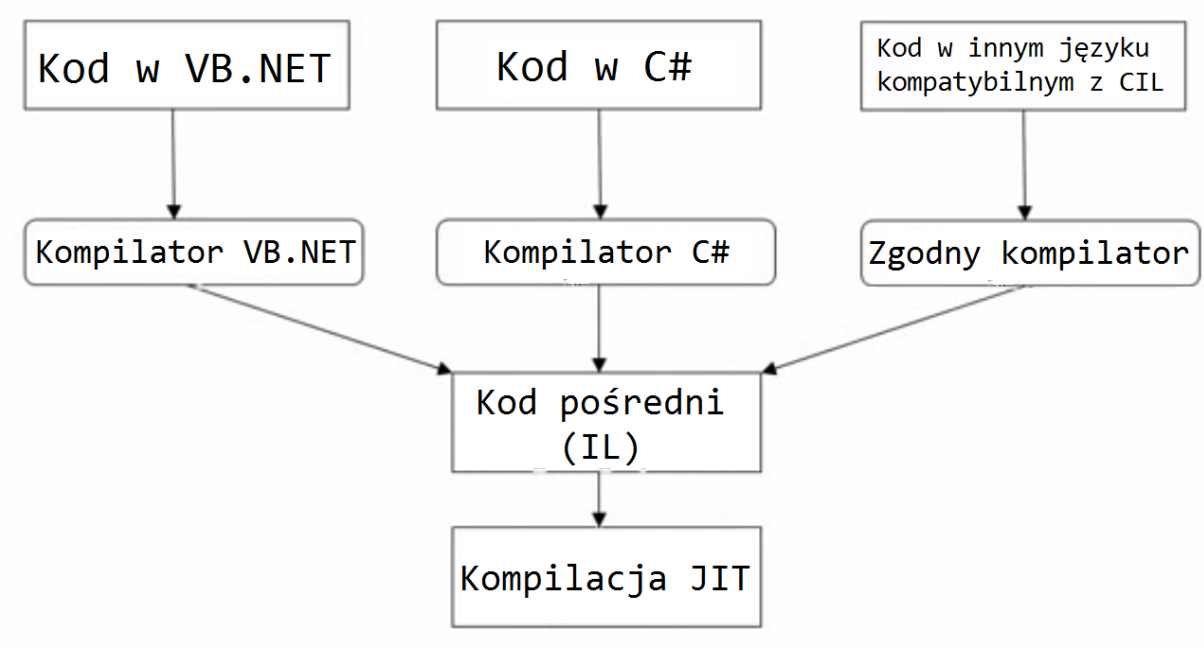
C# jest wieloparadygmatowym językiem programowania zgodnym ze standardem ECMA-334. C# jest oparty na językach C, C++ oraz Java[2], z tego też powodu współdzieli z nimi znaczną część składni oraz koncepcji. C# wspiera przede wszystkim programowanie obiektowe, jednakże również pozwala na programowanie funkcyjne, komponentowe oraz oczywiście imperatywne. Język ten posiada silną kontrolę typów, automatyczne zarządzanie dynamicznie przydzieloną pamięcią oraz wspiera dziedziczenie proste i rozłączne podobnie jak Java, dysponuje mechanizmem właściwości tak jak język C++ oraz CECHA WSPOLNA Z C TUTAJ. C# odznacza się jednolitym systemem typów(Unified Type System) nazwanym tutaj Common Type System(CTS). Oznacza to, że korzeniem hierarchii klas jest klasa System.Object, lecz w przeciwieństwie do Javy również typy proste są obiektami, co rozwiązuje między innymi problem wykorzystania klas opakowujących do parametryzowania kolekcji i usuwa tym samym potrzebę posiadania mechanizmu auto-boxingu i unboxingu. Podobnie do języka Java, C# jest językiem wieloplatformowym, oznacza to, że raz napisana aplikacja powinna funkcjonować na urządzeniach różnych klas wykorzystujących wielorakie systemu operacyjne.

Język C# można wykorzystać do tworzenia aplikacji pod:

- telefony, tablety, komputery wykorzystujące system Windows
- system iOS, poprzez Xamarin.iOS
- system Android, poprzez Xamarin.Android
- system Linux, wykorzystując Mono

Język C# jest częścią platformy .NET, która pozwala na tworzenie oprogramowania w: Visual C++, Visual C#, Visual Basic, JScript, F# i kilkudziesięciu innych językach. Jest to możliwe dzięki specyfikacji CLI (Common Language Interface). Kod napisany w dowolnym z w/w języków jest wstępnie kompilowany do kodu pośredniego (CIL Common Intermediate Language) poprzez kompilator dedykowany dla danego języka programowania. Jest to ostatni etap na którym otrzymany kod jest zrozumiały dla człowieka. Następnym etapem jest kompilacja kodu pośredniego poprzez CLR(Common Language Runtime), który przetwarza go na kod maszynowy. Jest to tzw. Kompilacja JIT (Just in Time), oznacza to, że dany fragment kodu jest kompilowany z kodu pośredniego do maszynowego

wówczas, gdy ma być on wykonany. Opisany proces można zaobserwować na rysunku 3. Standard CLI jest jedną z cech, dzięki której .NET zdobyło znaczną popularność, co zaowocowało wytworzeniem wielu języków kompatybilnych z platformą .NET z inicjatyw niepochodzących od wytwórcy .NET, firmy Microsoft.



Rysunek 3. Drzewo obrazujące proces uruchomienia programu

Platforma .NET została wybrana przez autora również ze względu na jej popularność pozwalającą na łatwy rozwój proponowanego rozwiązania przez innych programistów oraz bardzo dynamiczny rozwój .NET ze strony producenta, czyli firmy Microsoft.

4.2. Visual Studio 2013

Dziewiąta wersja zintegrowanego środowiska programistycznego (Integrated Development Enviroment, IDE) będącego kompletnym narzędziem wspierającym tworzenie oprogramowania konsolowego jak i z graficznym interfejsem użytkownika pod platformę .NET na urządzenia desktopowe i mobilne, jego publikowanie, zarządzanie bazami danych, pracę zespołową nad jednym projektem, analizę syntaktyczną wytworzonego kodu oraz generowanie szeregu metryk oprogramowania pomagających określić jego ogólną jakość. Visual Studio jest produktem firmy Microsoft i naturalnym wyborem w przypadku kreacji przy użyciu .NET.

4.3. HTML5

Hipertekstowy język znaczników(Hypertext Markup Language) zdefiniowany w standardowym uogólnionym języku znaczników (Standard Generalized Markup Language lub Goldfarb Mosher Lorie, SGML) mający swój początek w roku 1991 w formie szkicu, jednakże pierwszą ustandaryzowaną poprzez Inernet Engineering Task Force wersją o oficjalnej specyfikacji był HTML 2.0 opublikowany w roku 1995, a ostatnią zgodną z SGML HTML 4.01 z roku 2001. Wyżej wymienione standardy języka (od HTML po HTML 4.01) pozwalały na opisanie struktury informacji zawartych na stronie internetowej, przy pomocy kilkudziesięciu znaczników o różnej semantyce zawartych w nawiasach kątowych.

HTML5 jest znacznie szerszym pojęciem od swoich poprzedników, gdyż nie ogranicza się jedynie do opisu struktury dokumentu. Upraszcza problemy, które przez lata programiści byli zmuszeniu rozwiązywać żmudnymi sposobami. Standard konsorcjum W3C wymaga od przeglądarek internetowych natywnego wsparcia dla audio oraz video, walidacji danych, pól edycyjnych poprzez wbudowane edytory (np. dla pól przechowujących daty), a także rozszerza język znaczników stawiając znacznie większy nacisk na ich znaczenie semantyczne, wymusza także istnienie API m.in. do:

- geolokalizacji
- funkcjonalności drag&drop
- pamięci lokalnej oraz pamięci sesji
- dwukierunkowej aplikacji z serwerem w czasie rzeczywistym niezrywanej po załadowaniu strony
- WebGL (wersja OpenGL dla przeglądarek internetowych)

oraz wielu innych również realizowanych za pomocą języka JavaScript. Standard HTML5 nadal jest rozwijany i jego kolejna odsłona 5.1 jest zapowiadana na rok 2016.

4.4. CSS3 i responsywny rozkład okna

Zanim wprowadzono kaskadowe arkusze stylów każda z przeglądarek internetowych posiadała własny zestaw reguł opisujący jak mają wyglądać informacje przez nią prezentowane. Przyczyną takiej sytuacji była walka przez producentów przeglądarek o udział w rynku, który spychała na dalszy plan zgodność z ogólnym standardem. Sytuacja zaczęła się powoli zmieniać, gdy roku 1996 wprowadzono CSS 1.0, natomiast dopiero w roku 2004 wraz z wprowadzeniem szkicu wersji 2.1 deweloperzy przegląda-

rek jednogłośnie podjęli starania implementacji standardu. Ten nigdy nie został ukończony, gdyż wraz z pojawieniem się nowych urządzeń wspierających dotyk oraz posiadających mniejsze ekrany, to jest smartphonów i tabletów powstała potrzeba utworzenia nowszego standardu CSS3.

CSS3 oraz HTML5 oficjalnie są od siebie niezależnymi projektami, jednakże w praktyce są często nierozłączne. CSS3 umożliwia dokładny opis tego jak zawartość znacznika HTML ma być wyświetlana w oknie przeglądarki zależnie od jego nazwy, klasy lub identyfikatora, wielkości ekranu na którym występuje, relacji do innych komponentów. Dzięki CSS3 strony mogą mieć atrakcyjny, kolorowy, animowany interfejs dopasowujący się do urządzenia na którym ogląda je użytkownik. Podobnie jak HTML5 standard ten podlega ciągłym modyfikacjom, a jego dokumentacja ze względu na jego rozmiary została podzielona na niezależne moduły.

W zależności od przekątnej ekranu domyślny rozkład elementów w oknie przeglądarki przeznaczony dla pełnowymiarowego monitora[2] może być nieczytelny oraz niewygodny w obsłudze na urządzeniach mobilnych o niższej rozdzielczości i wielkości ekranu. Problem ten można rozwiązać przy pomocy responsywnego układu strony internetowej. Oznacza to, że układ, a także wygląd elementów będzie się dopasowywać do rozmiaru okna roboczego. Osiągnięcie tego celu jest możliwe od momentu wprowadzenia arkuszy stylów CSS3. Autor w rozwiązaniu skorzystał z arkusza CSS Bootstrap 3.0.0 udostępnionego na zasadach licencji MIT, a więc nieograniczonego prawa do używania, modyfikowania, kopiowania oraz rozpowszechniania.

4.5. Google Chrome

Autor w propozycji rozwiązania umieszczonej w rozdziale 3. zwrócił uwagę na konieczność implementacji systemu jako aplikacji internetowej. Współczesne przeglądarki obsługują opisane w rozdziale 4. HTML5 oraz CSS3, zatem są programem zdolnym do wyświetlania interaktywnego graficznego interfejsu użytkownika, opisanego w sposób jednakowy niezależnie od urządzenia.

Według źródeł World Wide Web Consortium 1 lipca 2015 roku 64.8% rynku jest opanowanych przez przeglądarkę Google Chrome[5]. Na każdą z wymienionych w rozdziale 3.1 platform istnieje kompatybilna wersja dominującej przeglądarki internetowej. Naturalnym więc wydaje się wybór najpopularniejszego rozwiązania, gdyż jest ono najbardziej uniwersalne dla potencjalnego odbiorcy.

[2] Na potrzeby pracy autor zakłada, że pełnowymiarowy monitor to taki, którego przekątna przekracza 10 cali.

