

Politechnika Warszawska

WYDZIAŁ ELEKTRONIKI
I TECHNIK INFORMACYJNYCH



Instytut Automatyki i Informatyki Stosowanej

Praca dyplomowa inżynierska

na kierunku Informatyka
w specjalności Systemy Informacyjno-Decyzyjne

Aplikacja wspierająca planowanie zadań pracownikowi
organizacji zwinnej

Maciej Suchocki

Numer albumu 277377

promotor
dr inż. Mariusz Kaleta

WARSZAWA 2019

Aplikacja wspierająca planowanie zadań pracownikowi organizacji zwinnej

Streszczenie

Praca poświęcona jest zaprojektowaniu, implementacji oraz przetestowaniu aplikacji na platformie CRM, która ma za zadanie umożliwienie planowania, monitorowania i analizowania procesów stworzonych z postępujących po sobie zadań. Aplikacja powinna być zaprojektowana w taki sposób, aby mogła służyć dowolnej organizacji zwinnej lub chcącej wdrożyć elementy zwinne do swojej pracy, niezależnie od dziedziny którą się zajmuje. Kluczowym aspektem było spełnienie warunku wirtualizacji przedsiębiorstwa, gdzie pracownicy za pomocą wachlarza funkcjonalności systemu mogą samodzielnie organizować swoją pracę oraz przenieść najważniejsze elementy zarządzania firmą do jednego programu. Zaimplementowany kreator procesów umożliwia zarządzanie w organizacji przebiegami stworzonymi z postępujących po sobie zadań w postaci acyklicznych grafów skierowanych, pomagających w szybkiej rekonfiguracji pracy i znalezienia najbardziej opłacalnego rozwiązania. W wyzwalaczach, odpowiadających za logikę aplikacji przy operacjach na zintegrowanej bazie danych, posłużono się wzorcem delegacji, co uchroni działanie od błędów związanych z limitami narzuconymi przez platformę i pomoże w dalszym rozwijaniu aplikacji.

Słowa kluczowe: organizacja zwinna, wirtualizacja organizacji, platforma CRM, skierowany graf acykliczny, wzorzec delegacji

Application supporting task planning for an agile organization employee

Abstract

The aim of this thesis was to design, implement and test an application on CRM platform that enables planning, monitoring and analyzing processes created from consecutive jobs. The application should be designed in such a way that it can serve any agile organization or those who want to implement agile elements for their work, regardless of the field they deals with. The key aspect was realization of organization virtualization, where employees using system functionalities can independently organize their work and transfer the most important elements of the company management to one program. The implemented process creator helps in managing processes consisting of tasks in the form of directed acyclic graphs which supports quick work reconfiguration to find the most cost-effective solution. Triggers which are responsible for application logic during operations on the integrated database have used the delegation pattern in order to protect from errors related to the limits imposed by the platform and help in further development of the application.

Key words: agile organization, organization virtualization, CRM platform, directed acyclic graph, delegation pattern



Politechnika Warszawska

załącznik do zarządzenia nr 28/2016 r.
Rektora PW

„załącznik nr 3 do zarządzenia nr 24/2016 Rektora PW

Warszawa, 30.08.2019
.....
miejscowość i data

.....
Maciej Suchocki
.....
imię i nazwisko studenta
.....
277377
.....
numer albumu
Informatyka
.....
kierunek studiów

OŚWIADCZENIE

Świadomy/-a odpowiedzialności karnej za składanie fałszywych zeznań oświadczam, że niniejsza praca dyplomowa została napisana przeze mnie samodzielnie, pod opieką kierującego pracą dyplomową.

Jednocześnie oświadczam, że:

- niniejsza praca dyplomowa nie narusza praw autorskich w rozumieniu ustawy z dnia 4 lutego 1994 roku o prawie autorskim i prawach pokrewnych (Dz.U. z 2006 r. Nr 90, poz. 631 z późn. zm.) oraz dóbr osobistych chronionych prawem cywilnym,
- niniejsza praca dyplomowa nie zawiera danych i informacji, które uzyskałem/-am w sposób niedozwolony,
- niniejsza praca dyplomowa nie była wcześniej podstawą żadnej innej urzędowej procedury związanej z nadawaniem dyplomów lub tytułów zawodowych,
- wszystkie informacje umieszczone w niniejszej pracy, uzyskane ze źródeł pisanych i elektronicznych, zostały udokumentowane w wykazie literatury odpowiednimi odnośnikami,
- znam regulacje prawne Politechniki Warszawskiej w sprawie zarządzania prawami autorskimi i prawami pokrewnymi, prawami własności przemysłowej oraz zasadami komercjalizacji.

Oświadczam, że treść pracy dyplomowej w wersji drukowanej, treść pracy dyplomowej zawartej na nośniku elektronicznym (płyce kompaktowej) oraz treść pracy dyplomowej w module APD systemu USOS są identyczne.

.....
czytelny podpis studenta”

Spis treści

1	Wstęp	9
1.1	Motywacja	10
1.2	Cel i zakres pracy	10
1.3	Układ pracy.....	11
2	Omówienie istniejących rozwiązań.....	12
2.1	ProcessClick - Your Agile BPM for Salesforce	13
2.2	Oracle Process Cloud	14
2.3	IBM Business Automation Workflow	15
3	Analiza potrzeb użytkowników	16
3.1	Użytkownicy.....	17
3.2	Wymagania funkcjonalne.....	17
3.3	Wymagania niefunkcjonalne	23
4	Projekt rozwiązania.....	23
4.1	Specyfika aplikacji na platformie Salesforce	24
4.2	Wzorzec Model-Widok-Kontroler	24
4.3	Model danych	25
4.4	Architektura aplikacji i moduły funkcjonalności.....	27
4.4.1	Tworzenie zadań oraz procesów	27
4.4.2	Monitorowanie postępów	33
4.4.3	Kontrola i analiza menadżerka.....	40
4.5	Ograniczenia platformy	45
5	Implementacja	46
5.1	Technologie	46
5.1.1	Lightning Web Components (LWC) i biblioteka D3.js.....	46
5.1.2	Logika aplikacji w języku APEX	49
5.2	Narzędzia implementacyjne.....	50
5.2.1	Visual Studio Code	50
5.2.2	Salesforce Extension Pack	50
5.2.3	Ant Migration Tool	50
5.3	System kontroli wersji	51

5.4	Walidacja acykliczności grafu skierowanego.....	51
5.5	Wzorzec delegacji w wyzwalaczach.....	53
5.6	Ograniczenia implementacyjne platformy.....	58
6	Testowanie	58
6.1	Testy jednostkowe	60
6.2	Scenariusze testowe.....	61
7	Podsumowanie	67
7.1	Efekt końcowy.....	67
7.2	Możliwości dalszego rozwoju	68
8	Bibliografia	68
9	Spis rysunków, tabel oraz listingów.....	72
10	Załącznik A. Zawartość płyty CD	75

1 Wstęp

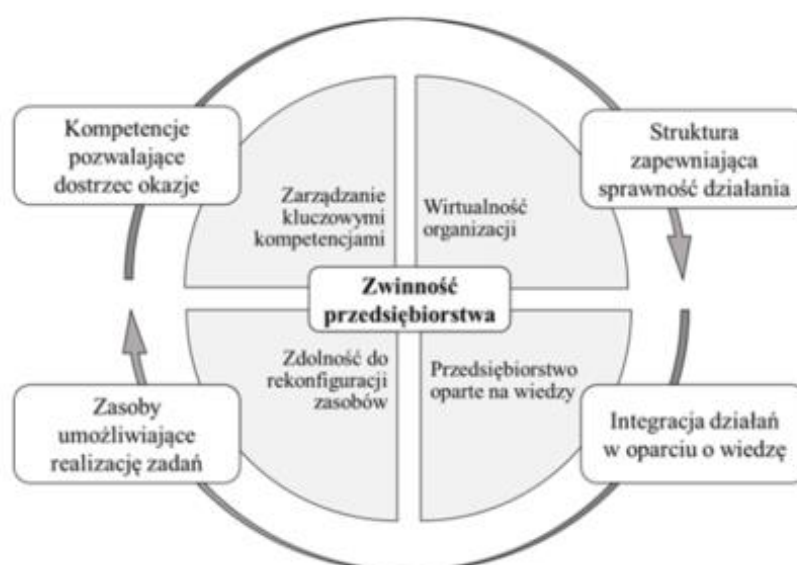
W przeciągu ostatnich kilkunastu lat przez intensywny rozwój gospodarczy wielu gałęzi przemysłu oraz usług, w szczególności związanych z branżą technologii informacyjnej (ang. IT) [1], firmy zmuszane są coraz szybciej reagować na zmieniający się rynek, agresywne działania konkurencji i potrzeby klientów. W konsekwencji, w dużej mierze korporacje, posiadając odpowiednie zasoby, starają się zaadaptować sposoby, aby zmierzyć się z wyżej wymienionymi problemami.

W związku z tym rozpoczęto stosowanie metod, które poprawiły nie tylko komunikację wewnątrz firmy czy zmieniające się spersonalizowane wymagania produktu pod klienta ale również elastyczne dostosowanie każdego z zasobów organizacji takich jak: technologie, wiedza, ludzie, cele przedsiębiorstwa czy organizacja pracy. Wszystkie te elementy można zaklasyfikować jako akcje podejmowane w ramach organizacji zwinnej [2].

Źródeł koncepcji zwinnego przedsiębiorstwa można szukać w wielu istniejących już od lat dziedzinach, które koncentrują się na elastycznym podejściu, takich jak:

- zwinne wytwarzanie (ang. agile manufacturing)
- zwinne zarządzanie (ang. agile management)
- zwinne programowanie (ang. agile software development)

Każde z powyższych elementów charakteryzuje się innymi metodami w zależności od analizowanego obszaru, jednak wszystkie z nich w uproszczeniu określane są jako zwinne z powodu możliwości dopasowania się do aktualnych okoliczności. Z tego powodu organizacja zwinna to w uogólnieniu taka, która jest w stanie błyskawicznie dostosować się do napotkanych różnorodnych przeciwności, przy pomocy wypracowanych metod, struktur, narzędzi oraz zasobów [3].



Rysunek 1.1 Relacje między czynnikami kształtującymi zwinność przedsiębiorstwa. [3]

1.1 Motywacja

Na rynku istnieją jednak również starsze przedsiębiorstwa, które mogłyby zaadaptować pewne elementy zwinne takie jak: wirtualność organizacji, możliwość szybkiej rekonfiguracji organizacji pracy, monitorowanie wyników tych zmian oraz zdefiniowanie funkcji powiązanych z bezpośrednim kontaktem z klientem już na niższych szczeblach pracowniczych, co mogłoby się przełożyć na dostarczanie bardziej spersonalizowanych produktów zgodnie z oczekiwaniami konsumenta, zmniejszenie kosztów i większą kontrolę przez kadrę menadżerską. Duża część z nich najprawdopodobniej korzysta z przestarzałych systemów informatycznych, do których pracownicy są przyzwyczajeni. Powyższe powody podwyższają próg wejścia w elastyczne funkcjonowanie, dlatego ważnym elementem nowego systemu musiałby być przejrzysty interfejs i intuicyjne z niego korzystanie. Jednak jednocześnie rozwiązanie musiałoby być na tyle generyczne, aby dostosować się do wielorakich branż, niekoniecznie w sektorze technologii informacyjnej.

1.2 Cel i zakres pracy

Głównym celem tej pracy jest zaprojektowanie, implementacja oraz przetestowanie aplikacji, która byłaby zdolna do kreowania, monitorowania oraz analizy zdefiniowanych procesów złożonych z postępujących po sobie zadań występujących w przedsiębiorstwie. Co więcej, spełniałaby rolę magazynu informacji o klientach oraz miała możliwość bezpośredniego kontaktu z nimi z poziomu systemu.

Całość miałaaby za zadanie łatwe i wygodne zaadaptowanie każdej firmy, która ma ściśle zdefiniowane procesy do podejścia zwinnego spełniając element wirtualizacji organizacji za pomocą aplikacji działającej w chmurze na platformie CRM [4], dostępnej z każdego urządzenia posiadającego przeglądarkę stron w sieci World Wide Web lub dedykowaną aplikację na urządzenia z systemem Android lub iOS [5].

Przedsiębiorstwo posiadając pewne cele do wykonania, związane z dostarczaniem produktu lub usługi, powinno posiadać system, dzięki któremu jest w stanie kontrolować aktualnie wykonywane zadania, planować te, które ma do wykonania i sprawdzać czy dane cele zostały osiągnięte.

Jeśli procesy w organizacji są powtarzalne (jak na przykład sprzedaż produktu w sklepie internetowym) najlepszym rozwiązaniem byłoby jednokrotne zdefiniowanie takiego procesu i wielokrotne jego wykorzystywanie, dzięki czemu można by sprawdzać ile czasu zostało poświęcone na taki przebieg średnio i czy to rzeczywiście jest opłacalne dla firmy. W razie chęci zmian wynikających z nieopłacalności możliwość zmiany takiego elementu w prosty sposób i prowadzenie dalszej analizy jest bardzo ważne. Bez podobnego mechanizmu przedsiębiorstwo nie ma wglądu na to jak funkcjonuje i jakie elementy można w nim poprawić.