4.6. JavaScript

JavaScript, znany na początku jako LiveScript, został wytworzony przez firmę Netscape na potrzeby przeglądarki internetowej tej firmy i mimo mylącej nazwy współcześnie nie jest w jakiejkolwiek relacji z językiem Java, a jego nazwa jest zabiegiem marketingowym[16], na który zgodę wyraziło Sun Microsystems. Nie współdzieli z nim również wielu założeń, a w obecnej wersji implementuje standard ECMAScript5. Jest to natomiast język nierozłącznie związany z technologiami internetowymi a dokładnie stronami internetowym i najczęściej wykorzystywany do wykonywania operacji po stronie klienta przez przeglądarkę internetową, czyli do frontendu.

Znaczną różnicą w stosunku do języków programowania popularyzowanych na PJATK, to jest języka Java, jest dynamiczne typowanie zmiennych, co oznacza że typ danych przypisanych do danej zmiennej może się zmieniać w trakcie wykonywania programu. Znaczną różnicą jest także brak dyktowania widoczności zmiennych poprzez bloki, zamiast tego deklaracje zmiennych wynoszone na początek kodu.

Najważniejszą jednakże funkcjonalnością języka JavaScript jest precyzyjna kontrola nad drzewem DOM pozwalająca na zmianę zawartości strony internetowej bez potrzeby jej przeładowywania, co umożliwia dynamiczną interakcję z użytkownikiem, w idealnym przypadku przypominającą aplikacje desktopowe. Polepsza to immersję użytkownika i zwiększa komfort pracy. Wszystkie API wyszczególnione w propozycji W3C dla HTML5 są obsługiwane poprzez język JavaScript.

4.7. jQuery

Jest to biblioteka dla języka JavaScript, która zdominowała scenę internetową w 2010 roku zostając najpopularniejszą nakładką na język JavaScript. Jej zadaniem jest ułatwienie trawersowania po dokumencie HTML i manipulacji DOM, obsługi zdarzeń, animacji oraz zapytań AJAX umożliwiających pobranie dodatkowych danych z serwera. jQuery zwiększa przejrzystość kodu i ułatwia tworzenie aplikacji internetowych początkującym programistom, co przeważyło w walce z innymi bibliotekami skierowanych do bardziej zaawansowanych odbiorców. Ważnym aspektem jest rozmiar biblioteki, jest to tylko 80kB, a także wsparcie ze strony każdej z popularnych przeglądarek. Wadą jest natomiast jej prędkość, która w skrajnych wypadkach może być kilkadziesiąt razy mniejsza od natywnego kodu w JavieScript. Przykładowo pobranie elementów HTML o zadanej klasie CSS przy użyciu natywnej metody `getElementsByClassName()` trwa 75ms na maszynie testowej, lecz użycie jQuery spo-

walniało tę operację do 3437ms.[17] Autor w listingach od 1. do 4. przytacza kilka przykładów porównawczych, aby wyklarować różnice pomiędzy JavaScript a jQuery.

Listing 1. Pobranie elementów o klasie CSS wynoszącej .comment w języku JavaScript:

```
var c = document.getElementsByClassName("comment");
```

Listing 2. Pobranie elementów o klasie CSS wynoszącej `.comment` przy użyciu jQuery:

```
var c = $(".comment");
```

Listing 3. Zmiana koloru tła po załadowaniu dokumentu HTML w języku JavaScript:

```
function changeBackground(color) {
        Document.body.style.background = color;
}

Onload=”changeBackground (‘red’);
```

Listing 4. Zmiana koloru tła po załadowaniu dokumentu HTML przy użyciu jQuery

```
$(‘body’).css(‘background’, ‘green’);
```

Można zaobserwować że, typowe operacje jak wybranie danego elementu z drzewa DOM, bądź rejestracja zdarzenia są łatwiejsze do wyrażenia za pomocą jQuery, zatem biblioteka ta zwiększa ekspresyjność JavaScript.

4.8. AJAX

Asynchroniczny JavaScript oraz XML. Technika wspierana przez JavaScript, z której korzystanie ułatwia jQuery i szereg innych bibliotek. Jej zadaniem jest asynchroniczna wymiana informacji pomiędzy klientem oraz serwerem bez konieczności przeładowywania strony HTML. Transmisja jest rozpoczynana po stronie klienta i z reguły jest reakcją na akcję ze strony użytkownika aplikacji. AJAX nie jest jednakże ściśle jednorodną techniką, posiada wiele odmian. Składają się na nie:

- HTML (lub XHTML) i plik CSS do prezentacji wyników
- dostęp do DOM
- XML lub znacznie częściej JSON(wariant AJAJ) lub inny format wymiany danych
- klasa XMLHttpRequest do asynchronicznej komunikacji

- JavaScript lub jego nakładka jak np. jQuery do współpracy z powyższymi technologiami

Technika ta jest wykorzystywana niemalże w każdej stronie internetowej, a w szczególności w jednostronicowych aplikacjach internetowych, których jednym z zadań jest symulacja zachowania aplikacji desktopowych.

4.9. XML

Opublikowany w 1998 roku, jeden z dwóch najpopularniejszych formatów wymiany danych, którego początek wywodzi się z języka SGML wynalezionego pod koniec lat 60-tych na zlecenie IBM na potrzeby integracji danych pomiędzy trzema aplikacjami przez Charlesa Godfarba ,Eda Moshera i Raya Lorie.[18] Z potrzeby oszczędzania każdego bitu pamięci składnia SGML była bardzo rozbudowana i posiadała wiele skrótów. XML jest uproszczeniem, a zatem podzbiorem SGML wytworzonym przez Tima Braya. Jedną ze znaczących różnic jest surowa składnia:

- każdy otwierający znacznik musi mieć odpowiadający mu znacznik zamykający
- wszystkie atrybuty muszą posiadać wartości, a te wartości muszą znajdować się w cudzysłowach
- język jest case-sensitive zatem znaczenie ma wielkość znaków
- występowanie pustych znaczników domykających samych siebie oraz nie zawierających żadnych elementów

Język ten jest z reguły używany do

- serializacji danych, często w celu późniejszego przesyłania ich pomiędzy klientem i serwerem
- opisu graficznych interfejsów użytkownika w sposób deklaratywny
- opisu konfiguracji aplikacji
- opisu tabel i relacji w relacyjnej bazie danych koniecznego do mapowania klas i asocjacji

W listingu 5. autor przytacza opisu menu restauracji przy użyciu XML[3] w celu przybliżenia składni języka.

Listing 5. Opis menu potraw z wykorzystaniem XML[13]

```
<?xml version="1.0" encoding="UTF-8"?>
<menu>
	<dish>
		<name>Scrambled eggs</name>
		<price>1.20</price>
		<description>Mixed well with tomatoes and bacon</description>
```

```
            <calories>400</calories>
      </dish>
      […]
      <dish>
            <name>Krispy Kreme donut</name>
            <price>0.95</price>
            <description>Guilty joy</description>
            <calories>500</calories>
      </dish>
</menu>
```

4.10. JSON

JavaScript Object Notation jest to format wymiany danych pomiędzy klientem a serwerem lub pomiędzy różnymi platformami programistycznymi. Idealnie służy do przekazywania list krotek (klucz, wartość). Istnieje wiele bibliotek dla serializacji danych po stronie serwera i deserializacji po stronie klienta, a samo przesyłanie danych jest realizowane poprzez protokół HTTP, zatem wymiana danych jest uniezależniona od platformy producenta i konsumenta. W odróżnieniu do XML nie jest to język znaczników a jego struktura jego składnia jest prosta do opanowania oraz ustandaryzowana poprzez ECMA-404, składa się ze operatorów: {}:”

Format JSON zamiast XML jest preferowany gdy:

- wiadomości nie muszą być walidowane
- wiadomości nie są transformowane do dokumentów
- użytkownikowi zależy na mniejszym rozmiarze danych oraz krótszym czasie parsowania poprzez uniknięcie redundantnych dla danego przypadku konstrukcji XML

Przykład opisu książek o atrybutach: `id, language, edition, author` autor przedstawia w listingu 6.

Listing 6. Opis listy książek przy użyciu JavaScript Object Notation

```
{
    "book": [
    {
       "id":"1",
       "language": "COBOL",
       "edition": "1st",
       "author": "John Ancient"
    },
    {
       "id":"2",
       "language": "Objective-C",
       "edition": "2nd",
       "author": "Donald Apple"
    }]
}
```

4.11. OData

Protokół sieciowy opracowany przez firmę Microsoft służący do pobierania danych w formacie JSON lub XML transferowanych poprzez protokół sieciowy HTTP. Strona zapewniająca dane jest w OData nazywana producentem, natomiast ich odbiorca konsumentem. Protokół ten umożliwia rozszerzenie aplikacji internetowej o serwis API umożliwiający pobieranie danych przez aplikacje trzecie poprzez stosowny URI. Aby otrzymać żądany sposób należy wykorzystać język zapytań OData.

Prefiks każdego zapytania stanowi adres serwisu `http://adresAplikacji.domena/odata` następująca ścieżka zasobu, przykładowo /Person(2)? a następnie opcjonalne operatory zapytań wskazujące na wybrane atrybuty lub kolejność rekordów np. `$select=FirstName,LastName,Address` zwrócone dane są w formacie JSON lub XML. OData wykorzystuje podlegający model danych aplikacji `DbContext` do dostępu danych.

Wyniki mogą być odpowiednio zawężane, rozszerzane, filtrowane lub sortowane w zależności operatorów użytych w URL, są to odpowiednio: `$select, $expand, $filter, $orderby` i wiele innych.

4.12. Wzorzec projektowy Model-View-Controller

Jeden z podstawowych wzorców architektonicznych zorientowanych obiektowo autorstwa firmy XEROX, wykorzystywany w tworzeniu oprogramowania. Jego głównym celem jest zachowanie porząd-

ku i podział obowiązków w implementacji poprzez oddzielenie warstwy prezentacji od logiki biznesowej aplikacji.

MVC został zaprojektowany w roku 1979 i przez lata ewoluował do wielorakich wariantów. Wersja wykorzystana w rozwiązaniu problemu postawionego w pracy dzieli się on na[10]:

- Model – odpowiada za klasy implementujące logikę biznesową aplikacji oraz sposób przechowywania danych
- View (widok) – warstwę prezentacji, która może udostępniać użytkownikowi wybraną część funkcjonalności aplikacji w zależności od jego uprawnień. Widok ma bezpośredni dostęp do Modelu oraz dostęp do metod kontrolerów
- Controller – warstwę pośredniczącą pomiędzy modelem a widokiem, na podstawie akcji użytkownika przechwyconych z widoku aktualizuje model, a także odświeża widoki

Podejście to ma między innymi ograniczyć nakład pracy w przypadku aktualizacji interfejsu użytkownika aplikacji, który często ewoluuje szybciej od jej bazowej funkcjonalności, jak również pozwala na stworzenie wielu interfejsów użytkownika bez konieczności zmiany modelu dla różnych odbiorców z uwzględnieniem ich praw lub urządzeń z których korzystają.

4.13. Platforma ASP.NET-MVC 5

Platforma do tworzenia aplikacji internetowych produkcji firmy Microsoft. Jej poprzednikami są Classic Active Server Pages oraz ASP.NET. Pierwsza wersja stabilna została wydana w roku 2009 a wykorzystana w rozwiązaniu problemu postawionego w pracy wersja 5.2.2 została wydana sierpniu 2014 roku. Zatem jest to stosunkowo młoda technologia, która jednakże w krótkim czasie uzyskała dużą dojrzałość.

W odróżnieniu od swoich poprzedników nie oferuje kreatorów WYSIWYG, a tworzenie aplikacji w ASP.NET-MVC nie przypomina programowania aplikacji desktopowych, zatem próg wejścia w technologię jest znacznie wyższy w stosunku do ASP.NET. Można z tego wysnuć prawidłowy wniosek, że mimo podobieństw w nazewnictwie są to zupełnie dwie różne platformy.