Sam proces będąc złożonym z pewnych postępujących po sobie elementów wymaga pewnej reprezentacji graficznej. Jeśli proces posiadałby jedynie zwykły opis tekstowy może to wprowadzić niejasności lub być zbyt zawile, aby go zrozumieć czy też zrealizować go w uporządkowany sposób.

Dodatkowo osoba zarządzająca nie mając wiedzy o aktualnie wykonywanych zadaniach nie jest w stanie zareagować. Kontakt z zaniedbanym klientem, zbyt duża lub zbyt mała ilość godzin pracy pracowników czy też zagospodarowanie ich czasu wydają się być istotnymi problemami funkcjonowania przedsiębiorstwa. Dzięki prowadzeniu procesów w systemie, jeśli pracownik miałby dostęp do aktualizowania swoich zadań z poziomu własnego konta, osoba zarządzająca byłaby w stanie śledzić jego postępy i w razie pojawienia się niepokojących elementów – podjąć odpowiednie akcje.

Co więcej, jeśli pracownik posiadałby wszystkie swoje zadania w jednym systemie w ustalonym miejscu, to zmniejszyłaby się szansa na to, że przeoczy któregoś z zadań. Takie monitorowanie swoich postępów i przegląd nadchodzących obowiązków, może pomóc mu w organizowaniu swojego czasu pracy i w konsekwencji zwiększyć jego wydajność. Przychodzące powiadomienia o aktualnościach, zbliżających się zadaniach w danym dniu lub zagrożeniu związanym z możliwością przekroczenia wyznaczonego czasu zakończenia zadania, pozwoliła by się ustrzec od niezadowolenia klientów z dostarczanych usług poprzez szybkie reagowanie pracowników.

Niniejsza praca opisuje aplikację rozwiązującą powyższe problemy i w jej zakres wchodzi:

- przeprowadzenie przeglądu istniejących rozwiązań na rynku dostarczających podobne rozwiązania opisane w tym rozdziale
- zdefiniowanie wymagań wraz ze scenariuszami przypadków użycia
- przedstawienie projektu rozwiązania z wyszczególnionymi modułami użytkowymi zrealizowanymi przy pomocy platformy CRM Salesforce.com działającej w chmurze
- opisanie szczegółów dotyczących implementacji aplikacji na platformie wraz z użytymi technologiami, narzędziami oraz dobrymi praktykami pozwalającymi na rozwój aplikacji
- przedstawienie modelu danych, użytych wzorców projektowych oraz metodę walidacji acykliczności grafu skierowanego przy budowaniu procesu z zadań
- przedstawienie metod testowania aplikacji z wynikami oraz ocena końcowa i wnioski z jej realizacji
- zaprezentowanie możliwych kierunków rozwoju zrealizowanej części systemu

1.3 Układ pracy

W niniejszym rozdziale przedstawione są motywacje autora do podjęcia tej pracy dyplomowej, krótki wstęp opisujący aktualne trendy na rynku, a także cel i zakres pracy. Następnie wyjaśnione są wykorzystywane skróty i pojęcia, które mogą wydać się nieoczywiste.

W rozdziale drugim przedstawione jest porównanie istniejących rozwiązań na rynku, które mogłyby spełniać podobną funkcję jak przedstawiony program.

Trzeci rozdział zawiera spis poszczególnych funkcji użytkowników, analizę ich potrzeb w formie scenariuszy przypadków użycia oraz sporządzone wymagania zarówno funkcjonalne jak i нефункционалне.

W rozdziale czwartym zdefiniowany jest ogólny projekt rozwiązania z podziałem na poszczególne moduły funkcjonalności z wyznaczeniem dostępów użytkownikom, opis wzorca Model-Widok-Kontroler, model danych oraz specyfikę wybranej platformy CRM wraz z jej ograniczeniami.

Rozdział piąty opisuje technologię używane do tworzenia aplikacji na platformie oraz narzędzia implementacyjne. Dodatkowo opisana jest używana metoda walidacji acyklicznego grafu skierowanego, jako procesu złożone z postępujących po sobie zadań oraz wzorzec projektowy delegacji – użyty w celu uproszczenia możliwego rozwoju programu bez napotkania się na problemy związane z ograniczeniami platformy.

Szósty rozdział poświęcony jest opisowi różnym metodom testowania aplikacji oraz przedstawia napisane scenariusze testowe i zaimplementowane testy jednostkowych, które sprawdzają pokrycie kodu.

W ostatnich rozdziałach opisane jest podsumowanie pracy z efektem końcowym i możliwościami dalszego rozwoju.

2 Omówienie istniejących rozwiązań

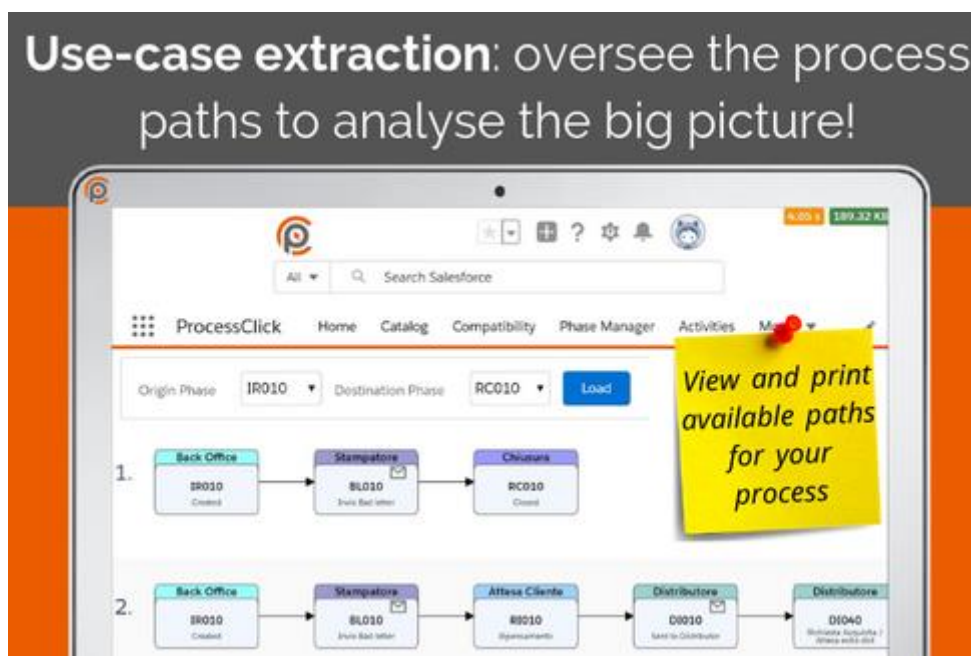
W ostatnich latach powstało bardzo wiele różnych platform CRM działających w chmurze. Sam skrót CRM (ang. Customer Relationship Management) jest to narzędzie informatyczne pomagające we wdrożeniu zbioru strategii i metod, które koncentrują się na tworzeniu długotrwałych związków z klientem, służących zwiększeniu efektywności sprzedaży oraz zmniejszeniu kosztów [4]. Należy jednak zaznaczyć tylko nieliczne mają możliwość tworzenia spersonalizowanych aplikacji i dzielenie się nimi w autoryzowanym sklepie platformy. Liderami na tym rynku są dwie firmy Salesforce.com ze sklepem AppExchange – którą wybrałem, oraz Microsoft Dynamics CRM z Microsoft AppSource. Na pierwszej z dwóch wymienionych w maju 2018 roku istniało ponad 3,400 aplikacji [6] dotyczących różnych dziedzin i liczba ta ciągle się powiększa, druga zaś ma ponad 1800 aplikacji [7] lub rozszerzeń do produktu potęgi z doliny krzemowej. Naturalnym jest, że trudno jest sprawdzić wszystkie programy dostępne w danych sklepach, jednak udało mi się znaleźć aplikację spełniającą podobną funkcję: ProcessClick - Your Agile BPM for Salesforce.

Istnieje również wiele aplikacji dzięki którym można budować procesy złożone z zadań, jednak nie są one zintegrowane z żadną dużą platformą CRM, co ogranicza możliwości rozwoju oraz funkcjonalności, dzięki którym można przeprowadzić proces prowadzący do wirtualności przedsiębiorstwa, największymi z takich aplikacji są: Oracle Process Cloud i IBM Business Automation Workflow. Każde z rozwiązań zostało ocenione w trzystopniowej skali (złe/średnie/dobre) używając tabeli kryteriów.

2.1 ProcessClick - Your Agile BPM for Salesforce

ProcessClick to aplikacja do kupienia w ramach platformy Salesforce.com służąca do automatyzacji procesów, która całkowicie wspiera użytkownika we wdrażaniu i zarządzaniu złożonymi procesami biznesowymi i działaniami. Według producenta aplikacja zapewnia następujące funkcjonalności [8]:

- logiczne reprezentowanie automatu stanów oraz konfiguracja, opisywanie i sterowanie pełnym cyklem życia każdego procesu biznesowego
- kontekstowa lista wszystkich dostępnych procesów
- automatyczne tworzenie i przypisywanie zadań opartych na szablonie podczas wykonywania procesu
- wbudowana usługa do obsługi wiadomości przychodzących / wychodzących z systemami zewnętrznymi



Rysunek 2.1 Materiał reklamowy przedstawiający fragment procesu w ProcessClick. [8]

Na podstawie wymienionych funkcjonalności można powiedzieć, że rozwiązuje ona część problemów przedstawionych we wstępie pracy. Logiczne reprezentowanie automatu stanów oraz kontekstowa lista pomogą pracownikowi w zrozumieniu zadania jakie ma do wykonania oraz pilnowanie zadań, które ma zaplanowane. Co więcej, przez to że jest na platformie Salesforce.com interfejs jest przejrzysty, posiada elementy CRM oraz integracja z innymi aplikacjami (niekoniecznie na platformie) jest prosta. Główną wadą tej aplikacji jest jej poziom skomplikowania w użytkowaniu i obsługiwaniu zadań. Program wydaje się być przeznaczony do naprawdę złożonych procesów biznesowych, integracji ze systemami wiadomości i posiada sporo skomplikowanych automatyzacji. Przez to, że organizacje, które chcą wdrożyć elementy zwinne mają najprawdopodobniej pracowników nieobytymi z podobnymi systemami, tak wielowarstwowe narzędzie będzie

zbyt niezrozumiałe i zbyt zawile do nauczenia się przy adaptacji nowych aspektów funkcjonowania.

Kryterium	Ocena
praca nad wyszkoleniem pracowników	średnia
przejrzystość interfejsu	dobra
prostota użytkowania	zła
elementy CRM przy wirtualizacji przedsiębiorstwa	dobra
skomplikowanie graficznej reprezentacji procesu	zła
możliwość integracji z zewnętrznymi systemami	dobra

Tabela 2.1 Ocena rozwiązania ProcessClick

2.2 Oracle Process Cloud

Oracle BPM i Process Cloud to zintegrowane narzędzia od firmy Oracle Corporation służące do tworzenia, monitorowania i analizy procesów w notacji BPMN. Do głównych zalet aplikacji należą [9]:

- silne możliwości analityczne, które zwiększają widoczność wydajności procesu i inteligentniej koordynują procesy dzięki analizom w czasie i obsłudze zdarzeń
- pulpity nawigacyjne i inne raporty wywiadu operacyjnego, który może wykorzystywać dane z różnych systemów źródłowych, w tym zarówno Oracle
- integrację z innymi produktami Oracle, w tym Oracle SOA Suite i produktami Oracle ERP, a także garść adapterów do aplikacji innych firm (takich jak Salesforce i SAP)
- wysoką dostępność i skalowalność przy mniejszej zależności od programistów IT oraz personelu zajmującego się operacjami i konserwacją
- aplikacje zbudowane na usłudze w chmurze można migrować do lokalnej wersji produktu



Rysunek 2.2 Wygląd aplikacji Oracle Process Cloud na różnych urządzeniach. [9]

Rozwiązanie to może być naprawdę przydatne jeśli omawiana firma korzysta z innych produktów firmy Oracle Corporation, integracja takiego rozwiązania jest zautomatyzowana, a interfejs aplikacji wydaje się prosty i przyjazny użytkownikowi.

Należy jednak nadmienić, że narzędzie to służy do projektowania procesów w notacji BPMN, co może wymagać przez pracowników danego przedsiębiorstwa nauki danej notacji. Jeśli rozważymy przypadek w którym firma nie jest z branży technologii informacyjnej, problem może pogłębić się do niezbędnych szkoleń, a i tak nie być do końca rozwiązany przez „nietechniczny” charakter branży i pracowników. Rozwiązanie zatem ma wysoki próg wejścia dla przeciętnej firmy niefunkcjonującej w dziedzinach informatyki w porównaniu do protego grafu skierowanego zbudowanego z wierzchołków i krawędzi.

Co więcej dane rozwiązanie nie jest związane z żadną platformą CRM, przez co rozwój jest ograniczony do produktów firmy Oracle i możliwych integracji, oraz trudniej jest stworzyć pewien społecznościowy model funkcjonowania takiej firmy, gdzie jednym z założeń wirtualizacji może być możliwość prowadzenia pewnych grup projektowych, dyskusyjnych czy nawet hobbystycznych w ramach użytkowania aplikacji jak sieci społecznościowej. Takie działania budują pewną kulturę firmy oraz podwyższają morale pracowników – takie rozwiązanie zapewnia moduł o nazwie Chatter na platformie Salesforce [10].

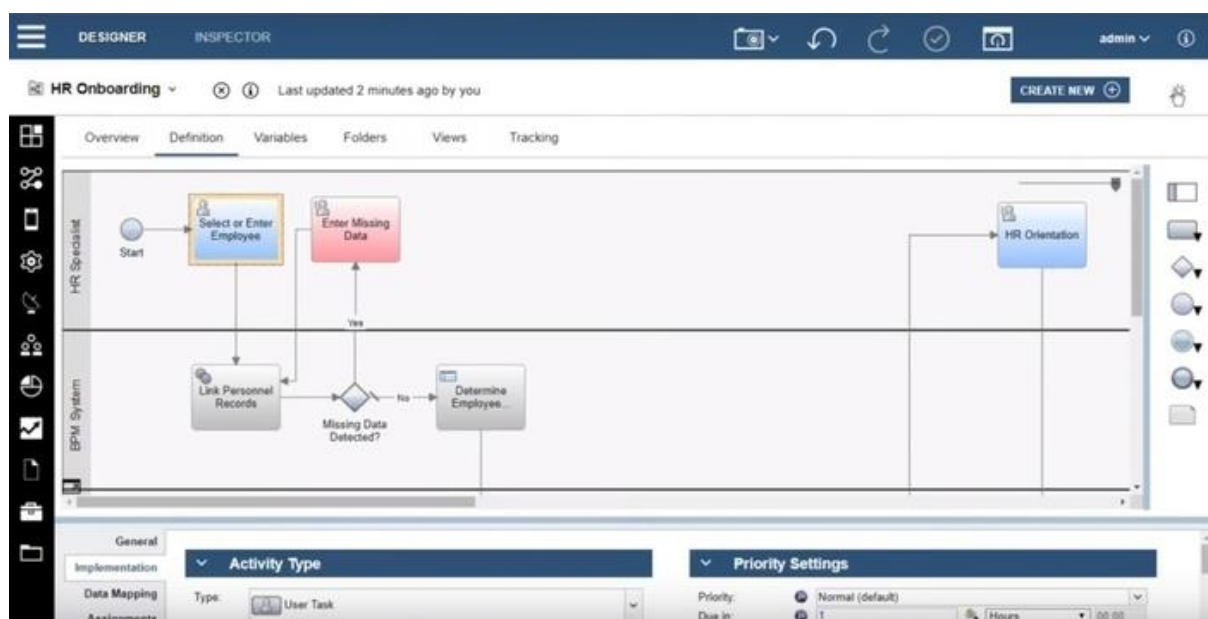
Kryterium	Ocena
praca nad wyszkoleniem pracowników	zła
przejrzystość interfejsu	średnia
prostota użytkowania	średnia
elementy CRM przy wirtualizacji przedsiębiorstwa	zła
skomplikowanie graficznej reprezentacji procesu	średnia
możliwość integracji z zewnętrznymi systemami	średnia

Tabela 2.2 Ocena rozwiązania Oracle Process Cloud

2.3 IBM Business Automation Workflow

Rozwiązanie służące do projektowania procesów w notacji BPMN innego giganta technologicznego. Według opinii użytkowników można nadmienić następujące jego zalety [11]:

- daje możliwość tworzenia zestawów narzędzi i korzystania z nich w wielu różnych aplikacjach w formie wykorzystywania szablonów
- zintegrowany interfejs użytkownika i modele wdrażania, a także bogaty zestaw funkcji ekosystemu konsultantów i dostawców usług
- nawet bez zaplecza technicznego, możliwość szybkiego poznania aplikacji i przejścia do jej używania już po kilku minutach
- połączenie między tak zwanym silnikiem reguł a jego innym produktem – podstawową platformą BPM, dzięki czemu jest to scentralizowany pakiet jako jedna oferta



Rysunek 2.3 Klatka z filmu prezentującego demo aplikacji IBM Business Automation Workflow. [11]

Podobnie jak w przypadku rozwiązania Oracle, jest to aplikacja skoncentrowana na obsługę procesów w profesjonalnej notacji oraz łatwo integruje się z innymi produktami firmy, która ją stworzyła w ramach jednego dużego zestawu systemów. Jednak należy nadmienić, że ma dużą przewagę nad rozwiązaniem Oracle przez prostszy interfejs i lepiej zorganizowane funkcjonalności, przez co rozpoczęcie pracy używając tej aplikacji wymaga mniej czasu i umiejętności technicznych. Jednak, podobnie jak w poprzednim przykładzie, biorąc pod uwagę rozwiązanie na platformie wykorzystanej przeze mnie w pracy, aplikacja ta nie ma wielu funkcjonalności dzięki którym proces wdrażania elementów zwinnych jest trudniejszy. Co więcej według opinii użytkowników istnieją obecnie niespójności między samym Business Automation Workflow w połączeniu z ich główną platformą BPM i powinna być to jeden nierozłączny system – tak jak w przypadku systemu Salesforce.

Kryterium	Ocena
praca nad wyszkoleniem pracowników	średnia
przejrzystość interfejsu	dobra
prostota użytkowania	dobra
elementy CRM przy wirtualizacji przedsiębiorstwa	zła
skomplikowanie graficznej reprezentacji procesu	średnia
możliwość integracji z zewnętrznymi systemami	średnia

Tabela 2.3 Ocena rozwiązania IBM Business Automation Workflow

3 Analiza potrzeb użytkowników

Głównymi odbiorcami programu byłyby mniejsze przedsiębiorstwa, które mają chęć wdrożyć w swoje struktury elementy zwinnego funkcjonowania. Zakładając małe zaznajomienie z podobnymi systemami, aplikacja powinna mieć instynktowny, przejrzysty interfejs. Funkcjonalności przeznaczone są firmom, które mają zdefiniowane procesy

stworzone z pewnych postępujących po sobie zadań, niezależnie od branży jaką się zajmują.

3.1 Użytkownicy

W większości małych organizacji można wyróżnić dwie główne role użytkowników: menadżera oraz pracownika. Oczywiście możliwości osoby wyżej położonej w hierarchii pracowniczej powinny być większe – osoba taka akceptuje pewne kwestie związane z funkcjonowaniem firmy, śledzi elementy związane z wydatkami i wydajnością, aby móc w przyszłości decydować o krokach poprawiających rentowność. Dostęp pracowników powinien zaś ograniczyć się do samodzielnej organizacji pracy, która jest kluczowym elementem w funkcjonowaniu organizacji zwinnej.

3.2 Wymagania funkcjonalne

Wymagania funkcjonalne zostały przedstawione z wykorzystaniem przypadków użycia (ang. use cases). Wymagania dotyczą aplikacji na platformie, której użytkownikiem może być pracownik przedsiębiorstwa na różnych szczeblach hierarchii, co zostało wyróżnione przy danym aktorze. Program powinien zapewniać możliwość kreowania, monitorowania oraz analizy zdefiniowanych procesów złożonych z postępujących po sobie zadań występujących w przedsiębiorstwie.

Kreowanie wyżej wymienionego procesu powinno opierać się na definiowaniu pojedynczych zadań w postaci obiektów posiadających stałe elementy takie jak: nazwa, pracownik odpowiedzialny za zadanie, klient, którego zadanie się tyczy, ostateczna data zakończenia czy planowany czas realizacji wraz wyznaczeniem poprzedników i następników w ramach procesu przy pomocy graficznego interfejsu.

Monitorowanie powinno zapewniać użytkownikowi łatwy dostęp do aktualnie wykonywanych i zaplanowanych zadań, które są do niego przypisane. Co więcej system powinien informować użytkownika o zbliżających się zadaniach z bliskim terminem ostatecznej daty zakończenia oraz tych które powinny być wykonane danego dnia.

Analiza wykonanych procesów powinna pozwalać na przegląd podstawowych informacji i statystyk o skończonych zadaniach użytkownika lub wszystkich pracowników, jeśli użytkownikiem jest menadżer. Pracownicy powinni mieć wzgląd do automatycznie stworzonych raportów czasowych po aktualizacji zraportowanego czasu na dane zadanie. Dodatkowo należałoby wyróżnić takie statystyki jak procent niedoszacowanych zadań, porównanie miesięcznych zarobków użytkownika na podstawie liczby wypracowanych godzin, czy też koszty z podziałem na pracowników, klientów lub ich branże.

Jeśli w przedsiębiorstwie występują powtarzalne procesy, należy dać możliwość tworzenia tak zwanych szablonów. Procesów, które definiuje się jedynie raz, a następnie istnieje możliwość prostego tworzenia z nich identycznych instancji przebiegu w celu jego realizacji.

Aplikacja powinna zawierać informacje o klientach, przetrzymując je jako rekordy, oraz mieć możliwość bezpośredniego kontaktu z nimi w formie wysyłki email na samej platformie.

Każdy użytkownik ma mieć możliwość organizacji w niej swojej pracy w postaci utrzymywania notatek, zadań, zaplanowania tak zwanych akcji powiązanych z obiektami w systemie (na przykład rekord przypominający o wysyłce wiadomości email do klienta czy wykonania rozmowy telefonicznej do współpracownika w sprawie zadania). W systemie powinien również znajdować się kalendarz z możliwością tworzenia w nim wydarzeń – pomagającym w organizacji dnia pracownikowi. Wydarzenie oraz akcje powiązaną można by połączyć z obiektami systemu takimi jak na przykład: klient, zadanie, proces, użytkownik. Dzięki temu użytkownik jest w stanie lepiej kontrolować swoje obowiązki planując kolejne akcje z nimi związane.

Przypadek użycia: UC01: Rejestracja użytkownika
Aktorzy: Użytkownik (Menadżer), Użytkownik (Pracownik)
Przebieg w krokach:
<ol style="list-style-type: none"> 1. Menadżer wybiera opcję rejestracji nowego pracownika. 2. Menadżer podaje adres e-mail pracownika. 3. Pracownik rejestruje się do systemu przy pomocy linku otrzymanego w wiadomości. 4. Pracownik podaje dane wymagane do rejestracji w systemie. 5. Pracownik zakłada konto.

Tabela 3.1 I przypadek użycia

Przypadek użycia: UC02: Logowanie użytkownika
Aktorzy: Użytkownik
Przebieg w krokach:
<ol style="list-style-type: none"> 1. Użytkownik wpisuje poprawne dane do zalogowania. 2. Użytkownik wybiera opcje zalogowania. 3. System pozytywnie weryfikuje podane przez użytkownika dane 4. Użytkownikowi ukazują się elementy dostępne tylko dla zalogowanych.
Sytuacje wyjątkowe:
<ul style="list-style-type: none"> • podane dane nie pasują do żadnego z zarejestrowanych użytkowników – system wyświetla komunikat błędu

Tabela 3.2 II przypadek użycia

Przypadek użycia: UC03: Rejestracja klienta
Aktorzy: Użytkownik
Przebieg w krokach:
<ol style="list-style-type: none"> 1. Tak jak w UC02. 2. Użytkownik wybiera opcje tworzenia klienta. 3. Użytkownik wypełnia wymagane informacje dotyczące klienta. 4. Użytkownik zapisuje klienta w systemie.

Tabela 3.3 III przypadek użycia

Przypadek użycia: UC04: Utworzenie instancji procesu
Aktorzy: Użytkownik
Przebieg w krokach: <ol style="list-style-type: none"> 1. Tak jak w UC02. 2. Użytkownik wybiera opcje tworzenia procesu. 3. Użytkownik wypełnia wymagane informacje dotyczące procesu. 4. Użytkownik zapisuje proces w systemie.

Tabela 3.4 IV przypadek użycia

Przypadek użycia: UC05: Utworzenie zadań w procesie
Aktorzy: Użytkownik
Przebieg w krokach: <ol style="list-style-type: none"> 1. Tak jak w UC04. 2. Użytkownik wchodzi na stronę procesu. 3. Użytkownik inicjuje edycję procesu. 4. Użytkownik tworzy i zapisuje zadania w ramach procesu.

Tabela 3.5 V przypadek użycia

Przypadek użycia: UC06: Definiowanie kolejności zadań w procesie
Aktorzy: Użytkownik
Przebieg w krokach: <ol style="list-style-type: none"> 1. Tak jak w UC05. 2. Użytkownik inicjuje lub modyfikuje kolejność zadań w procesie.
Sytuacje wyjątkowe: <ul style="list-style-type: none"> • użytkownik próbuje połączyć zadania już połączone – system wyświetla komunikat błędu • użytkownik próbuje połączyć zadania tworząc skierowany graf acykliczny – system wyświetla komunikat błędu • użytkownik próbuje połączyć niezapisane zadania – system wyświetla komunikat błędu

Tabela 3.6 VI przypadek użycia

Przypadek użycia: UC07: Utworzenie szablonu procesu
Aktorzy: Użytkownik
Przebieg w krokach: <ol style="list-style-type: none"> 1. Tak jak w UC02. 2. Użytkownik wybiera opcje tworzenia procesu. 3. Użytkownik wypełnia wymagane informacje dotyczące procesu. 4. Użytkownik zapisuje proces w systemie jako szablon.

Tabela 3.7 VII przypadek użycia

Przypadek użycia: UC08: Utworzenie szablonów zadań
Aktorzy: Użytkownik
Przebieg w krokach: <ol style="list-style-type: none"> 1. Tak jak w UC07. 2. Użytkownik schodzi na stronę szablonu. 3. Użytkownik inicjuje edycję szablonu. 4. Użytkownik tworzy i zapisuje zadania oraz ich hierarchie w ramach procesu. 5. System zapisuje zadania jako szablony.