Mimo pozornych trudności platforma ASP.NET-MVC nie tylko umożliwia, a wymusza zachowanie porządku w tworzonym oprogramowaniu poprzez podział aplikacji internetowej na kompozycję trzech warstw logicznych: model, widok, kontroler opisany poprzednim podrozdziale, jednakże należy ten opis rozwinąć w kontekście ASP.NET-MVC.

Struktura aplikacji jest podzielona wiele folderów w tym najważniejsze są trzy z nich. Pierwszy folder `Models` zawiera:

- Klasy których instancje maja być zapisane w bazie danych. W każdej klasie możliwe jest wykorzystanie adnotacji do oznaczenia sposobu mapowania danego atrybutu na kolumnę, bądź klasy na tabelę jak i również wskazówki dotyczące sposobu wyświetlania oraz edycji atrybutu w widoku
- Klasy definiujące wykorzystywane połączenie z bazą danych, informacje które klasy mają być mapowane do bazy danych oraz jej podstawowe konwencje np. automatycznego nazewnictwa tabel czy sposobu ładowania rekordów z bazy

Drugi folder `Controllers` zawiera:

- Klasy zawierające metody zwane metodami akcji (ang. Action methods). Nazwa kontrolera skonkatenowana z nazwą metody tworzy link, który wybrany prze klienta powoduje wywołanie danej metody. Przykładowo link `/Person/Details` spowoduje wywołanie metody `Details` z kontrolera o nazwie `PersonController`, metoda akcji może również posiadać parametry, które są pobierane z query string. Metoda ta może pobierać rekordy z bazy danych oraz zwracać jeden z następujących wyników[11]:
    - `ViewResult` – renderuje widok skojarzony z kontrolerem
    - `PartialViewResult` – zwraca widok częściowy skojarzony z kontrolerem, do obu wyników można przekazywać referencję do obiektów
    - `EmptyResult` – pusta odpowiedź
    - `RedirectResult` – przeprowadza przekierowanie pod inny URL i tym samym do innej metody akcji
    - `JsonResult` – serializuje dany obiekt do postaci Java Object Notation
    - `JavaScriptResult` – zwraca kod JavaScript który ma być wykonany po stronie klienta
    - `FileContentResult, FileStreamResult, FilePathResult` – zwracają pliki do klienta

Trzeci folder `Views` zawiera pliki `.cshtml` parsowane poprzez silnik Razor, składające się z jednocześnie przeplatającego się kodu HTML, C#, JavaScript lub jego rozszerzeń. W czasie wykonywania programu widok jest parsowany, podczas tego procesu wszystkie konstrukcje w języku C# są zamieniane na HTML i w tej postaci wysyłany do użytkownika. Każdy widok jest skojarzony z metodą

akcji której jest wynikiem i znajduje się w pod ścieżką `Views/NazwaKontrolera/NazwaAkcji`. Metoda akcji może przekazywać dane do zwracanego przez nią widoku poprzez:

- `ViewBag` – dostępny po stronie metody akcji oraz widoku obiekt dynamiczny, to jest taki którego właściwości są dodawane w czasie wykonywania programu
- `ViewData` – wykorzystywany głównie pojawieniem się obiektów dynamicznych wraz z C# 4.0, jest słownikiem będącym zbiorem mapowań `klucz -> wartość` , w przeciwieństwie do `ViewBag` wymaga rzutowania w przypadku pobierania ze słownika referencji do obiektów typów złożonych oraz sprawdzania istnienia wartości przypisanej do klucza w celu uniknięcia wyjątku `NullReferenceException`[12]
- `Model` – referencja do obiektu przekazywana jako argument do widoku
- `ViewModel` – klasa zdefiniowana tylko na potrzeby przekazania danych do widoku, akomodująca jako właściwości dane które programista chce przekazać do widoku
- `Tuple` – krotka pozwalająca przechowywać referencje do obiektów różnych typów, umożliwiająca uniknięcie tworzenia wielu `ViewModeli` tylko na potrzeby pojedynczego ich wykorzystania.[14]

4.14. Object Relationship Mapping I Entity Framework 6

Oprogramowanie ORM umożliwia ograniczenie niedogodności wynikającej z niezgodności impedancji świata obiektowego i relacyjnego, mapując klasy, atrybuty, asocjacje odpowiednio na tabele, kolumny, relacje w relacyjnej bazie danych. Umożliwia tym samym komfortową współprace programu napisanego w języku obiektowym z zapleczem bazodanowym.

Entity Framework 6 jest jednym z najbardziej zaawansowanych ORM dostępnych na rynku, wyprodukowany został przez firmę Microsoft. EF tworzy w sposób automatyczny Entity Data Model na podstawie definicji wybranych przez programistę klas. Entity Framework tak jak ASP.NET-MVC przewodzi założenie „konwencja nad konfigurację", co oznacza że podczas analizy zadeklarowanych w klasie właściwości wybrane nazwy i konstrukcje mają domyślne semantyczne znaczenie, tak na przykład właściwość o nazwie Id jest domyślnie kluczem głównym. Asocjacje do innych klas są mapowane są na relacje 1-1, 1-*, *-* w zależności czy mamy do czynienia z atrybutami pojedynczymi, czy wielokrotnymi. ORM przy mapowaniu bierze pod uwagę metadane w postaci adnotacji do właściwości. Adnotacje te pozwalają również nadpisać domyślne konwencje, a także nałożyć ograniczenia na atrybuty jak np. dodać górne ograniczenie długości napisu poprzez adnotację [MaxLength(250)]. Entity Framework domyślnie mapuje dziedziczenie w trybie Table per Hierarchy, dodając dyskryminator w tabeli. Oprogramowanie wspiera także wiele mechanizmów umożliwiających zwięk-

szenie wydajności pobierania rekordów z bazy danych w celu mapowania ich na obiekty, są to leniwe ładowanie, które opóźnia pobranie rekordu do momentu bezpośredniego odwołania się do niego w kodzie C#, oraz Fluent API pozwalające na bezpośrednie definiowanie reguł pobierania rekordów. Oczywiście aktualizacja, wstawianie rekordów i wiele innych operacji jest obsługiwane.

Entity Framework nie tylko pozwala na zautomatyzowane tworzenie bazy danych, lecz również na migracje pomiędzy kolejnymi jej wersjami, przy tym oprogramowanie chroni programistę przed utratą bądź rozspójnieniem danych należytymi ostrzeżeniami. Autor wybrał ten ORM ze względu na domyślne wsparcie dla ASP.NET-MVC oraz jako że jest to jedna z najbardziej dojrzałych bibliotek, która już wypiera swego poprzednika NHibernate. Świadczą o tym statystyki aktywności programistów mówiące o ilości tworzonych i aktualizowanych projektów w repozytoriach przy użyciu w/w technologii. Entity Framework odnotowuje znaczny wzrost popularności, a NHibernate odnotowuje przeciwną tendencję.[15]

4.15. Geolokalizacja poprzez sieć

Wraz z wdrażaniem HTML5 na scenie informatycznej wśród serwisów internetowych popularnym stało się precyzyjne ustalanie umiejscowienia użytkownika. Geolokalizacja została włączona przez W3C do specyfikacji HTML5, nie wyszczególniony został jednakże sposób implementacji tej funkcjonalności. Najbardziej popularną implementacją tej funkcjonalności jest Google Maps Geolocation API.

Gdy użytkownik natrafi na stronę internetową chcącą uzyskać informacje o jego współrzędnych zostanie poproszony o wyrażenie zgody, jeżeli ją wyrazi przeglądarka wyśle do usługi Google następujące parametry dla każdej będącej w zasięgu sieci WiFi:[19]

- adres MAC węzła sieci komputerowej
- siłę sygnału mierzonej jednostce dBm wyrażającej stosunek sygnału go szumu.
- czas który upłynął od wykrycia danej sieci WiFi
- SSID czyli identyfikator sieci WiFi

Przykładowe wartości dla pojedynczego węzła sieci bezprzewodowej przekazywane usłudze Google Maps Geolocation zostały przedstawione w listingu 7.

Listing 7. Informacje wysyłane do usługi Google Maps Geolocation API o jednym z dostępnych węzłów WiFi

```
"mac_address": "98-76-54-32-10-ba",
"signal_strength": 4,
"age": 0,
"SSID": "AccessPoint"
```

Na podstawie tych informacji będzie w stanie uzyskać lokalizację użytkownika. Dane o sieciach WiFi firma Google zbierała podczas przejazdów drogami publicznymi samochodami wykonującymi zdjęcia do usługi Street View dostępnej w Mapach Google. W niektórych państwach m.in. w Irlandii i Niemczech dane zbierane przez samochody Google powodowały kontrowersje i obawy organów państwowych do spraw ochrony danych, które następnie domagały się ich audytu. [9]

4.16. SIP oraz SIP Trunking

Session Initiation Protocol jest internetowym protokołem komunikacyjnym umożliwiającym wyrażanie chęci nawiązania połączenia audio lub video z innym urządzeniem posiadającym adres IP. Po nawiązaniu połączenia urządzenia przesyłają informacje za pomocą Real Time Transport Protocol.

SIP Trunking jest to natomiast usługa telefoniczna dostarczana przez dostawcę telefonii internetowej(ITSP) umożliwiająca wykonywanie wszelkich połączeń telefonicznych przy użyciu łączy internetowych. W odróżnieniu od klasycznej telefonii VoIP operator traktuje organizację jako całość i rozmowy w jej obszarze są darmowe, dodatkowo możliwe jest równoczesne podjęcie wielu rozmów z tego samego numeru, przykładowo w celu prowadzenia infolinii.

5. Przedstawienie implementacji prototypu

W poniższym rozdziale autor przedstawia sposób implementacji systemu zgodnego z założeniami przedstawionymi w rozdziale 3 z wykorzystaniem technologii i koncepcji opisanych w rozdziale 4.

5.1 Architektura prototypu

W niniejszym podrozdziale autor przedstawia na diagramie 1. architekturę opisanego prototypu proponowanego rozwiązania. Składają się na nią:

- silnik routingowy parsujący zapytanie przeglądarki internetowej użytkownika oraz odnajdujący właściwą metodę akcji kontrolera
- baza danych SQL przechowująca rekordy powstałe w wyniku mapowania obiektowo relacyjnego wybranych instancji klas
- kontrolery odpowiadające za wygenerowanie odpowiedzi na zapytanie HTTP przeglądarki:
    - `PersonController` – obsługuje akcje utworzenia, edycji, usunięcia klienta z bazy danych, a także umożliwia wygenerowanie tokena zezwalającego na inicjalizację połączenia głosowego z klientem
    - `AccountController` – umożliwia utworzenie, edycję, usunięcie profilu użytkownika systemu
    - kontrolery API OData – generujące odpowiedzi w formacie JSON na zapytania zgodne z protokołem OData
- modele odpowiadające za logikę biznesową i sposób przechowywania danych:
    - `ApplicationUser` – opisuje użytkownika aplikacji, atrybuty tej klasy przechowują informacje niezbędne do autentykacji użytkownika oraz dane na temat wykonanych połączeń
    - `Call` – reprezentuje rozmowę telefoniczną
    - `Person` – klasa ta składa się z atrybutów przechowujących dane osobowe klienta przedsiębiorstwa oraz informacje o przeprowadzonych z nim rozmowach
    - `ApplicationDbContext` – służy do interakcji z relacyjną bazą danych. Pobiera, zapisuje oraz uaktualnia instancje wybranych klas w bazie danych.
- widoki na podstawie których silnik Razor generuje dokument HTML. Dla każdej z klas, której obiekty przechowywane są w bazie danych dostępne są widoki:
    - `Create` – widok umożliwiający stworzenie instancji danej klasy poprzez uzupełnienie formularza
    - `Details` – wyświetla informacje o stanach atrybutów danego obiektu
    - `Edit` – umożliwia zmianę stanu atrybutów obiektu
    - `Delete` – umożliwia usunięcie obiektu
- Usługa telefonii internetowej Twilio realizująca połączenie głosowe klientem. Przeglądarka internetowa użytkownika i usługa Twilio w czasie rzeczywistym przesyłają strumień audio.