Tabela 3.8 VIII przypadek użycia

Przypadek użycia: UC09: Utworzenie instancji procesu z szablonu
Aktorzy: Użytkownik
Przebieg w krokach: <ol style="list-style-type: none"> 1. Tak jak w UC02. 2. Użytkownik przechodzi na listę szablonów. 3. Użytkownik wybiera z listy istniejący w systemie szablon. 4. Użytkownikowi ukazuje się strona szablonu. 5. Użytkownik wybiera opcję stworzenia procesu z szablonu. 6. Użytkownikowi ukazują się strona instancji procesu wygenerowanego z szablonu.

Tabela 3.9 IX przypadek użycia

Przypadek użycia: UC10: Rejestracja wydarzenia w kalendarzu
Aktorzy: Użytkownik
Przebieg w krokach: <ol style="list-style-type: none"> 1. Tak jak w UC02. 2. Użytkownik wchodzi do widoku kalendarza. 3. Użytkownik wybiera dzień w którym chce zarejestrować wydarzenie w kalendarzu. 4. Użytkownik konfiguruje wydarzenie wypełniając wymagane pola. 5. Użytkownik zapisuje wydarzenie w kalendarzu.
Przebieg alternatywny: <ol style="list-style-type: none"> 1. Tak jak w UC02. 2. Użytkownik wchodzi do widoku dowolnego obiektu w systemie. 3. Użytkownik wybiera opcje rejestracji wydarzenia powiązanego z obiektem. 4. Użytkownik konfiguruje wydarzenie wypełniając wymagane pola. 5. Użytkownik zapisuje wydarzenie w kalendarzu.

Tabela 3.10 X przypadek użycia

Przypadek użycia: UC11: Rejestracja akcji powiązanej
Aktorzy: Użytkownik
Przebieg w krokach: <ol style="list-style-type: none"> 1. Tak jak w UC02. 2. Użytkownik wchodzi do widoku akcji. 3. Użytkownik wybiera opcje stworzenia nowej akcji. 4. Użytkownik konfiguruje akcje wypełniając wymagane pola. 5. Użytkownik zapisuje akcje w systemie.
Przebieg alternatywny: <ol style="list-style-type: none"> 1. Tak jak w UC02. 2. Użytkownik wchodzi do widoku dowolnego obiektu w systemie. 3. Użytkownik wybiera opcje rejestracji akcji powiązanej z obiektem. 4. Użytkownik konfiguruje akcje wypełniając wymagane pola. 5. Użytkownik zapisuje akcje w systemie.

Tabela 3.11 XI przypadek użycia

Przypadek użycia: UC12: Przegląd statystyk
Aktorzy: Użytkownik
Przebieg w krokach: <ol style="list-style-type: none"> 1. Tak jak w UC02. 2. Użytkownik wchodzi do widoku list statystyk. 3. Użytkownik wybiera dostępny widok statystyk z listy. 4. Użytkownikowi ukazują się wybrane statystyki.

Tabela 3.12 XII przypadek użycia

Przypadek użycia: UC13: Aktualizowanie zadania
Aktorzy: Użytkownik
Przebieg w krokach: <ol style="list-style-type: none"> 1. Tak jak w UC02. 2. Użytkownik przechodzi do widoku listy procesów. 3. Użytkownik wybiera dostępny proces z listy. 4. Użytkownikowi ukazuje się widok procesu. 5. Użytkownik wybiera edycja procesu. 6. Użytkownikowi ukazuje się widok zadań w procesie. 7. Użytkownik wybiera opcję edycji zadania. 8. Użytkownikowi pokazuje się ekran edycji zadania. 9. Użytkownik aktualizuje pola zadania. 10. Użytkownik zapisuje zmiany zadania.
Przebieg alternatywny: <ol style="list-style-type: none"> 9a. Użytkownik aktualizuje przeznaczony czas na zadanie. 10a. System tworzy lub aktualizuje raport czasowy użytkownika.
Sytuacje wyjątkowe: <ul style="list-style-type: none"> • użytkownik próbuje zaktualizować czas poświęcony na zadanie zakończone lub zaplanowane – system wyświetla komunikat błędu

- pracownik próbuje zaktualizować datę planowanego zakończenia zadania - system wyświetla komunikat błędu

Tabela 3.13 XIII przypadek użycia

Przypadek użycia: UC14: Podjęcie kontaktu z klientem
Aktorzy: Użytkownik
Przebieg w krokach: <ol style="list-style-type: none"> 1. Tak jak w UC02. 2. Użytkownik przechodzi do widoku listy klientów.. 3. Użytkownik wybiera klienta z którym chce podjąć kontakt. 4. Użytkownik wybiera opcje wysłania e-maila z systemu. 5. Użytkownik wpisuje treść w podane pola wraz z treścią wiadomości. 6. Użytkownik wysyła wiadomość e-mail.

Tabela 3.14 XIV przypadek użycia

Przypadek użycia: UC15: Dodanie notatki
Aktorzy: Użytkownik
Przebieg w krokach: <ol style="list-style-type: none"> 1. Tak jak w UC02. 2. Użytkownik wchodzi do widoku dowolnego obiektu w systemie. 3. Użytkownik wybiera opcje tworzenia notatki powiązanej z obiektem. 4. Użytkownik konfiguruje notatkę wypełniając wymagane pola. 5. Użytkownik zapisuje notatki powiązanej z obiektem.

Tabela 3.15 XV przypadek użycia

Przypadek użycia: UC16: Prośba o przedłużenie ostatecznej planowanej daty zakończenia zadania
Aktorzy: Użytkownik (Pracownik), Użytkownik (Menadżer)
Przebieg w krokach: <ol style="list-style-type: none"> 1. Tak jak w UC02. 2. Pracownik przechodzi do widoku zadania. 3. Pracownik wypełnia wszystkie pola w prośbie o zmianę planowanej daty zakończenia zadania. 4. Pracownik wysyła prośbę. 5. Menadżer dostaje powiadomienie o otrzymanym procesie akceptacji. 6. Menadżer akceptuje prośbę o zmianę. 7. System aktualizuje planowaną datę zakończenia zadania.
Przebieg alternatywny: <ol style="list-style-type: none"> 6a. Menadżer odrzuca prośbę o zmianę. 7a. System informuje pracownika o odrzuceniu procesu akceptacji.

Tabela 3.16 XVI przypadek użycia

Przypadek użycia: UC17: Przegląd raportów czasowych
Aktorzy: Użytkownik
Przebieg w krokach: 1. Tak jak w UC02. 2. Użytkownikowi ukazuje się jego przegląd ostatnich raportów czasowych.

Tabela 3.17 XVII przypadek użycia

Przypadek użycia: UC18: Przegląd nadchodzących zadań
Aktorzy: Użytkownik
Przebieg w krokach: 1. Tak jak w UC02. 2. Użytkownikowi ukazuje się jego przegląd nadchodzących zadań przypisanych do użytkownika.

Tabela 3.18 XVIII przypadek użycia

3.3 Wymagania niefunkcjonalne

- łatwość użycia aplikacji bez znajomości specjalistycznych notacji
- bezpieczne – niemożliwy przegląd informacji bez dostępu do konta
- skalowalność
- otwartość – możliwość rozbudowy
- odporność na awarie

4 Projekt rozwiązania

Aby zrealizować wymagania przedstawione we wcześniejszych rozdziałach zdecydowałem się skonstruować aplikację usadzoną w chmurze na portalu CRM. Platforma Salesforce.com jest to światowy lider w swojej dziedzinie. Bardzo dobrze sprawdza się w zarządzaniu danymi i procesami biznesowo – sprzedażowymi. Standardowe narzędzia na platformie pozwalają automatyzować logikę aplikacji na podstawie warunków wejściowych. Na przykład aktualizacja pól na rekordzie lub wysyłka wiadomości email w momencie spełnienia zdefiniowanych warunków rekordu. Innym przykładem może być mechanizm akceptacji rekordu przez menadżera zdefiniowanego na rekordzie użytkownika, dzięki któremu to osoba postawiona wyżej w hierarchii decyduje o tym czy daną akcję można wykonać (na przykład zmiana ceny na produkcie, jeśli aplikacją byłby sklep). Tego typu elementy platformy sprawnie sprzyjają podejściu procesowemu pozwalając adaptować istniejące już (standardowe) funkcjonalności pod własne potrzeby odpowiednio je konfigurując. W połączeniu z możliwością tworzenia niestandardowej logiki oraz komponentów przy pomocy kodu, narzędzie to wydaje się odpowiednim do realizacji aplikacji spełniającej wymagania przedstawione w poprzednim rozdziale.

4.1 Specyfika aplikacji na platformie Salesforce

Projekt rozwiązania w pewnym stopniu różni się od standardowego projektowania aplikacji webowych. Na samej platformie istnieje już wiele aplikacji przygotowanych przez producenta wraz obiektami z ustaloną hierarchią (wiele obiektów jest wspólnych dla różnych aplikacji – tylko pełnią różną funkcję). Dzięki, temu klienci mają możliwość zakupu licencji od firmy na gotowe programy służące na przykład do: obsługi zgłoszeń serwisowych, zarządzania sprzedażą, zarządzania marketingiem, automatyzacji marketingu czy nawet zarządzania pacjentami i ich komunikacji z lekarzami. Jednak każda z tych aplikacji wymaga oprócz licencji za każdego użytkownika do korzystania z platformy, dodatkowej licencji per aplikacja. Z kosztem standardowej licencji do korzystania z platformy pochodzi zestaw podstawowych obiektów – z bardzo ograniczonymi powiązanymi funkcjonalnościami. Oczywiście można adaptować standardowe obiekty do swoich potrzeb, dodają do nich niestandardowe pola, oraz stworzyć własne, niestandardowe obiekty. Jednak istnieją pewne ograniczenia z tym związane, które zostaną opisane w rozdziale 4.5, dlatego dobrą praktyką jest jak największe wykorzystanie istniejących już obiektów jednocześnie pamiętając o tym, żeby nie używać elementów w sposób do którego nie są przeznaczone.

Co więcej pewien szkielet wyglądu aplikacji zawsze jest taki sam – są to zdefiniowane przez autora zakładki związane z obiektami lub funkcjonalnościami. Na każdej z zakładek można osobno zdecydować co ma się na niej znajdować. Ich zawartość definiuje się tworząc komponenty. Mogą być to tak zwane Lightning Web Components lub konfigurowalne standardowych elementy. Powyższe komponenty zaprojektowane są przy użyciu wzorca Model-Widok-Kontroler opisany poniżej, który jest kluczowy dla tworzenia uniwersalnych, elastycznych, łatwo modyfikowalnych programów i systemów. Najczęstszym standardowym komponentem jest taki, który opisuje ułożenie pól jakie mają się wyświetlać w przejrzysty sposób na instancji obiektu. Szczegóły dotyczące implementacji komponentów i ich komunikacji z logiką aplikacji zostały opisane w rozdziale piątym – Implementacja.

4.2 Wzorzec Model-Widok-Kontroler

Wzorzec ten został użyty do zaimplementowania niestandardowych komponentów (Lightning Web Components) składa się z trzech głównych elementów [12]:

- modelu - reprezentującego strukturę i logikę aplikacji,
- widoku - opisującego, w jaki sposób konkretna część modelu powinna być wyświetlana w interfejsie aplikacji i może zawierać wiele widoków podrzędnych dla jednego widoku
- kontrolera - akceptującego dane wejściowe, zarządzającego widokiem i aktualizującego dane wyjściowe