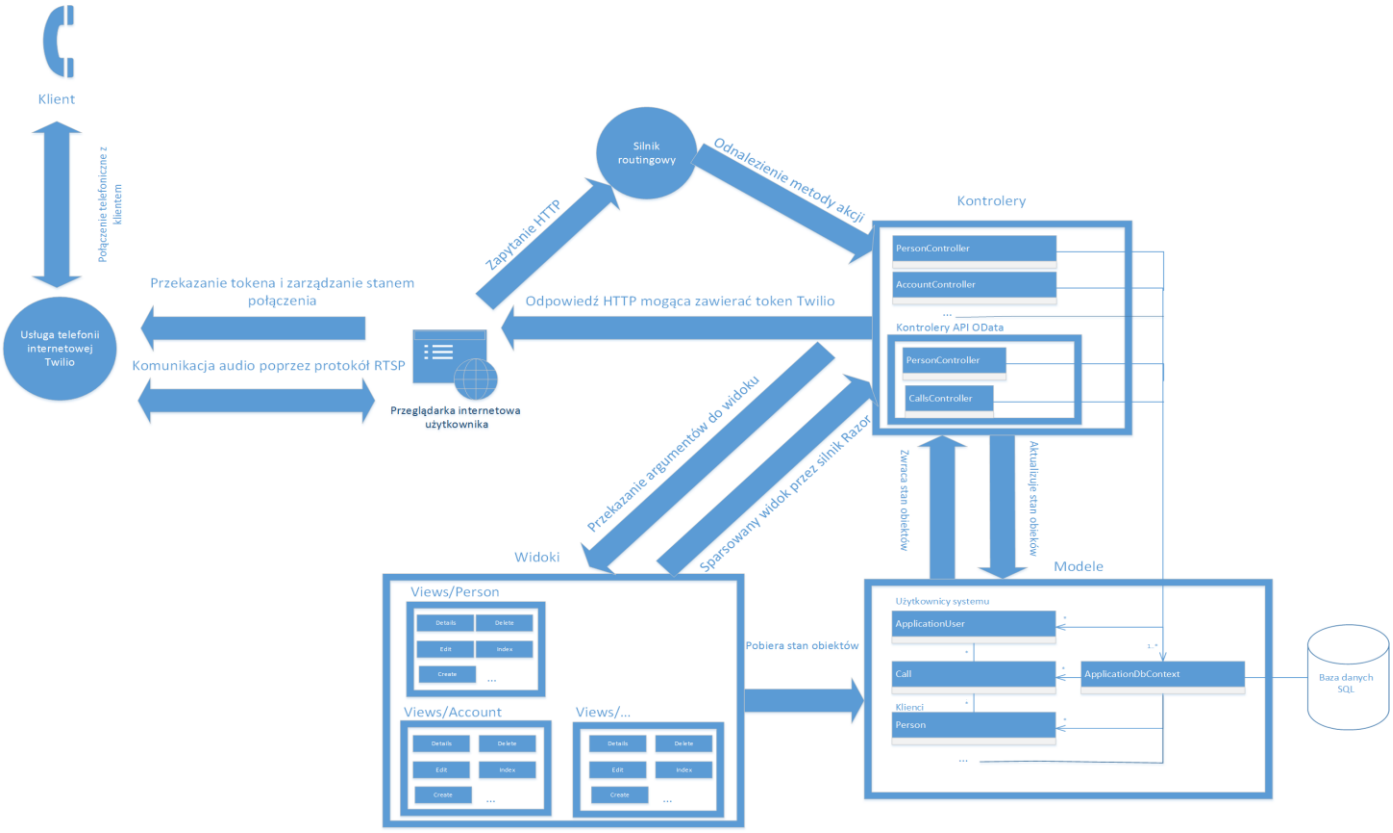
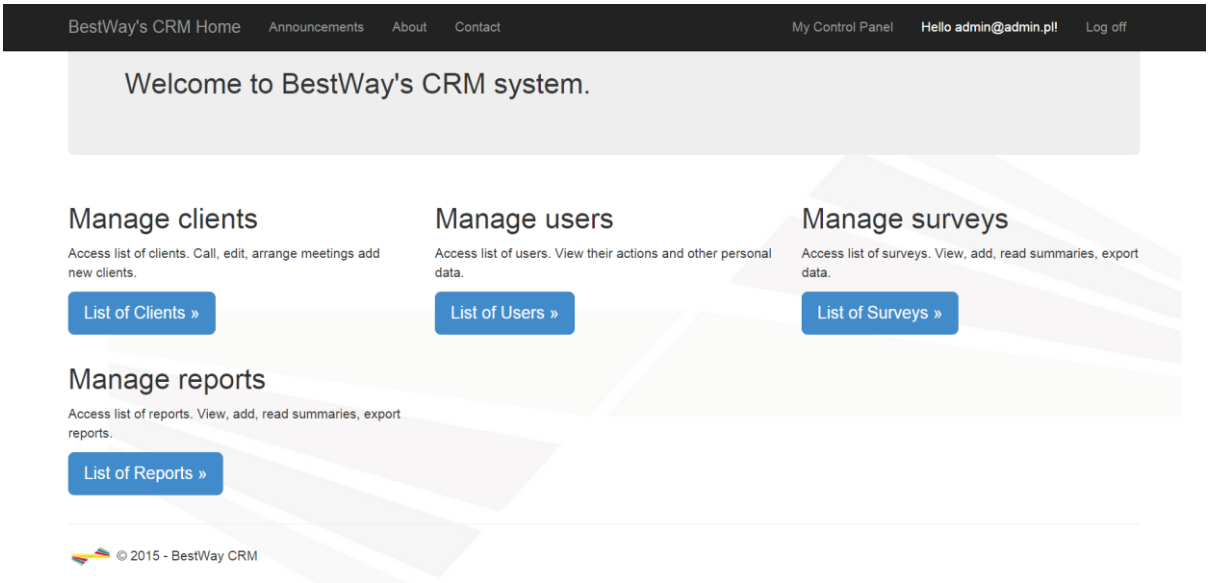


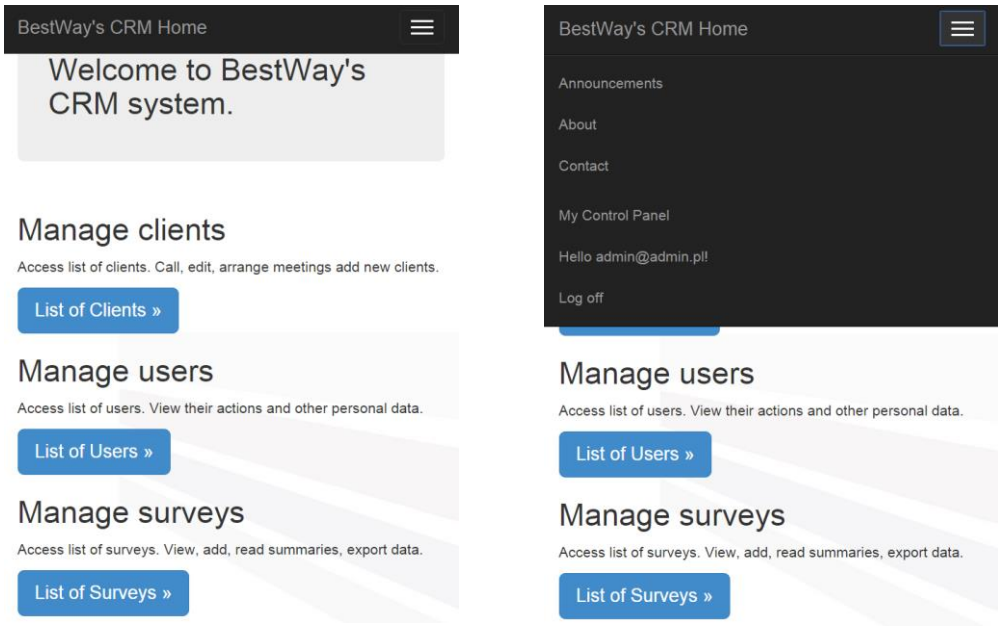
Diagram 1. Architektura prototypu rozwiązania

5.2. Interfejs użytkownika

W rozdziale 3.4.2. autor wskazał wymóg responsywnego układu strony internetowej szerzej opisanego w rozdziale 4.4. Dzięki użyciu odpowiednich klas Bootstrap w dokumencie `.cshtml` możliwe jest uzyskanie interfejsu użytkownika reagującego na wielkość i rozdzielczość posiadanego ekranu.



Rysunek 4. Rozkład interfejsu użytkownika uzyskany na ekranie Full HD o przekątnej 17”



Rysunki 5a. oraz 5b. Rozkład interfejsu użytkownika uzyskany na urządzeniu mobilnym

Rysunki 4. oraz 5a. i 5b. prezentują różnicę w układzie oraz wyglądzie poszczególnych elementów. W obu przypadkach pełna zawartość strony mieści się w oknie roboczym przeglądarki bez konieczności jej horyzontalnego przewijania. Warto zwrócić uwagę nie tylko na wertykalny ułożenie przycisków i etykiet na ekranie urządzenia mobilnego a również na pojawienie się animowanego i rozwijanego menu użytkownika. Zostało to uzyskane dzięki użyciu odpowiednich klas CSS oraz poprzez osadzenie każdej z podstron w widoku głównym `_Layout.cshtml` którego kod zawarty jest w listingu 8.

Listing 8. Przykład implementacji responsywnego układu strony

```
<!DOCTYPE html>
<html>
<head>
[...]
</head>
<body>
    <div id="background">
        <div class="navbar navbar-inverse navbar-fixed-top">
            <div class="container">
                <div class="navbar-header">
                    <button type="button" class="navbar-toggle" data-toggle="collapse"
                     data-target=".navbar-collapse">
                        <span class="icon-bar"></span>
                        <span class="icon-bar"></span>
                        <span class="icon-bar"></span>
                    </button>
                    @Html.ActionLink("BestWay's CRM Home", "Index", "Home", new { area =
                    "" }, new { @class = "navbar-brand" })
                </div>
                <div class="navbar-collapse collapse">
```

```
                    <ul class="nav navbar-nav">
                        <li>@Html.ActionLink("Announcements", "Index", "Announc
                            ments")</li>
                        <li>@Html.ActionLink("About", "About", "Home")</li>
                        <li>@Html.ActionLink("Contact", "Contact", "Home")</li>
                    </ul>
                    @Html.Partial("_LoginPartial")
                </div>
            </div>
        </div>
        <div class="container body-content">
            @RenderBody()
            <hr />
            <footer>
                <p><img src="~/Content/logo.gif" /> &copy; @DateTime.Now.Year - BestWay
                CRM</p>
            </footer>
        </div>
[...]
</body>
</html>
```

5.3. Wsparcie dla telepracy i kontrola jej jakości

Proces uwierzytelniania opisany w rozdziale 3.4.3 został zrealizowany za pomocą ASP Identity 2.2. Autor za pomocą metody `Seed()`, wywoływanej podczas każdej migracji bazy danych, zasiedlił bazę danych poszczególnymi rolami oraz użytkownikami, co zostało zademonstrowane w listingu 9.