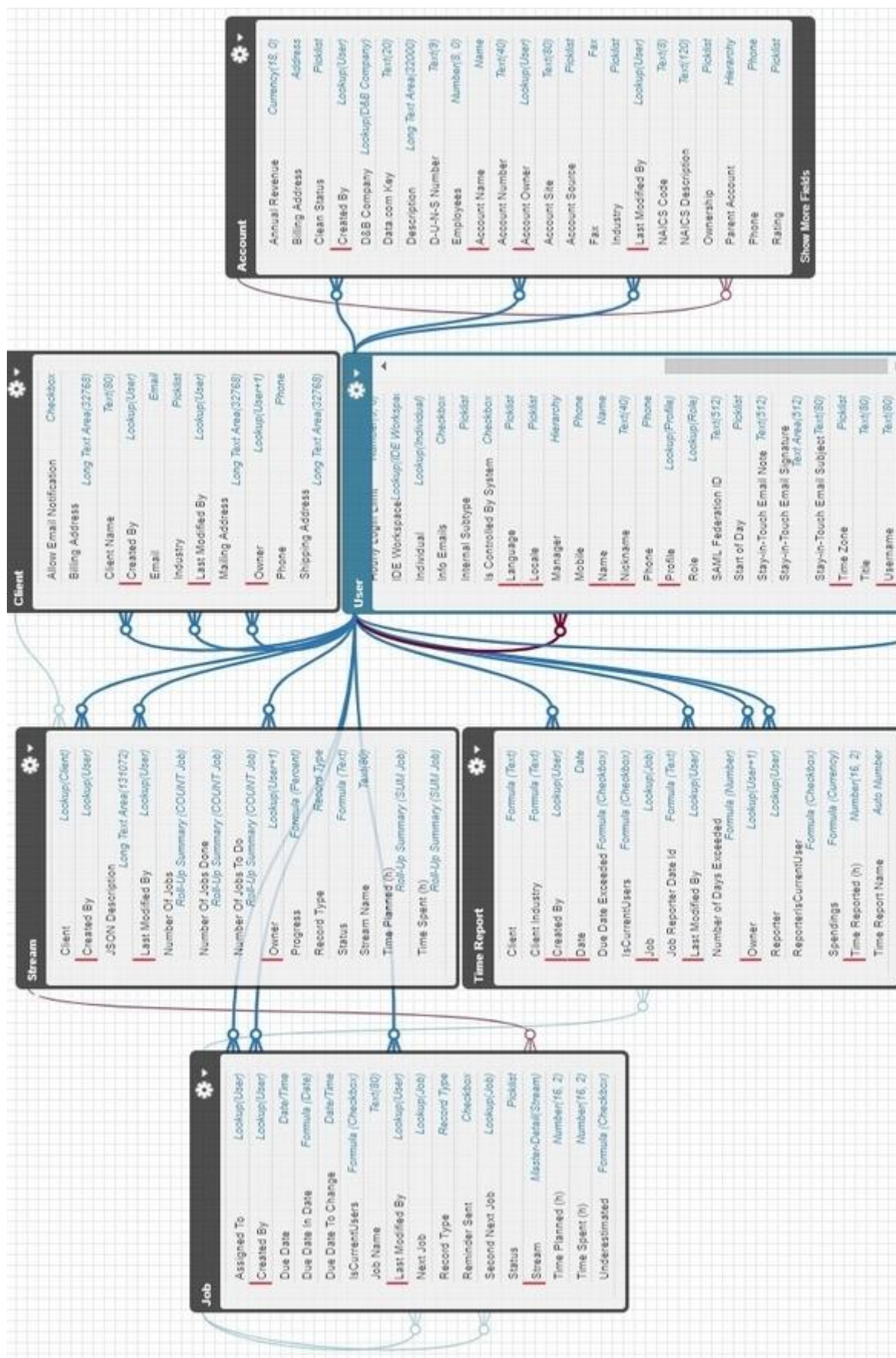
Tego typu rozwiązanie jest bardzo spójne i zrozumiałe. Dzięki niemu jeden komponent może być wielokrotnie używany przy lekkich modyfikacjach innych elementów wzorca. Wadą rozwiązania jest złożony wzór projektowy. Należy dobrze zrozumieć przepływ sterowania między modelem, widokiem i kontrolerem. Ponadto programista powinien posiadać wiedzę na temat wielu technologii.

Na platformie Salesforce Model-Widok-Kontroler jest wykorzystywany przez tak zwane Lightning Web Components. Są one responsywnymi, samowystarczalnymi elementami, które można ustawić na stronie. Dodatkowo można je łączyć w hierarchie – dzięki takiej modularności nie ma potrzeby pisać podobnych komponentów dwa razy [13].

4.3 Model danych

Na platformie jest również możliwość ukazania modelu danych z poziomu konfiguracji administratora, jednak należy nadmienić, że przy użyciu standardowych obiektów istnieje wiele pól, których w tym rozwiązaniu nie są używane, a nie da ich się ukryć ani usunąć na diagramie. Co więcej nazwy istniejących na platformie obiektów zostały zaadaptowane pod aplikację. W programie można wyróżnić następujące obiekty:

- Account (Entity) – zaadaptowany obiekt standardowy odpowiadający za gromadzenie informacji o przedsiębiorstwie
- Client – niestandardowy obiekt służący do utrzymywania bazy klientów, wraz ze szczegółami o nich, powiązanych akcji i dostarczanych dóbr lub usług opisanych procesami
- Stream – niestandardowy obiekt opisujący proces lub jego szablon włącznie ze statystykami, hierarchią zadań oraz informacją kogo się dotyczy
- Job – niestandardowy obiekt reprezentujący zadania z którego złożone są procesy, posiada szczegóły opisujące aktualny status, wycenę wraz z ostatecznym terminem zakończenia i liczbę zaraportowanych już godzin
- Time Report – niestandardowy obiekt odpowiadający za utrzymywanie informacji o czasie poświęconym na dane zadanie przez danego pracownika w danym dniu
- Event – standardowy obiekt platformy służący do rejestracji wydarzeń w kalendarzu
- Task – standardowy obiekt odpowiadający za zaplanowane i wykonane akcje powiązane z różnymi obiektami systemu
- User – standardowy obiekt reprezentujący użytkownika logującego się do systemu



Rysunek 4.1 Model danych aplikacji

Obiekty Event oraz Task łączą się relacją wiele do jeden z każdym istniejącym obiektem w aplikacji, więc w celu zachowania czytelności zostały pominięte na powyższym diagramie.

4.4 Architektura aplikacji i moduły funkcjonalności

Aplikacje tworzone na platformie Salesforce tak jak większość aplikacji internetowych posługują się architekturą trójwarstwową, w której każda z warstw (prezentacji, logiki biznesowej i bazy danych) jest oddzielnym modułem, który można zmodyfikować nie zmieniając reszty modułów. Dużą różnicą w porównaniu do klasycznej aplikacji internetowej jest brak możliwości podmiany modułu biorąc pod uwagę aspekt technologiczny. Przez to, że aplikacja napisana jest na platformie programista zmuszony jest implementacji warstwy logiki aplikacji oraz prezentacji w wyznaczonym języku. Jedyną możliwą modyfikacją w warstwie prezentacji, może być użycie zewnętrznych bibliotek bazujących na JavaScript wpierających platformę.

Od strony interfejsu użytkownika aplikacje tworzą niezwiązane ze sobą bezpośrednio komponenty odpowiedzialne za interakcję z użytkownikiem (elementy do wyświetlania i wprowadzania danych) komunikujące się z warstwą biznesową programu za pomocą kontrolerów. W warstwie biznesowej przygotowywane są dane do komponentów (warstwy prezentacji) oraz odpowiedzialna jest ona za komunikację z bazą danych stanowiącą ostatnią warstwę. Szczegóły dotyczące implementacji poszczególnych warstw zostały przedstawione w rozdziale 5.1.

W całym programie można wyróżnić trzy podstawowe rodzaje funkcjonalności: tworzenie procesów złożonych z zadań, monitorowanie postępów i statystyk klienckich oraz blok monitorowania i analizy menadżerskiej. Każdy z modułów odpowiada za szereg funkcjonalności, dzięki którym można wprowadzić elementy funkcjonowania zwinnego do mniejszego przedsiębiorstwa. Wszystkie te elementy bazują na wcześniej wspomnianych niezależnych od siebie komponentach.

Pokrycie niektórych przypadków użycia nie zostało przedstawione w poniższych modułach, ponieważ zostają spełnione przez podstawowe funkcjonalności platformy Salesforce. Przypadki UC01, UC02 dotyczący rejestracji i logowania użytkownika zostały spełnione poprzez standardowy mechanizm platformy i nie wymagały ingerencji programisty. Przypadki UC03, UC10, UC11 tyczą się tworzenia nowych rekordów poprzez wypełnienie formularzy na poszczególnych zakładkach lub obiektach w aplikacji i wymagały jedynie konfiguracji programisty. Natomiast przypadki UC14, UC15 tyczą się standardowych funkcjonalności Salesforce związanych z przetrzymywaniem plików na platformie i wysyłce wiadomości email

4.4.1 Tworzenie zadań oraz procesów

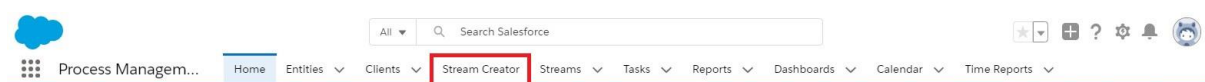
Jest to element aplikacji, dzięki któremu istnieje możliwość wygodnego i instynktownego tworzenia procesów z postępujących po sobie zadań. W jego ramach został zaimplementowany przejrzysty graficzny konfigurator umożliwiający w prosty sposób stworzenie lub edycje dowolnego procesu, niezależnie od tego czy jest on konkretną instancją czy często używanym szablonem, a składa się on z następujących elementów:

- zakładki kreatora, dzięki której w szybki sposób każdy z użytkowników może stworzyć dowolny proces złożony z postępujących po sobie zadań, który ułatwi mu organizację swojej pracy
- przejrzystym podglądzie z poziomu rekordu procesu, dzięki któremu można w mgnieniu oka ocenić jego postęp i hierarchie
- edytorze procesu, dostępnego z poziomu rekordu, który ma wszystkie funkcjonalności kreatora w ramach danej instancji

Interfejs graficzny jest jednak dostępny jedynie z poziomu wersji przeglądarkowej aplikacji. Wersja z aplikacji mobilnej nie ma możliwości tworzenia, edycji czy też oglądania hierarchii procesów w formacie grafu.

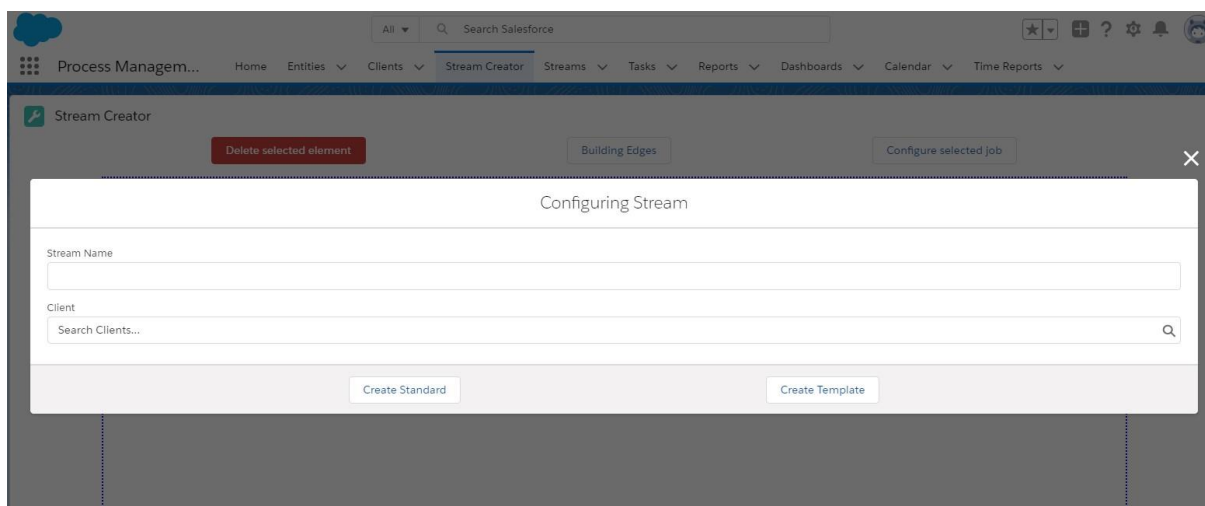
Zakładka kreatora

Z poziomu prawie każdego widoku, widoczne są wszystkie zakładki aplikacji do których można przejść w dowolnej chwili. W tym zakładka kreatora procesów oznaczona czerwonym kwadratem na rysunku poniżej:



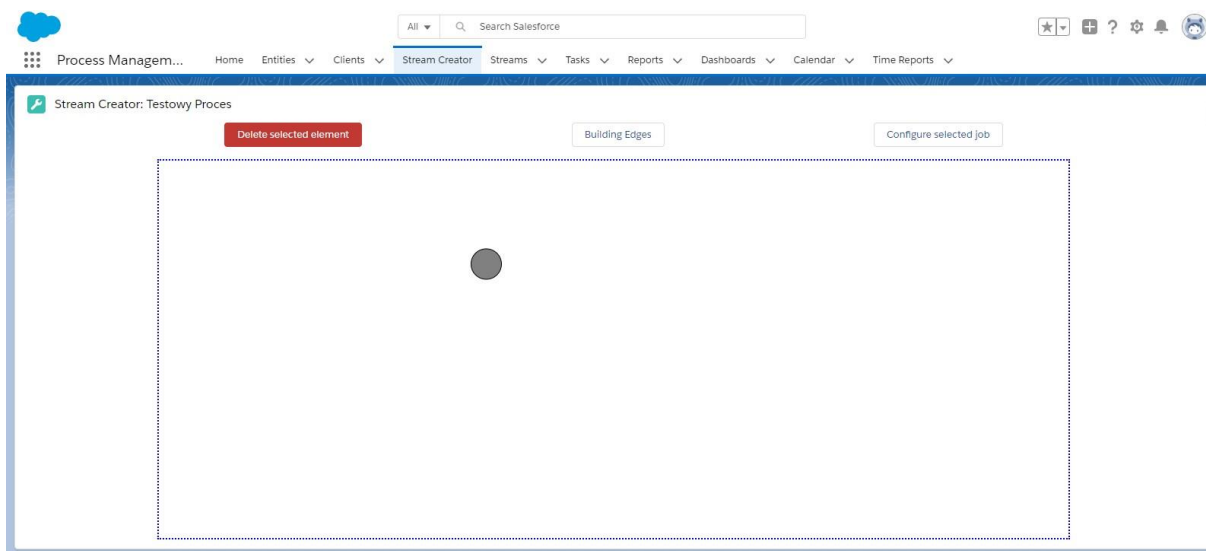
Rysunek 4.2 Widok zakładek aplikacji z oznaczeniem kreatora

Po przejściu na zakładkę otrzymujemy formularz tworzenia procesu (Rysunek 4.3), gdzie nadajemy mu nazwę i klienta, którego dotyczy, a następnie decydujemy czy tworzony ciąg zadań jest normalną instancją (UC04) czy powtarzalnym szablonem (UC07).



Rysunek 4.3 Formularz inicjujący proces

Wybranie jednej z dwóch opcji stworzenia przebiegu ukazuje użytkownikowi sektor dodawania zadań z trzema przyciskami w nagłówku kolejno: usunięciem wybranego zadania, przejścia do trybu tworzenia hierarchii zadań (UC06 i część UC08) i konfiguracji zadania (UC05 i część UC08). Aby dodać zadanie należy w puste pole na wyznaczonej strefie.



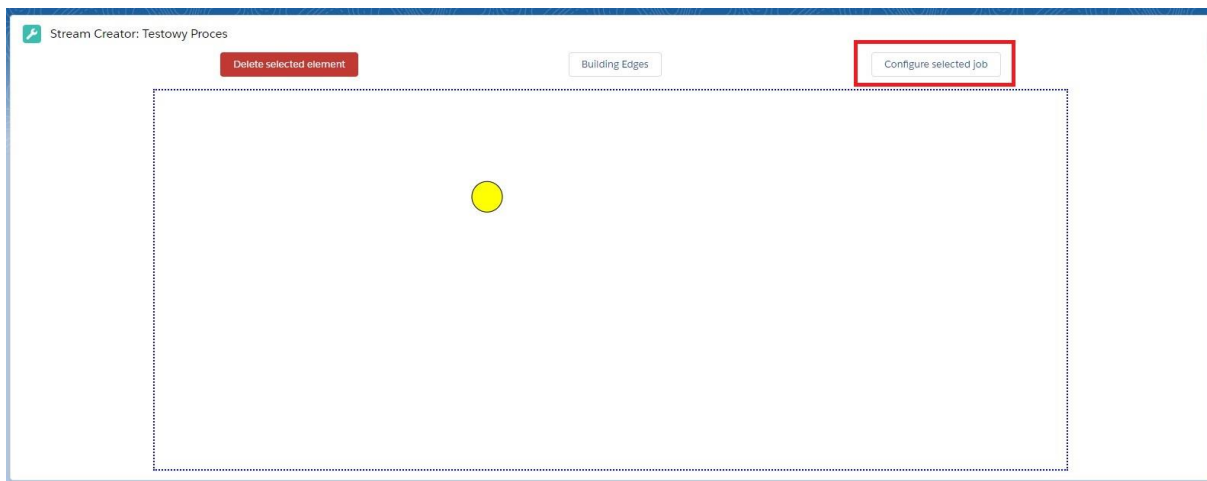
Rysunek 4.4 Dodanie wierzchołka do procesu

W tym momencie warto nadmienić, że różne kolory wierzchołków oznaczają różne stany w jakich może znajdować się zadanie. Legenda kolorów wierzchołków przedstawia poniższa tabela:

Kolor wierzchołka	Znaczenie wierzchołka
szary	nieskonfigurowane zadanie
żółty	zaznaczenie do dodania, edycji lub usunięcia
czerwony	skonfigurowane zadanie w statusie „TO DO”
niebieski	skonfigurowane zadanie w statusie „IN PROGRESS”
zielony	skonfigurowane zadanie w statusie „DONE”
fioletowy	zaznaczenie do dodania hierarchii między wierzchołkami

Tabela 4.1 Legenda znaczeń koloru wierzchołków w konfiguratorze procesów

Aby skonfigurować zadanie należy zaznaczyć dany wierzchołek klikając w niego, a następnie wybrać opcję konfiguracji wybranego zadania pod przyciskiem w nagłówku (UC05 i część UC08).



Rysunek 4.5 Zainicjowanie konfiguracji zadania w procesie

Configuring Job (Process Instance)

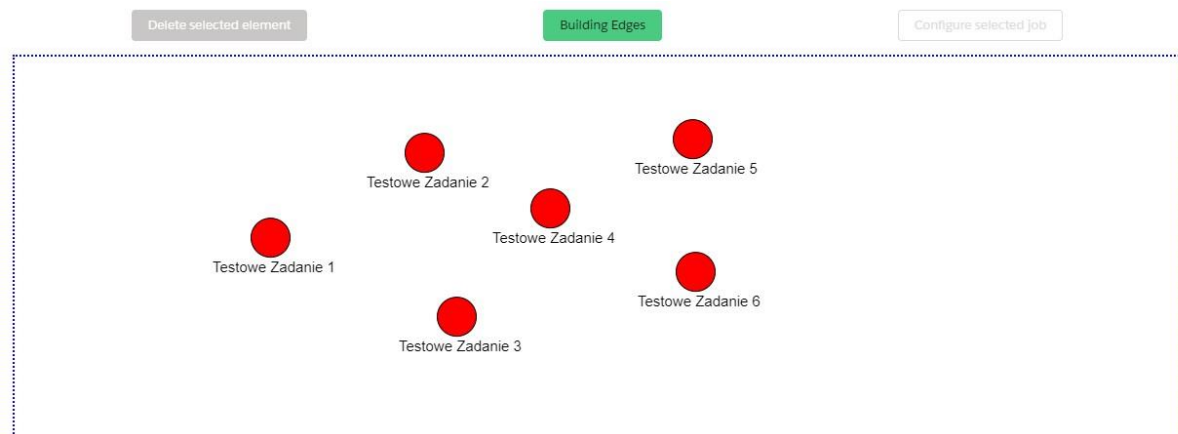
Job Name	Testowe Zadanie 1
* Stream	Testowy Proces
Time Planned (h)	4.00
Assigned To	Maciej Manager Employee
Time Spent (h)	0.00
Due Date	Date: 19-Oct-2019 Time: 15:42
Status	TO DO

Configuring Job (Template)

Job Name	
* Stream	Testowy Szablon
Time Planned (h)	0.00
Assigned To	Search People...

Rysunek 4.6 Porównanie formularzy zadań dla instancji procesu oraz szablonu

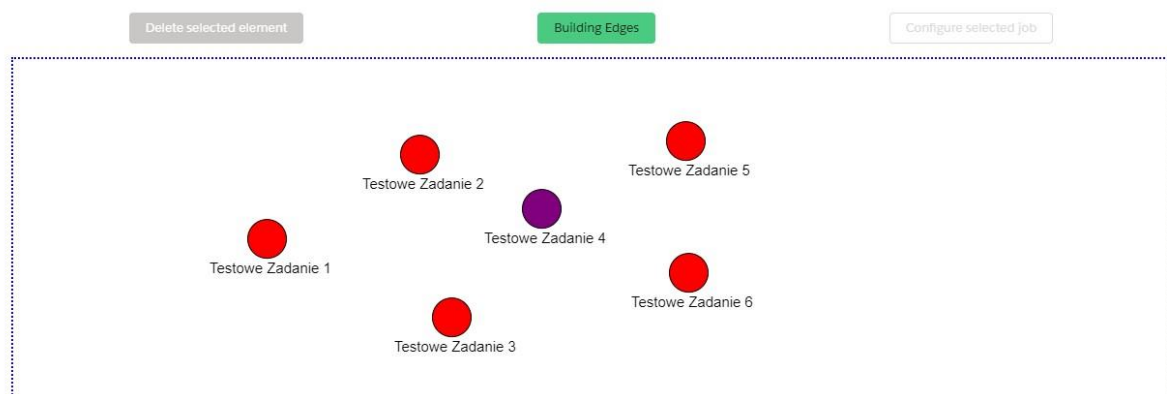
W celu dodania hierarchii pomiędzy zadaniami należy wybrać opcję przejścia do trybu dodawania krawędzi grafu (UC06 i część UC08).



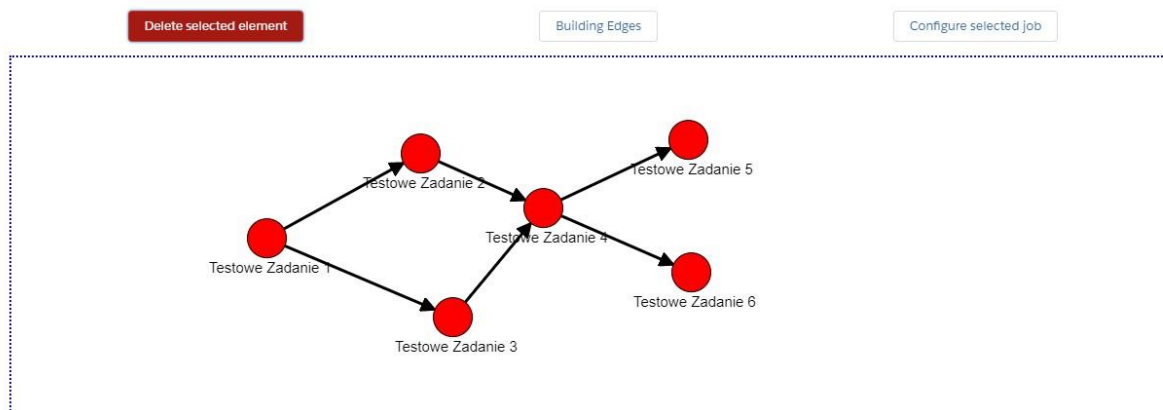
Rysunek 4.7 Konfigurator procesu w trybie dodawania krawędzi

Konfiguracja hierarchii polega na wybraniu wierzchołka od którego chce się poprowadzić krawędź oraz wierzchołka docelowego. W kontrolerze komponentu została zaprojektowana walidacja ochraniająca przed tworzeniem krawędzi (UC06 i część UC08):

- do wierzchołka z którego chce się poprowadzić krawędź
- pomiędzy wierzchołkami między którymi istnieje już krawędź
- jeśli stworzony w ten sposób skierowany graf będzie acykliczny (więcej w rozdziale 5.4)
- jeśli dane zadanie ma już dwie wychodzące krawędzie



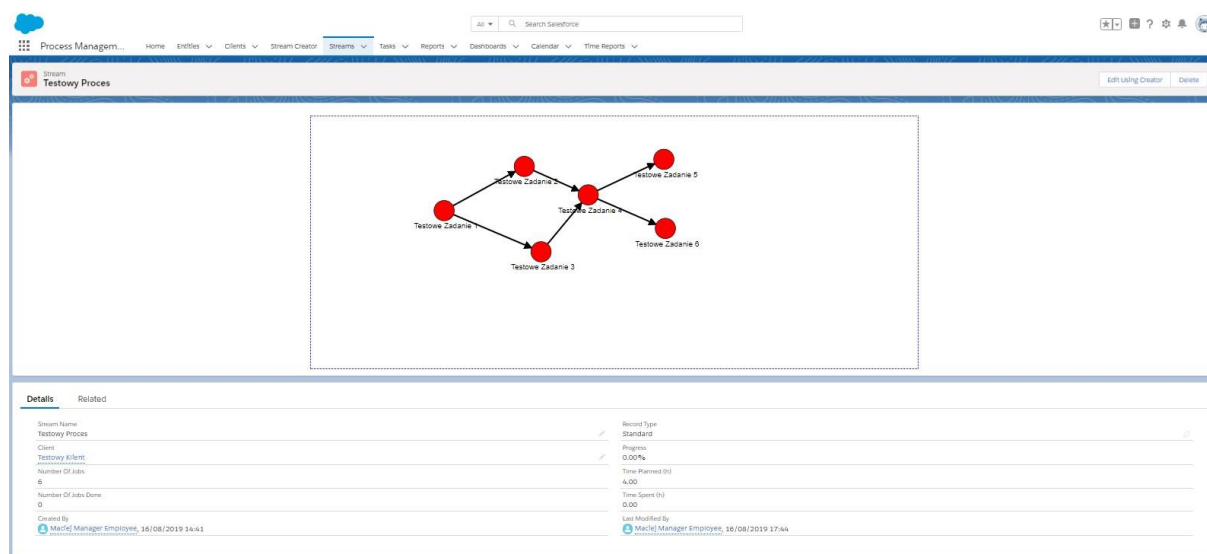
Rysunek 4.8 Zainicjowanie tworzenia krawędzi między wierzchołkami



Rysunek 4.9 Proces z utworzoną hierarchią zadań

Podgląd procesu

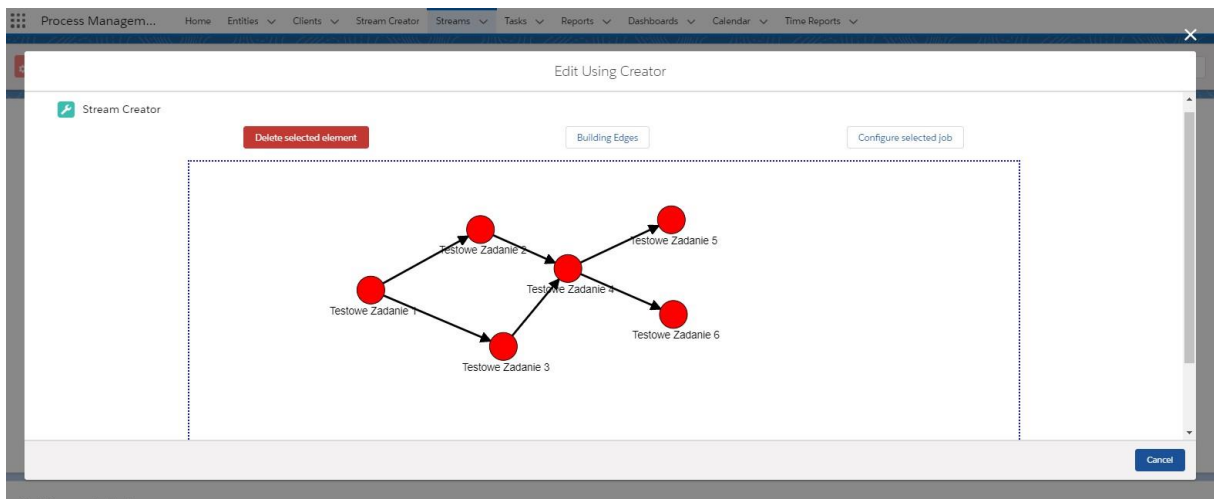
Z poziomu rekordu, w celu przejrzystego oglądu informacji na temat procesu został zaimplementowany nieedycyjny komponent wyświetlający połączone ze sobą zadania, a w dalszej części tego typu strony możemy zaobserwować standardowy komponent posiadający dwie zakładki: jedną opisującą jego najważniejsze statystyki i drugi opisujący powiązane elementy.