Listing 9. Sposób utworzenia ról i powiązanych z nimi użytkowników oraz utrwalenie ich w bazie danych

```
try {
     var store = new UserStore<ApplicationUser>(context);
     var userManager = new ApplicationUserManager(store);
     var roleManager = new RoleManager<IdentityRole>(new
     RoleStore<IdentityRole>(context));
     //dodanie roli Admin, TeleMarketer, Marketer do bazy danych
     string roleName = "Admin";
     if (!roleManager.RoleExists(roleName)) {
         roleManager.Create(new IdentityRole(roleName));
     }
     roleName = "TeleMarketer";
     if (!roleManager.RoleExists(roleName)) {
         roleManager.Create(new IdentityRole(roleName));
     }
     roleName = "Marketer";
     if (!roleManager.RoleExists(roleName)) {
         roleManager.Create(new IdentityRole(roleName));
     }

     //dodanie poszczególnych użytkowników do bazy danych
     var user = new ApplicationUser() { user = new ApplicationUser() { Email =
     "s8359@pjwstk.edu.pl", UserName = "s8359@pjwstk.edu.pl" };
     userManager.Create(user, "TestPass44!");
     userManager.AddToRole(user.Id, "Admin");
```

```
    user = new ApplicationUser() { Email = "marketerbestway1@gmail.pl", UserName = "marketerbest" +
    "way1@gmail.pl" };
    userManager.Create(user, "TestPass44!");
    userManager.AddToRole(user.Id, "Marketer");

    user = new ApplicationUser() { Email = "telemarketerbestway1@gmail.pl",
    UserName = "telemarketerbestway1@gmail.pl" };
    userManager.Create(user, "TestPass44!");
    userManager.AddToRole(user.Id, "TeleMarketer");

} catch (DbEntityValidationException e) {
       System.Diagnostics.Debug.WriteLine("EXC: ");
            foreach (DbEntityValidationResult result in e.EntityValidationErrors) {
                foreach (DbValidationError error in result.ValidationErrors) {
                    System.Diagnostics.Debug.WriteLine(error.ErrorMessage);
                }
            }
}
```

Aby zwiększyć kontrolę nad miejscem wykonywania obowiązków przez pracownika, mechanizm ASP Identity 2.2 został wzbogacony prze autora o zapis współrzędnych geograficznych miejsca logowania. Autor uzyskał dostęp do szerokości, wysokości geograficznej oraz dokładności pomiaru poprzez skorzystanie z Google Maps Geolocation API. Skrypt ten, zawarty w listingu 10., jest wykonywany po załadowaniu strony logowania:

Listing 10. Implementacja wyszukiwania współrzędnych geograficznych użytkownika

```
@section Scripts {
    <script type="text/javascript"
src="http://maps.googleapis.com/maps/api/js?sensor=false"></script>
    <script>
        var options = {
            enableHighAccuracy: true,
            timeout: 5000,
            maximumAge: 0
        };
        var x = document.getElementById("coordinates");
        var latitude = document.getElementById("Latitude");
        var longitude = document.getElementById("Longitude");
        var accuracy = document.getElementById("Accuracy");
        function getLocation() {
            if (navigator.geolocation) {
                var position = navigator.geolocation.getCurrentPosition(showPosition,
null, options);

            } else {
                x.innerHTML = "Geolocation is not supported by this browser.";
            }
        }
```

```
        function showPosition(position) {
            latitude.value = position.coords.latitude;
            longitude.value = position.coords.longitude;
            accuracy.value = position.coords.accuracy;
            InitializeMap(position)
        }
        var map;
        var geocoder;
        function InitializeMap(position) {
            var latlng = new google.maps.LatLng(position.coords.latitude, posi-
tion.coords.longitude);
            var mapOptions =
            {
                zoom: 16,
                center: latlng,
                mapTypeId: google.maps.MapTypeId.ROADMAP,
                disableDefaultUI: true
            };
            map = new google.maps.Map(document.getElementById("map"), mapOptions);
            var marker = new google.maps.Marker({
                position: latlng,
                title: 'Your position'
            });
            marker.setMap(map);

        }
        window.addEventListener('load', getLocation);
    </script>
    @Scripts.Render("~/bundles/jqueryval")
}
```

Brak zgody użytkownika na ustalenie położenia skutkuje uniemożliwieniem zalogowania. Aby umożliwić takie zachowanie ze strony serwera należy zdefiniować odpowiedni `LoginViewModel` przedstawowiony w listingu 11., zawierający atrybuty wymagane przy logowaniu do systemu.

Listing 11. ViewModel specyfikujący parametry logowania do aplikacji

```
public class LoginViewModel {
        [Required]
        [EmailAddress]
        [Display(Name = "Email")]
        public string Email { get; set; }

        [Required]
        [DataType(DataType.Password)]
        [Display(Name = "Password")]
        public string Password { get; set; }

        [Display(Name = "Remember me?")]
        public bool RememberMe { get; set; }
        //współrzednę geograficzne
```

```
        public double Latitude { get; set; }
        public double Longitude { get; set; }
        public double Accuracy { get; set; }

    }
```

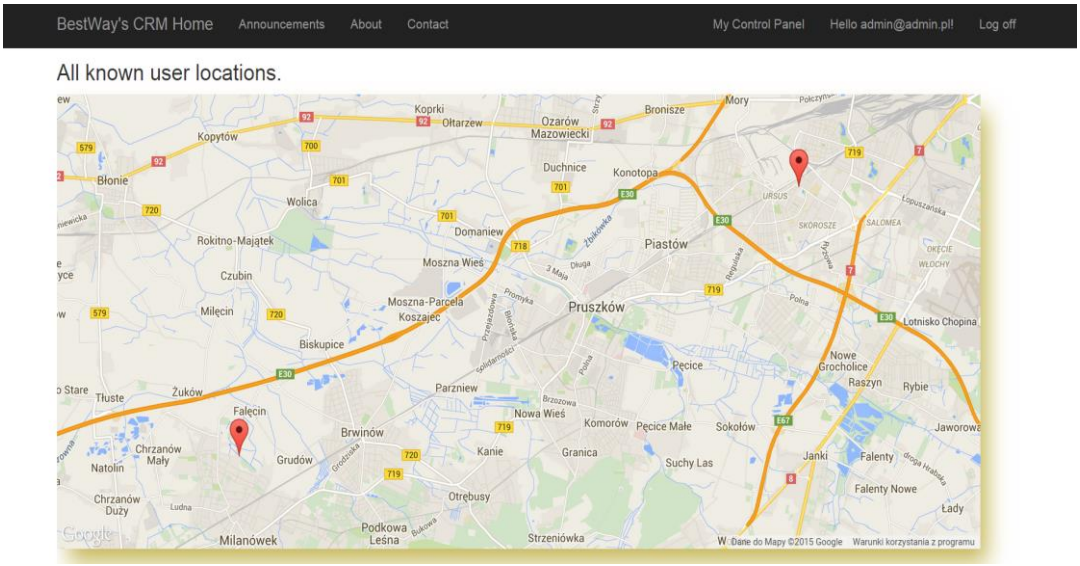
Zapisanie w bazie danych informacji o miejscu logowania danego użytkownika odbywa się po pomyślnej autoryzacji po stronie kontrolera. Metoda akcji `Login()` zawarta w listingu 12., jest wywoływana po zatwierdzeniu przez użytkownika formularza logowania. Wraz z nazwą użytkownika, hasłem oraz współrzędnymi przesyłany jest również `AntiForgeryToken` uniemożliwiający przeprowadzenie ataku typu Cross Site Scripting.

Listing 12. Metoda akcji obsługująca zatwierdzony przez użytkownika formularz

```
        [HttpPost]
        [AllowAnonymous]
        [ValidateAntiForgeryToken]
        public async Task<ActionResult> Login(LoginViewModel model, string returnUrl) {
            if (ModelState.IsValid) {
                var user = await UserManager.FindAsync(model.Email, model.Password);
                if (user != null) {
                    ApplicationUser userSecondInstance = db.Users.Find(user.Id);

                    //dodanie do bazy adresu IP oraz daty logowania
                    ApplicationUserAction action = new ApplicationUserAction { Description =
                    "Logged in at " + DateTime.Now + " from IP " + Request.UserHostAddress +
                    ".", TimeStamp = DateTime.Now, Actor = userSecondInstance };
                    db.ApplicationUserActions.Add(action);

                    //dodanie do bazy współrzędnych użytkownika
                    Coordinates loginCoordinates = new Coordinates() { Latitude = mod
                    el.Latitude, Longitude = model.Longitude, Accuracy = model.Accuracy, User
                    = userSecondInstance };
                    db.Coordinates.Add(loginCoordinates);

                    //zatwierdzenie zmian w bazie
                    db.SaveChanges();
                    await SignInAsync(user, model.RememberMe);
                    return RedirectToLocal(returnUrl);
                } else {
                    ModelState.AddModelError("", "Invalid username or password.");
                }
            }
```

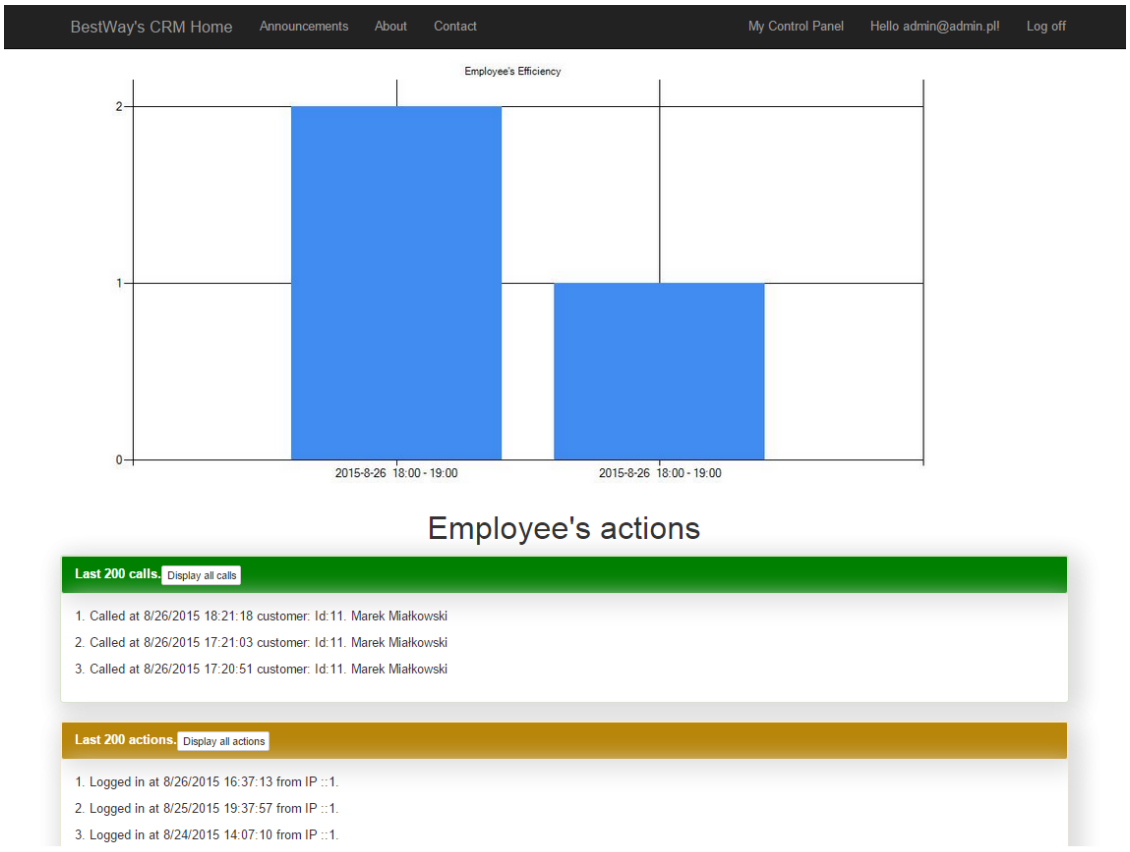
Gdy zajdzie potrzeba sprawdzenia czasu, adresu IP oraz miejsca z których pracownik się logował, administrator może wybrać kontrolowanego użytkownika z listy i obejrzeć informacje zebrane przez

system. Zebrane lokalizacje są prezentowane na zamieszczonej poniżej interaktywnej mapie automatycznie dopasowującej swoje powiększenie do prezentowanych miejsc.



Rysunek 6. Mapa prezentująca miejsca logowania pracownika

System umożliwia również prezentację wydajności pracownika wyrażonej zależnością ilości wykonanych połączeń w godzinnych przedziałach czasu, a także informacje o ostatnich logowaniach i przeprowadzonych rozmowach widoczne na rysunku 7.



Rysunek 7. Wykres wydajności, lista ostatnich rozmów oraz logowań do systemu

5.4. Bezpieczeństwo informacji

Informacje opisane w rozdziale 5.2 są dostępne wyłącznie dla użytkownika zalogowanego w roli Admin. Podobnie możliwość eksportowania wyników przeprowadzonych ankiet, czy edycja profilów każdego z użytkowników leży gestii administratora.

Podczas projektowania zróżnicowanej funkcjonalności aplikacji internetowej zależnej od aktora należy powziąć starania w dwóch obszarach. W pierwszej kolejności należy zadbać aby użytkownik nie miał świadomości istnienia niedostępnych dla niego funkcji. Jest to składnik, który można nazwać częścią ideologii Security by Obscurity, jednakże niewiedza nie zachęca szeregowego użytkownika do nawet najprostszych prób penetracji zabezpieczeń. Efekt ten uzyskać można poprzez usunięcie z interfejsu graficznego elementów, których funkcjonalności programista nie chce ujawniać pracownikowi w danej roli.

Aby osiągnąć ten cel, należy w pliku wybranego widoku opakować komponenty widoczne tylko dla wybranych ról, w tym wypadku dla roli Admin, w następującą instrukcję warunkową zawartą w listingu 13.

Listing 13. Instrukcja warunkowa sprawdzająca uprawnienia użytkownika

```
@if (Request.IsAuthenticated && User.IsInRole("Admin")) { /*komponenty*/ }
```

Ukrycie możliwości systemu jest jednakże tylko pozornym zabezpieczeniem. Przykładowo, marketer znający charakterystykę linków w obrębie systemu, w tym wypadku http://adresAplikacji.domena/Person/Details/[IdKlienta] mimo braku dostępu do listy klientów poprzez interfejs użytkownika, może uzyskać dostęp do danych osobowych poprzez odgadnięcie adresu do chronionego zasobu. W tej sytuacji wymagana jest zatem ochrona danych po stronie serwera, odpowiednio obsługująca przypadki prób nieuprawnionego dostępu do podstrony.

Efekt ten można osiągnąć w dwojaki sposób:

1. Nałożyć na kontroler lub pojedynczą metodę akcji stosowną adnotację wymuszającą autoryzację użytkownika w sposób zrealizowany w listingu 14.

Listing 14. Adnotacja wymagająca od użytkownika posiadania uprawnień administratora lub telemarketera

```
[Authorize(Roles = "Admin, TeleMarketer")]
```

2. W przypadku gdy użytkownik ma posiadać tylko ograniczony do pewnego stopnia dostęp do zasobów, przykładowo gdy tylko może przeglądać dane przydzielonych mu klientów, należy ręcznie zaimplementować rozwiązanie realizujące ograniczenie. Z reguły wymaga to sprawdzenia roli użytkownika, a następnie jego uprawnień od których będzie zależeć odpowiedź serwera. Do opisanego scenariusza autor proponuje poniższe rozwiązanie.

Listing 15. Implementacja metody akcji zwracającej wynik zależny od uprawnień użytkownika

```
public ActionResult Details(int? id) {
            //jeżeli nie wybrano klienta zwróć odpowiedź HTTP 400
            if (id == null) {
                return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
            }
            Person person = db.Persons.Find(id);
            //jeżeli klient nie istnieje w baize zwróć HTTP 404
            if (person == null) {
                return HttpNotFound();
            }
            //jeżeli użytkownik jest w roli Marketer i nie posiada uprawnień do
            przeglądania danych wybranego klienta zwróc HTTP 400
            if (User.IsInRole("Marketer")) {
                var user = UserManager.FindById(User.Identity.GetUserId());
                if (!user.PersonsPermittedToView.Any(pptv =>
                    pptv.CustomerToVisit.Id == person.Id)) {
                    return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
                }
            }
            return View(person);
        }
```

Kombinacja obu podejść oraz ukrycia zastrzeżonych części interfejsu została wykorzystana w przekroju całego systemu CRM.

Aby utrudnić kopiowanie danych z systemu poprzez urządzenia nad którymi administrator systemu nie ma pełnej kontroli autor uniemożliwił zaznaczenie jakiejkolwiek nieedytowalnej treści umieszczonej w interfejsie użytkownika. Nie ograniczył jednakże funkcjonalności pól edycyjnych. Zostało to uzyskane poprzez zastosowanie odpowiednich stylów CSS w pliku `site.css` wyszczególnionych w listingu 16.

Listing 16. Style CSS uniemożliwiające zaznaczenie nieedytowalnych komponentów strony

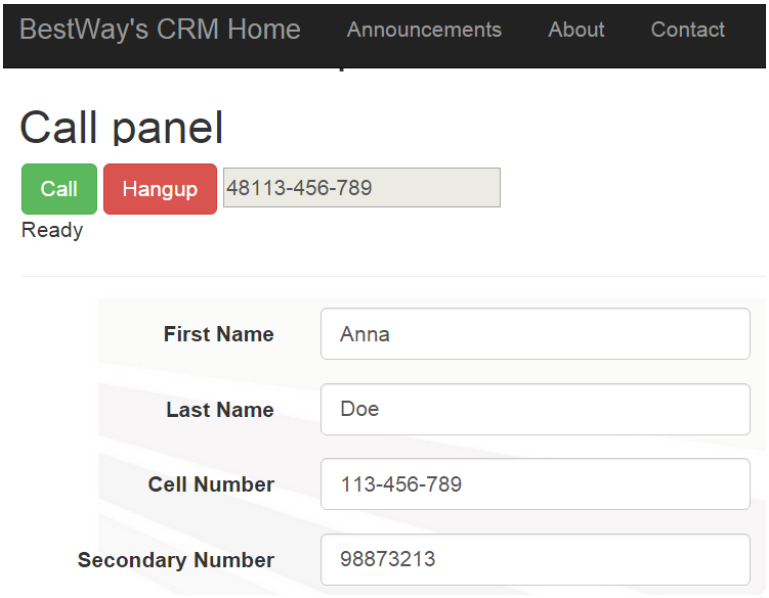
```
body div#background {
```

```
    [...]
    /*Unimożliwienie zaznaczenia tekstu*/
    -webkit-touch-callout: none;
    -moz-user-select: none;
    -khtml-user-select: none;
    -webkit-user-select: none;
    user-select: none;
    -ms-user-select: none;
}
```

5.5. Inicjowanie i obsługa połączeń z poziomu aplikacji

W rozdziale 3.4.4. autor zaznaczył konieczność integracji obsługi połączeń z prototypem aplikacji internetowej. Wyżej wymienioną kwestię rozwiązano poprzez wykorzystanie architektury operatora telefonii internetowej Twilio.

Aktor po wciśnięciu przycisku `Call` widocznego na rysunku 8. inicjuje połączenie, po wciśnięciu przycisku `Hangup` aktor kończy połączenie, transmisja audio jest obsługiwana poprzez przeglądarkę w czasie rzeczywistym.



Rysunek 8. Panel sterowania połączeniem

Aby umożliwić aktorowi wykonywanie połączeń użytkownikowi, należy w pierwszej kolejności po stronie serwera wygenerować token w sposób zaprezentowany na listingu 17., który przekazany użytkownikowi poprzez obiekt dynamiczny `ViewBag` jest poświadczeniem wyrażenia zgody na inicjalizację połączenia.

Listing 17. Tworzenie tokena i przekazanie go użytkownikowi

```
            var capability = new Twilio.TwilioCapability(Settings.AccountSid, Set
            tings.AuthToken);
            capability.AllowClientOutgoing(Settings.AppSid);
            capability.AllowClientIncoming("jenny");
            string token = capability.GenerateToken();
            ViewBag.token = token;
```

Na potrzeby obsługi należy do projektu dodać referencje do następujących bibliotek:

- Twilio.Api
- Twilio.Client.Capability
- Twilio.Twiml
- Twilo.Mvc

Token jest generowany na podstawie identyfikatora konta Twilio oraz identyfikatora aplikacji obsługującej połączenia w organizacji. Może być to aplikacja zewnętrzna względem systemu CRM, zostanie jej poświęcony odpowiedni segment niniejszego rozdziału. Oba parametry są niejawne dla użytkownika.

Po stronie użytkownika należy zaimplementować interfejs użytkownika w postaci co najmniej dwóch przycisków służących do podejmowania rozmów i ich rozłączania, co ukazano w listingu 18. Autor ze względu na znaczną długość, brak szczególnej wartości merytorycznej oraz publiczną dostępność w formie zbliżonej do wykorzystanej[6], nie publikuje treści metod `call()` oraz `hangup()`.

Listing 18. Implementacja interfejsu użytkownika – panelu telefonicznego

```
@if (ViewBag.Token != null) {
    <h2>Call panel</h2>
    <button class="btn btn-success" onclick="call();">
        Call
    </button>
    <button class="btn btn-danger" onclick="hangup();">
```

```
        Hangup
    </button>

    <input type="text" id="number" name="number"
           value="48@(Model.CellNumber)" disabled />
}
```

Autor stworzył zewnętrzną aplikację, która decyduje o parametrach rozmowy wychodzącej z aplikacji. Aplikacja ta musi być umieszczona na serwerze pod ogólnodostępnym adresem URL. Aplikacja tak jak główny CRM została wytworzona przy pomocy ASP.NET-MVC5. Adres usługi[3] to http://voiceapp-001-site1.myasp.net/voice, musi być ona dodana w panelu konfiguracyjnym Twilio.

Składa się z metody akcji w której w której parsowane jest żądanie połączenia z danym numerem. Do metody akcji przyporządkowany jest widok umieszczony w listingu 19.

Listing 19. Implementacja widoku aplikacji akceptującej żądanie połączenia

```
<?xml version="1.0" encoding="UTF-8" ?>
<Response>
    <Dial record="true" callerId="@ViewBag.CallerId">
        @Html.Raw(ViewBag.NumberOfClient)
    </Dial>
</Response>
```

który po przetworzeniu przez silnik Razor w czasie wykonania programu przyjmuje postaci arkusza XML z listingu 20.

Listing 20. Przykładowa odpowiedź aplikacji akceptującej żądanie połączenia

```
<?xml version="1.0" encoding="UTF-8"?>
<Response>
      <Dial record="true" callerId="48326304351">
            <Client>jenny</Client>
      </Dial>
</Response>
```

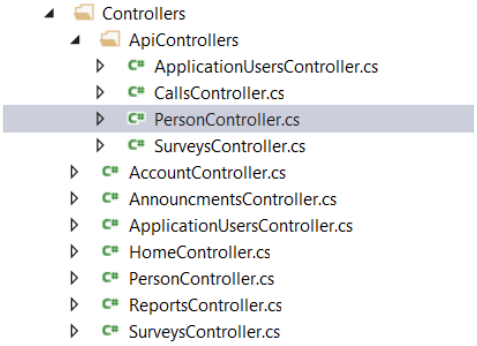
[3] Serwis dostępny 14.08.2015

W tym konkretnym przypadku autor zarządził, że rozmowa telemarketera będzie nagrywana, wybrany został także numer, który zostanie wyświetlony na ekranie telefonu odbiorcy. Wszystkie nagrania są przechowywane na serwerach należących do Twilio i w każdym momencie dostępne poprzez panel klienta dostępny na stronie internetowej Twilio.