Rysunek 4.10 Strona rekordu procesu

Edytor procesu

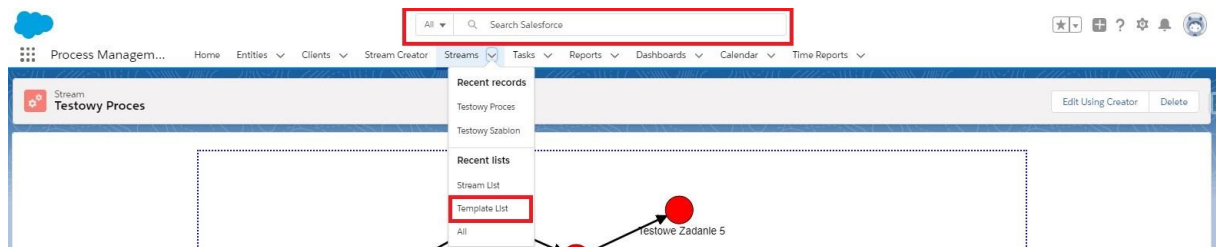
W celu edycji procesu za pomocą konfiguratora należy wejść na rekord procesu (przykładowy rekord procesu widnieje na rysunku 4.10), następnie wybrać opcję edycji przy pomocy kreatora w prawym górnym rogu strony. Po kliknięciu w przycisk, użytkownikowi ukaże się konfigurator w wersji do edycji. Wersja ta wygląda i działa identycznie jak opisany w segmencie opisującym kreator procesu – jedyną różnicą jest to, że edytor wyświetla się w wyskakującym oknie. Po zapisie zmian należy odświeżyć i automatycznym przejściu do strony rekordu należy odświeżyć stronę w celu zaobserwowania modyfikacji (UC13).



Rysunek 4.11 Edytor rekordu procesu przy pomocy konfiguratora

Tworzenie instancji procesu z szablonu

We wcześniejszej sekcji opisałem już tworzenie szablonu procesu. Zakładając, że przedsiębiorstwa mają ściśle zdefiniowane procesy, które mogą je zdefiniować właśnie jako wzór, bardzo ważna jest funkcjonalność polega na szybkim użyciu szablonu i przetworzeniu go na instancję przebiegu. Aby to wykonać należy wejść na rekord szablonu procesy przy pomocy wyszukiwarki na górze strony (wyszukując jego nazwę) lub w zakładce procesów wybrać listę zawierającą wszystkie szablony,



Rysunek 4.12 Opcje znalezienia szablonu procesu

Na rekordzie wzoru wystarczy wybrać opcję stworzenie instancji z szablonu w prawym górnym rogu ekranu. Spowoduje to sklonowanie procesu, zapisanie go w formacie instancji procesu (ze zmianą nazwy przebiegu oraz zadań dodając po nazwie szablonu dodatek identyfikujący) i przekierowanie na stronę właśnie stworzonego rekordu (UC09).

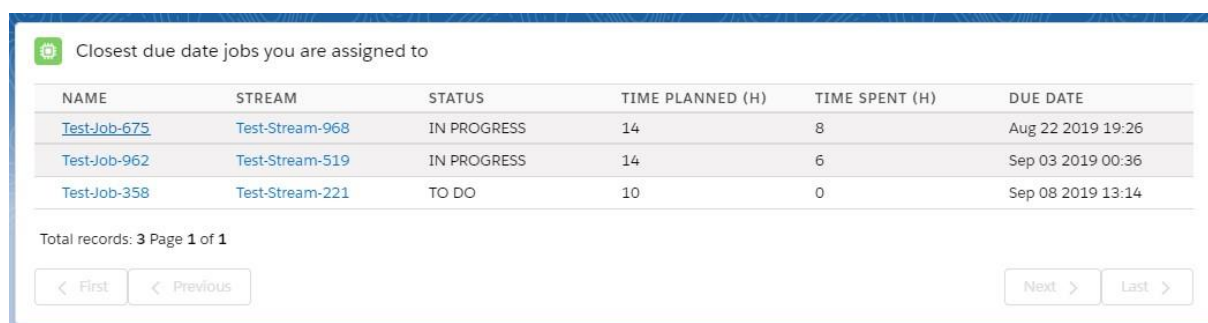
4.4.2 Monitorowanie postępów

Bardzo istotnym modulem w aplikacji pomagającej pracownikom w organizowaniu swojego czasu jest monitorowanie swoich postępów i najbliższych planów. Wylistowanie zbliżających się obowiązków i garść statystyk wydają się kluczowe w prawidłowym codziennym funkcjonowaniu każdego członka firmy. Przeniesienie całego planowania do jednego systemu z pewnością spełnia element wirtualizacji firmy, będący istotną częścią przedsiębiorstwa zwinnego.

Ekran główny

Tuż po zalogowaniu systemu użytkownikowi ukazuje się ekran główny, zawierający 4 komponenty (2 niestandardowe zaimplementowane i 2 standardowe platformy). Każdy z elementów odpowiada za inny obiekt informujący użytkownika o jego obowiązkach.

Niestandardowy komponent zawierający listę przypisanych do zalogowanego użytkownika nadchodzących zadań, posegregowanych po wyznaczonej dacie zakończenia zadania. Dzięki temu elementowi każda osoba logująca się do systemu jest w stanie szybko wyznaczyć priorytety na dany dzień, sprawdzić ile ma już rozpoczętych zadań czy też ile czasu ma zaplanowane na dane zadanie, a ile już przeznaczyła (UC18).



NAME	STREAM	STATUS	TIME PLANNED (H)	TIME SPENT (H)	DUE DATE
Test-Job-675	Test-Stream-968	IN PROGRESS	14	8	Aug 22 2019 19:26
Test-Job-962	Test-Stream-519	IN PROGRESS	14	6	Sep 03 2019 00:36
Test-Job-358	Test-Stream-221	TO DO	10	0	Sep 08 2019 13:14

Total records: 3 Page 1 of 1

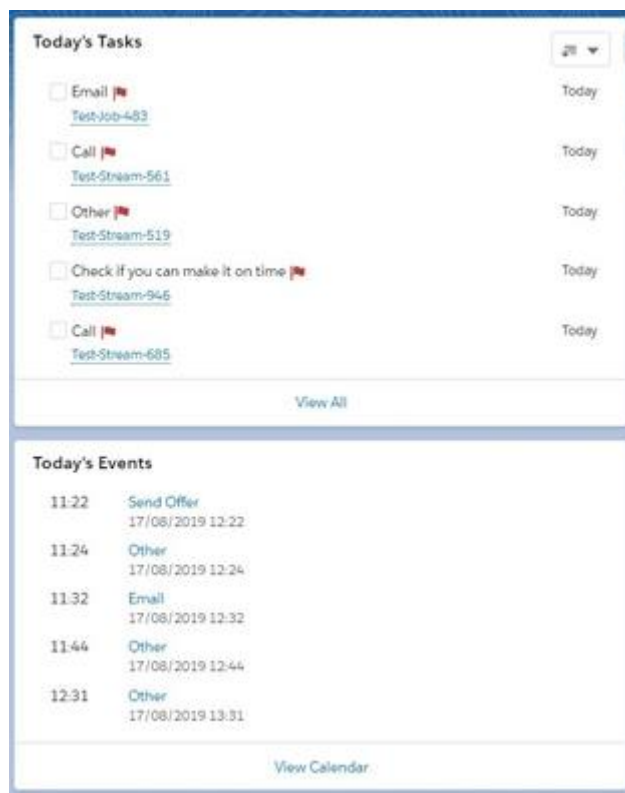
< First < Previous Next > Last >

Rysunek 4.13 Komponent z nadchodzącymi zadaniami użytkownika

Kolejnymi elementami są dwa standardowe komponenty oferowane przez platformę. Wyświetlają one zaplanowane na dziś akcje powiązane z różnymi obiektami w systemie oraz dzisiejsze wydarzenia w kalendarzu. Poniższa tabela opisuje dostępne wartości w polach akcji powiązanych. Dzięki rozróżnieniu na różne typy, statusy i priorytety jesteśmy w stanie śledzić czynności jakie mamy wykonać w danym dniu. Czasem mogą być one związane z konkretnym klientem, a innym razem z zadaniem lub całym procesem. Wartości te dają pogląd na to w jaki sposób można planować swój dzień pracy.

Pole akcji powiązanej	Dostępne wartości
typ	Telefon, Email, Sprawdź czy zdążysz na czas, Inny
status	Nie rozpoczęto, W trakcie realizacji, Skończone, Oczekuje na kogoś innego, Odroczone
priorytet	Niski, Średni, Widoki

Tabela 4.2 Pola z dostępnymi wartościami na obiekcie akcji powiązanej



Rysunek 4.14 Komponenty z dzisiejszymi wydarzeniami i akcjami powiązanymi użytkownika

Ostatnim niestandardowym komponentem na stronie głównej aplikacji jest lista automatycznie stworzonych raportów czasowych danego użytkownika posortowana od najnowszego elementu. Każdy z raportów czasowych zawiera informacje o powiązanym zadaniu, dacie i zaraportowanym czasie co pozwala przypominać o codziennym zaraportowaniu czasu na zadania, śledzić swój poświęcony czas w dane dni i wyciągać z nich wnioski – więcej o tym w segmencie o raportach indywidualnych (UC17).

Your time reports

NAME	JOB	DATE	TIME REPORTED (H)
TR# 30681	Test-Job-896	2019-08-17	5
TR# 30624	Test-Job-306	2019-08-17	5
TR# 30546	Test-Job-866	2019-08-16	7
TR# 30146	Test-Job-766	2019-08-16	6
TR# 30620	Test-Job-523	2019-08-15	6

Total records: 137 Page 1 of 28

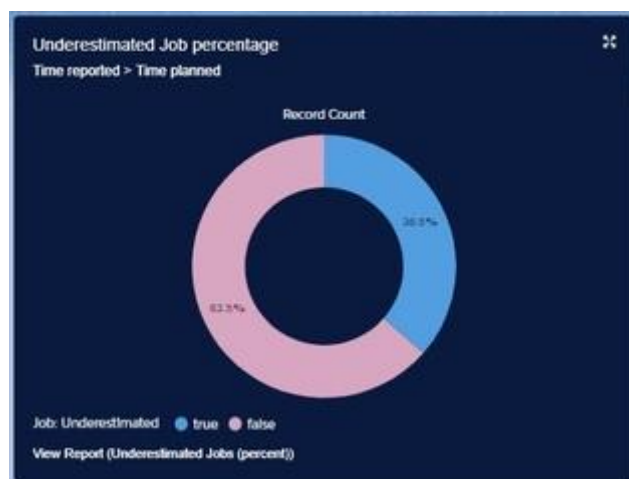
[< First](#)
[< Previous](#)
[Next >](#)
[Last >](#)

Rysunek 4.15 Komponent z raportami czasowymi pracy użytkownika

Raporty indywidualne

Każdy użytkownik ma dostępną standardową funkcjonalność polegającą na śledzeniu szeregu statystyk związanych ze swoją pracą. Mechanizm ten polega na stworzeniu odpowiedniego raportu z dostępnych obiektów, pól, okresu i innych filtrów, dzięki którym można przedstawić w formie różnorodnych wykresów czy wskaźników. Zrobione raporty można w estetyczny sposób umieścić na przeznaczonej do tego interaktywnej tablicy

(Dashboard). W ramach segmentu aplikacji dla każdego pracownika zostało zrealizowanych 6 spersonalizowanych raportów, dzięki którym każdy może otrzymać informację na temat swojej wydajności, tego jak dużo ma opóźnień i innych. Co więcej zostały stworzone raporty, dostępne dla wszystkich, opisujące podstawowe statystyki dotyczące klientów. W ramach rozdziału opiszę kolejno część z nich, aby nakreślić możliwości funkcjonalności. . Wszystkie raporty realizują wymaganie funkcjonalne UC12.



Rysunek 4.16 Statystyka o procencie niedoszacowanych zadań

Wykres kołowy przedstawiony na rysunku 4.16 opisuje procentową część zadań jakie zostały niedoszacowane przez zalogowanego użytkownika. Dzięki tego typu raportowi można ocenić jakość szacowań i przekazuje informację użytkownikowi czy powinien popracować nad tym aspektem swojej pracy. Istnieje podobny wykres opisujący procentowy udział zadań, w których użytkownik zakończył zadanie po spodziewanej dacie zakończenia. Dzięki temu możemy zaobserwować czy dany pracownik powinien mieć mniejszy czy też większy okres na realizację swoich zadań.