5.6. Rozszerzenie systemu o API dla aplikacji trzecich

Aby zgodnie z rozdziałem 3.4.6 umożliwić rozwój systemu przez osoby trzecie, autor stworzył interfejs programistyczny wykorzystujący protokół OData umożlwiający czerpanie informacji z bazy danych.

Autor dla każdej klasy, której obiekty postanowił udostępnić utworzył kontroler Web API2 i umieścił go w osobnym folderze ApiControllers, widocznym poniżej.



Rysunek 9. Struktura kontrolerów w wytworzonym rozwiązaniu

Każdy z kontrolerów dziedziczy po klasie `ODataController` i implementuje szereg metod zwracających wyniki pobrane z bazy danych poprzez instancję `ApplicationDbContext`. Kod każdego z kontrolerów jest wykonany według standardowych szablonów dla technologii WebApi2, jednakże aby aplikacja była w stanie prawidłowo parsować URL z zapytaniami OData należy zarejestrować wszystkie udostępniane zasoby w klasie `WebApiConfig`, w sposób przedstawiony w listingu 21.

Listing 21. Rejestracja zasobów udostępnianych poprzez protokół OData

```
public static class WebApiConfig {
        public static void Register(HttpConfiguration config) {
            config.MapHttpAttributeRoutes();

            config.Routes.MapHttpRoute(
                name: "DefaultApi",
```

```
                routeTemplate: "api/{controller}/{id}",
                defaults: new { id = RouteParameter.Optional }
            );
            ODataConventionModelBuilder builder = new ODataConventionModelBuilder();
            builder.EntitySet<Person>("Person");
            builder.EntitySet<Answer>("Answers");
            builder.EntitySet<CallToMake>("CallsToMake");
            builder.EntitySet<CustomerStatus>("CustomerStatuses");
            builder.EntitySet<Meeting>("Meetings");
            builder.EntitySet<Call>("Calls");
            builder.EntitySet<Report>("Reports");
            builder.EntitySet<Status>("Statuses");
            builder.EntitySet<Call>("Calls");
            builder.EntitySet<ApplicationUser>("Users");
            builder.EntitySet<IdentityUserClaim>("IdentityUserClaims");
            builder.EntitySet<ApplicationUserAction>("ApplicationUserActions");
            builder.EntitySet<Coordinates>("Coordinates");
            builder.EntitySet<IdentityUserLogin>("IdentityUserLogins");
            builder.EntitySet<AuthorizationToView>("AuthorizationsToView");
            builder.EntitySet<IdentityUserRole>("IdentityUserRoles");
            builder.EntitySet<Survey>("Surveys");
            builder.EntitySet<Question>("Questions");
            config.Routes.MapODataRoute("odata", "odata", builder.GetEdmModel());
        }
    }
```

Poprawnie skonfigurowana aplikacja, posiada imponujące możliwości. Przykładowo zwraca zbiór rekordów w notacji JSON, na poniższe zapytania.

http://adresAplikacji.domena/odata - wspólny prefiks

Sufiksy:

- /Surveys?$expand=Questions – zwraca listę wszystkich ankiet I pytań w nich zawartych
- /Surveys?$expand=Questions/Answers – jw. dodatkowo zawiera listę możliwych odpowiedzi
- /Person - zwraca listę klientów zawartych w bazie wraz z ich atrybutami
- /Person?$select=FirstName,LastName,Address - jw. lecz ogranicza listę atrybutów

- /Person?$expand=ReceivedCalls - zwraca listę wszystkich rozmów z wyszczególnionymi atrybutami klientów
- /Person?$expand=ReceivedCalls/Caller – jw. dodatkowo zwraca dane agenta dzowniącego do klienta
- /Person(2)?$expand=ReceivedCalls/Caller – jw. jednakże zwrócone rozmowy dotyczą tylko klienta o id wynoszącym 2, wynik żądania został zaprezentowany w listingu 22.
- /Person?$expand=ReceivedCalls&$select=ReceivedCalls - zwraca tylko listę rozmów wraz z ich atrybutami bez informacji o klientach

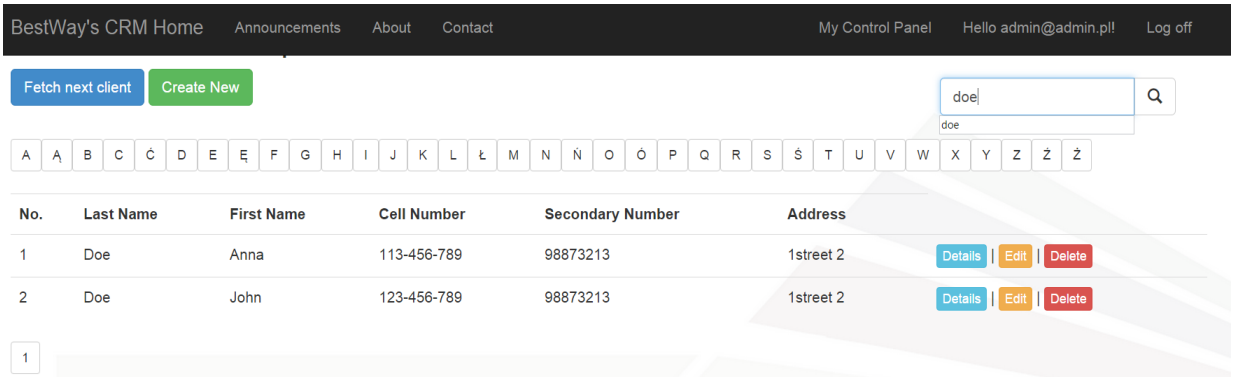
Listing 22. Przykładowa odpowiedź aplikacji na URL http://localhost:17697/odata/Person(2)?$expand=ReceivedCalls/


```
{
  "oda-
ta.metadata":"http://localhost:17697/odata/$metadata#Person/@Element","ReceivedCal
ls":[
    {
      "Id":8,"TimeStamp":"2015-08-27T03:29:39.943"
    },{
      "Id":9,"TimeStamp":"2015-08-27T04:12:21.943"
    }
  ],"Id":2,"FirstName":"Anna","LastName":"Doe","CellNumber":"113-456-
789","SecondaryPhoneNumber":"98873213","Email":null,"Address":"1street
2","BirthDate":"2015-06-20T00:00:00","Pesel":"548555672","Notes":"Lorem ipsum note
other","StatusId":null,"PersonalDataProcessing":false,"IsCalledRightNow":false,"Cu
stomerStatusId":null,"NIP":null
}
```


5.7. Pozostała funkcjonalność

W niniejszym rozdziale autor prezentuje zaimplementowane rozwiązania, niebędące bezpośrednią odpowiedzią na postawione w pracy problemy, jednakże odpowiadają za pełną użyteczność aplikacji. Proponowane rozwiązanie jest w pełni funkcjonalnym prototypem. Pozwala na przeglądanie listy klientów. Wyniki są filtrowane po pierwszej literze nazwiska oraz stronicowane, jak to przedstawiono

na rysunku 10. Istnieje również opcja wyszukiwania po nazwisku, imieniu, miejscu zamieszkania, numerze telefonu.



Rysunek 10. Prezentacja wyników wyszukiwania dla frazy: „doe”

Aplikacja umożliwia przeglądanie, edytowanie, usuwanie profili klientów, a także przeprowadzanie z nimi ankiet telefonicznych. Ankiety są ładowane asynchronicznie przy pomocy technologii AJAX na żądanie telemarketera, a ich wyniki są zapisywane do relacyjnej bazie danych w czasie rzeczywistym, bez konieczności zatwierdzania formularza, a co za tym idzie przeładowywania strony internetowej. Ankieta z racji swoich rozmiarów może być w dowolnym momencie zostać zwinięta przez pracownika.

Asynchroniczne ładowanie ankiety oraz jej chowanie zostało osiągnięte poprzez dwie funkcje JavaScript zawarte w listingu 23.

Listing 23. Implementacja asynchronicznego pobierania ankiet z serwera

```
    function BtnOnclick() {
        $.ajax({
            type: 'POST',
            url: '@Url.Content("~/Person/_Survey1")',
            data: {
                id: '@Model.Id'
            },
            success: function (data) {
                $('#divpopup').css("display", "block");
                $('#btnExpand').css("display", "none");
                $('#divpopup')[0].innerHTML = data;
            }
        });
```

```
    }
    function CollapseDiv() {
        $('#divpopup').css("display", "none");
        $('#btnExpand').css("display", "block");
    }
```

Zaś zapis danych w czasie rzeczywistym poprzez wysłanie identyfikatora pytania i wybranej przez ankietera odpowiedzi do kontrolera, a następnie aktualizację bazy danych po jego stronie.

Funkcję wywoływaną po stronie przeglądarki po wybraniu odpowiedzi przez ankietera umieszczono w listingu 24, zaś aktualizacja bazy odbywa się po stronie serwera poprzez wykonanie kodu z listingu o numerze 25.

Listing 24. Funkcja asynchronicznie przesyłająca do serwera wybraną odpowiedź

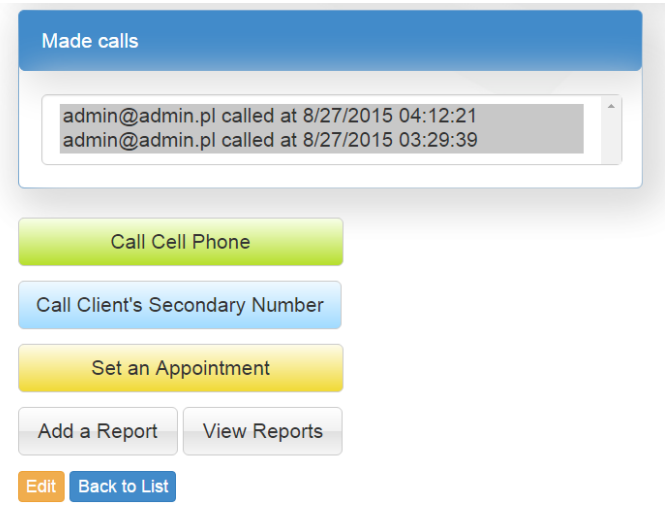
```
    function SubmitClick(qid, aid) {
        var url = '@Url.Action("SubmitSurvey", "Person")';

        $.post(url, { questionId: qid, answerId: aid, personId:@Model.Id }, function (data)
        {});
    }
```

Listing 25. Aktualizacja bazy danych odpowiedzi poprzez metodę akcji

```
public void SubmitSurvey(int questionId, int answerId, int personId) {
            //Usuwanie starej odpowiedzi danego klienta,  jeżeli takowa istnieje
            Person person = db.Persons.Find(personId);
            removeOldAnswer(person, questionId);//usuwa starą odpowiedź powiązaną z
                                                //klientem
            //Znajdowanie nowej odpowiedzi danego klienta
            Answer answerToAdd = db.Answers.Find(answerId);
            //tworzenie powiązania pomiędzy udzieloną odpowiedzią, a klientem
            person.Answers.Add(answerToAdd);
            answerToAdd.Persons.Add(person);
            db.SaveChanges();
        }
```

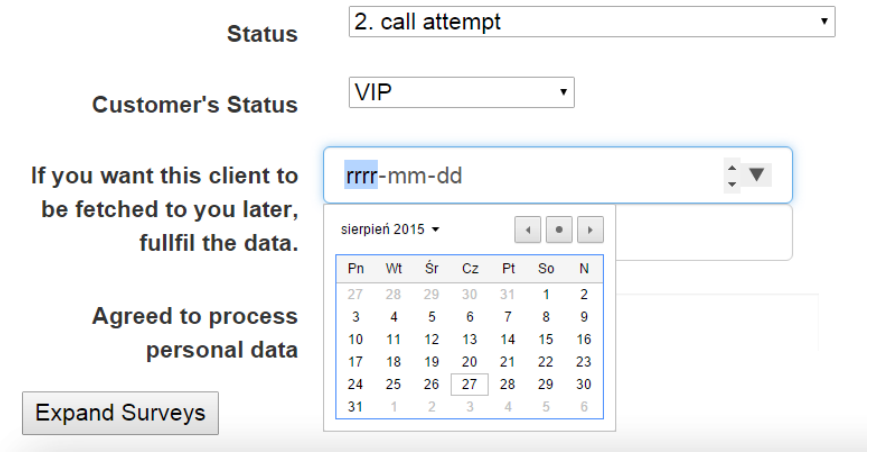
Administrator może również eksportować zebranie wyniki z ankiet do pliku o rozszerzeniu .xls. Każda z możliwych odpowiedzi ma przypisaną wartość liczbową z przedziału [0;1]. W pierwszym wierszu podana jest treść pytania, a w pierwszej kolumnie wymienione są numery id każdego klienta. Komórka leżąca na przecięciu danego pytania i numeru id klienta zawiera wartość liczbową oznaczającą wybraną odpowiedź. Nieaktualne ankiety można także dezaktywować.

Wytworzony prototyp wspiera również przeglądanie oraz dodawanie raportów do każdego klienta jako podsumowanie rozmów lub spotkań w postaci notatek, jak i przypisywanie klientów do marketerów w celu udzielenia im prawa przeglądania profili wybranych klientów i nawiązania z nimi kontaktu. Przyciski inicjalizujące w/w przypadki użycia umieszczono na rysunku 12.



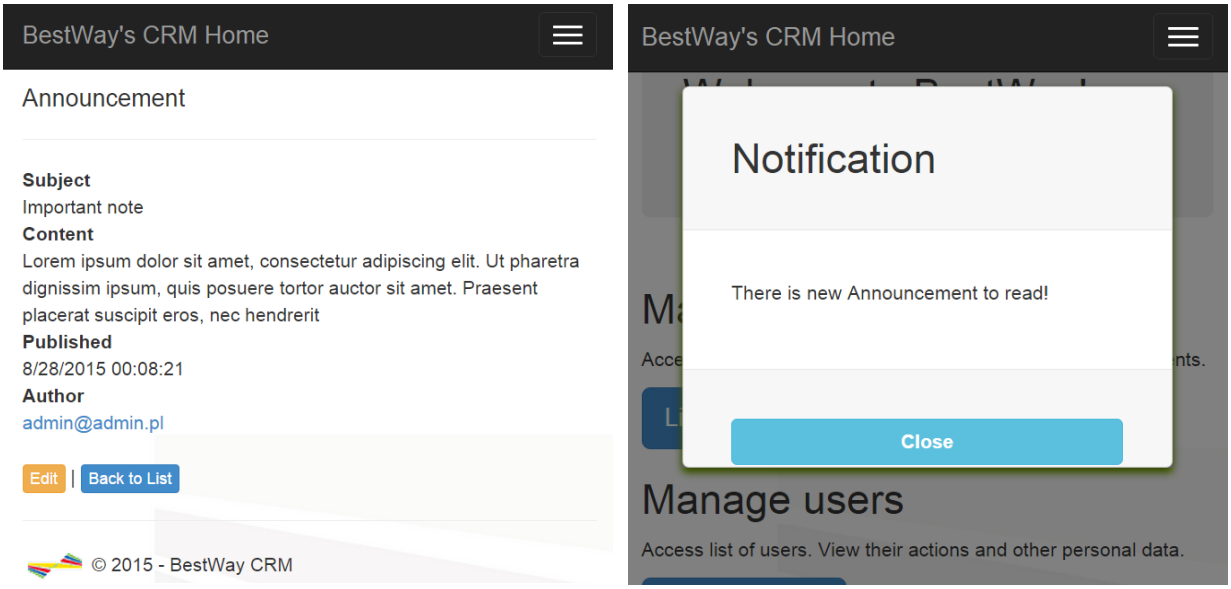
Rysunek 11. Panel akcji w profilu klienta

Każda rozmowa może być podsumowana jednym ze statusów: pierwsza, druga, trzecia próba połączenia, zadzwoń później, odmówiono rozmowy. Może być też również przeniesiona na wybraną godzinę co można spostrzec na rysunku 12. Kontakt automatycznie wróci do agenta wykonującego połączenia po wciśnięciu przycisku `Fetch Next Client` widocznym na rysunku 10.



Rysunek 12. Panel umożliwiający przeniesienie rozmowy na późniejszy termin

Istnieje również funkcjonalność dodawania ogłoszeń o których zostanie powiadomiony każdy z użytkowników animowanym banerem widocznym na rysunkach 14a. i 14b., co ułatwia przełożonemu przekazanie informacji zespołowi, szczególnie w przypadku pracy zdalnej.



Rysunek 14a. oraz 14b. Widok ogłoszenia oraz powiadomienia o nim na stronie głównej

6. Podsumowanie

W tym rozdziale autor ujął cele które pomyślnie zrealizowano, napotkane problemy oraz kierunku rozwoju systemu, które umożliwiłyby wprowadzenie oprogramowania do komercyjnej oferty.

6.1. Zrealizowane założenia

Zaproponowany prototyp rozwiązuje problem pracy zdalnej poprzez dowolną platformę komputerową wspierającą przeglądarkę internetową. W praktyce oznacza dowolny komputer klasy PC, laptop, tablet lub telefon komórkowy. System dopasowuje swój wygląd do użytkowanego urządzenia. Autor dużo uwagi poświęcił bezpieczeństwu dostępu do danych jak i kontroli jakości pracy. System wymaga uwierzytelnienia, jest również odporny na ataki typu XSS. System uniezależnia call center od sprzętowej architektury telefonicznej w placówce przedsiębiorstwa.

Z powyższych przesłanek wynika wniosek iż wymagania postawione w rozdziale 1. oraz 2. zostały spełnione.

6.2. Napotkane trudności

Największą trudność autorowi sprawiła konfiguracja usługi Twilio. Nacisk materiałów dostarczonych przez usługodawcę jest skierowany w stronę innych platform takich jak np. PHP, przez co wsparcie dla programistów platformy .NET okazało się niezadowalające. Trudność sprawiło również generowanie arkuszy XML poprzez Visual Studio, które automatycznie formatowało znaczniki XML zmieniając wielkość znaków. XML jest językiem case-sensitive, przez co autoformatowanie przyczyniło się to do nieudanych prób połączeń.

Drugim stosunkowo dużym wyzwaniem dla autora było zaimplementowanie wygodnego mechanizmu obsługi ankiet, które ładowane są tylko gdy są potrzebne, nie wymagają przeładowywania strony i nie obciążają swoją wielkością bazy danych.

6.3. Kierunki rozwoju

W przyszłości opracowany system zostanie wzbogacony o:

- mechanizm kolejkowania połączeń przychodzących
- tonowe menu głosowe dla połączeń przychodzących
- kreator kampanii z indywidualnymi ankietami, zespołami agentów przyporządkowanych do kampanii, scenariuszami rozmów dla agentów
- obsługę komunikatorów internetowych takich jak: Skype, GG.

Powyższe funkcjonalności umożliwią sprawną obsługę połączeń od klientów, jak i jednoczesne prowadzanie wielu kampanii w obrębie jednego przedsiębiorstwa, przykładowo na zewnętrzne zlecenie. Obsługa komunikatorów pozwoli natomiast na obsługę klientów dla których możliwy jest jedynie kontakt pisemny.

7. Bibliografia

[1]. "Call Center." – Wikipedia, Wolna Encyklopedia. Dostęp 19. stycznia, 2015. http://pl.wikipedia.org/wiki/Call_center.

[2]. "The History of Programming Languages." New Titles. Dostęp 20. czerwca 2015.http://archive.oreilly.com/pub/a/oreilly//news/languageposter_0504.html

[3]. "XML Tutorial." XML Tutorial. Dostęp 15. lipca, 2015. http://www.w3schools.com/xml/

[4]. "Standard ECMA-404 The JSON Data Interchange Format" Dostęp 11. sierpnia 2015. http://www.ecma-international.org/publications/files/ECMA-ST/ECMA-404.pdf

[5]. "Browser Statistics." Browser Statistics. Dostęp 5. lipca 2015. http://www.w3schools.com/browsers/browsers_stats.asp

[6]. "Twilio Client Quickstart." Twilio Cloud Communications. Dostęp 20. sierpnia 2015. https://www.twilio.com/docs/quickstart/csharp/client/incoming-calls oraz program załączony na płycie

[7]. Kodeks Pracy. Warszawa, 2015.

[8]. Scacchi, Walt. "When Is Free/Open Source Software Development Faster, Better, and Cheaper than Software Engineering?" 1. lipca 2004. Dostęp 14. sierpnia 2015. http://www.ics.uci.edu/~wscacchi/Papers/New/Scacchi-BookChapter.pdf

[9]. "WiFi Data Collection: An Update." Official Google Blog. Dostęp 2. sierpnia 2015. http://googleblog.blogspot.com/2010/05/wifi-data-collection-update.html

[10]. Freeman, Eric, and Elizabeth Freeman. "Chapter 12. Patterns of Patterns." w *Head First Design Patterns*, strony:530, 531. O'Reilly, 2004.

[11]. Allen, Scott. "Building Applications with ASP.NET MVC 4." Wykład.

[12]. Habib, Monjurul. "MVC Options for Passing Data Between Current and Subsequent Request." Code Project. October 15, 2012. Dostęp 2 września, 2015. http://tinyurl.com/TempData-MVC-Option

[13]. "XML Examples." W3Schools. Dostęp 2. września, 2015. http://www.w3schools.com/xml/simple.xml

[14]. Prajapati, Snesh. "How to Choose the Best Way to Pass Multiple Models in ASP.NET MVC." - CodeProject. Dostęp 16. czerwca, 2015. http://www.codeproject.com/Articles/717941/How-to-Choose-the-Best-Way-to-Pass- Multiple-Models

[15]. Kucinskas, Darius. "Entity Framework 6 vs NHibernate 4." Devbridge Group. 16. stycznia, 2014. Dostęp 17 czerwca, 2015. https://www.devbridge.com/articles/entity-framework-6-vs-nhibernate-4

[16]. CrockFord, Douglas. "JavaScript: The World's Most Misunderstood Programming Language." Crockford. 2001. Dostęp 10. sierpnia, 2015. http://www.bibme.org/chicago/website-citation?new=true

[17]. Buckler, Craig. "Native JavaScript Equivalents of JQuery Methods: The DOM and Forms." 10. lipca, 2013. Dostęp 11. sierpnia, 2015. http://www.sitepoint.com/jquery-vs-raw-javascript-1-dom-forms/

[18]. "History of Computers and Computing, Internet, Birth, Charles Goldfarb." History of Com puters. Dostęp 14. sierpnia, 2015. http://history-computer.com/Internet/Birth/Goldfarb.html

[19]. "How, Exactly Does HTML5's GeoLocation Work?" Stack Overflow. 14. czerwca, 2010. Dostęp 15. sierpnia, 2015. http://stackoverflow.com/a/3041637/1123020

8. Spis ilustracji oraz tabel

9. Listingi