Inne przydatne sprawozdanie może dotyczyć liczby godzin przeznaczonych na zadania ponad estymacje. Wskaźnik ten powinien poinformować użytkownika czy w danym miesiącu przekroczył już pewien bufor bezpieczeństwa przeznaczony na niedoestymowanie zadań.



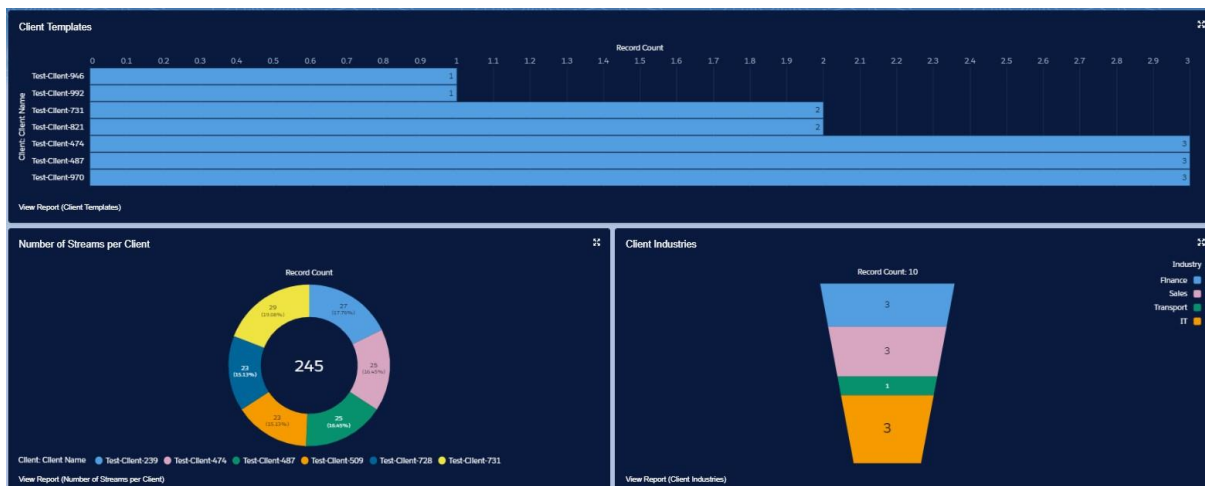
Rysunek 4.17 Wskaźnik niedoestymowanych godzin w danym miesiącu

Pomimo tego, że w ramach aplikacji przygotowałem jeszcze więcej raportów i powiązanych z nim wykresów mówiących o statystykach danego użytkownika, Ostatnim wykresem, który chciałbym opisać jest to suma zarobionych pieniędzy przez użytkownika na podstawie zaraportowanych godzin i zarobków opisanych w obiekcie Użytkownik, zakładając, że pracownik zarabia mając stawkę godzinową, a nie z góry założoną wypłatę.



Rysunek 4.18 Statystyka miesięcznych zarobków użytkownika

Bardzo ważnym elementem w funkcjonowaniu firmy chcącej wdrożyć elementy zwinne do swojego funkcjonowania jest wspólne podejmowanie decyzji operacyjnych przez wszystkich pracowników. Mając na uwadze, że firmy z danych branż wymieniają informacje między sobą i polecają sobie nawzajem dostawców czy podwykonawców. To jakie i w jaki sposób obsługujemy poszczególne branże, może świadczyć o rynku docelowym. Co więcej biorąc pod uwagę podejście do przejrzystości informacji o klientach w przedsiębiorstwach zwinnych przygotowany został zestaw raportów, dostępnych dla każdego pracownika, informujący o procencie udziałów klientów w świadczonych usługach, ich typach czy też liczbie szablonów jakie istnieją w systemie i są przypisane do danego klienta – co może świadczyć o stałej współpracy.



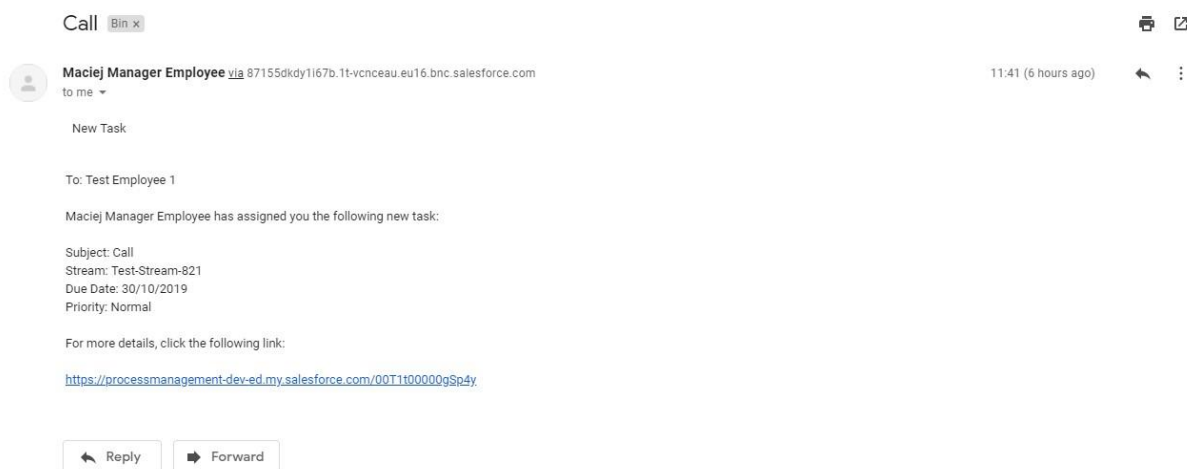
Rysunek 4.19 Tablica z wykresami dotyczącymi klientów

Powiadomienia email

W celu szybkiej i wygodnej kontroli nad swoimi zadaniami została zaimplementowana funkcjonalność wysyłki email do użytkownika. Pracownik z poziomu rekordu opisującego użytkownika ma możliwość wyłączyć powiadomienia zaznaczając odpowiednie pole wyboru. W aplikacji można wyróżnić trzy rodzaje powiadomień email:

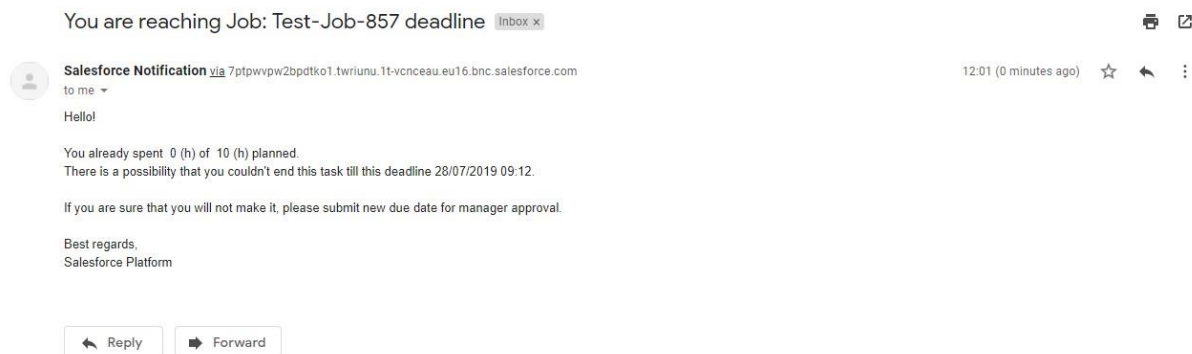
- powiadomienie o stworzeniu akcji powiązanej (standardowa funkcjonalność platformy)
- powiadomienie o możliwości niedokończenia zadania
- codzienne powiadomienie email o zaplanowanych zadaniach

Po stworzeniu akcji powiązanej użytkownik otrzymuje krótką wiadomość email stworzoną przez aplikację opisującą szczegóły utworzonego rekordu. Wpierają one wymaganie funkcjonalne UC18.



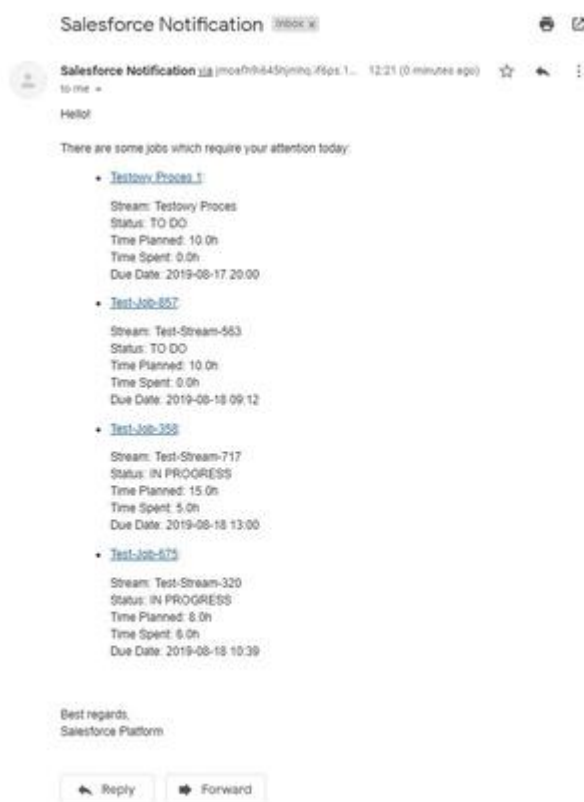
Rysunek 4.20 Przykładowa wiadomość email opisująca stworzenie akcji powiązanej

Bardzo przydatnym elementem jest powiadomienie o możliwości niedokończenia zadania. System co 15 minut sprawdza, czy istnieją zadania, które mają zaplanowaną datę zakończenia wcześniej niż teraźniejsza data z dodanym zaplanowanym czasem poświęcenia na pomniejszonym o czas poświęcony. Jest to wyraźny sygnał, że warto wysłać prośbę o przedłużenie ostatecznego terminu zakończenia zadania.



Rysunek 4.21 Wiadomość email opisująca listę zadań z prawdopodobnym przekroczeniem ostatecznej zaplanowanej daty zakończenia

Innym ważnym elementem pracy każdego pracownika jest planowanie dnia. z tego powodu został zaimplementowany codzienny mechanizm wysłania wiadomości email, która przedstawia listę zaplanowanych zadań na dziś.



Rysunek 4.22 Wiadomość email opisująca listę zadań zaplanowanych na dziś

4.4.3 Kontrola i analiza menadżerka

Pomimo tego, że idea zwinnej firmy zakłada przełożenie większej odpowiedzialności na pracowników i samodzielną organizację pracy oraz fakt, że każdy ma głos w sprawach dotyczących zarządzania przedsiębiorstwem, to należy założyć, że istnieje pewna hierarchia, gdzie kadra menadżerska odpowiada za elementy finansowe, płacowe czy zarządzania na wyższym poziomie. Biorąc pod uwagę te założenia przygotowane zostały funkcjonalności spełniające następujące cele:

- Monitorowanie kosztów związanych z zatrudnianiem pracowników
- Monitorowanie czasu poświęconego przez pracowników
- Monitorowanie przekroczeń ostatecznego terminu zakończenia i niedoestymowanych zadań
- Decydowanie o prośbach o przedłużenie ostatecznego terminu zakończenia

W ramach monitorowania przygotowane zostało 12 raportów dostępnych tylko menadżerowi, podzielone na 3 kategorie (wobec powyższej listy) oraz niestandardowy komponent oferujący wysłanie prośby o przedłużenie ostatecznego terminu zakończenia zadania. W rozdziale tym przedstawię kilka sporządzonych raportów i przebieg procesu akceptacji. Wszystkie raporty realizują wymaganie funkcjonalne UC12.

Raporty kosztów

Ta grupa raportów opisuje w różnorodny sposób statystyki kosztów grupując je branżami klientów czy też pracownikami i klientami w danym miesiącu. W tym rozdziale przedstawię kilka z nich:



Rysunek 4.23 Wykres kosztów miesięcznych z wyróżnieniem klientów

Dzięki powyższej statystyce menadżer jest w stanie stwierdzić w jakim miesiącu w przeciągu ostatnich 8 miesięcy przedsiębiorstwo miało najwyższe koszty i na którego klienta zostało poświęcone najwięcej pieniędzy na podstawie zaraportowanych godzin.



Rysunek 4.24 Miesięczne koszty poszczególnych pracowników

Wykres powyżej pozwala stwierdzić, który z pracowników w jakim miesiącu spędził najwięcej czasu wypełniając swoje obowiązki oraz trendy zapracowania każdego z nich. Menadżer jest w stanie zareagować jeśli zauważy, że jakaś z osób pracujących może wykorzystywać zbyt duży budżet lub poświęcać zbyt mało godzin pracy na swoje obowiązki.



Rysunek 4.25 Sumaryczne koszty zadań z podziałem na branże klientów

Dzięki skategoryzowaniu kosztów na branże klientów jesteśmy w stanie stwierdzić, która branża pochłania największe zasoby finansowe przedsiębiorstwa. Niestety w ramach tej pracy nie zostały przygotowane statystyki związane z przychodami skorelowanymi z realizacją procesów złożonych z postępujących po sobie zadań. Dopiero porównując dochody z kosztami byłibyśmy w stanie określić, która branża jest najbardziej opłacalna do współpracy.

Raporty czasu pracowników



Rysunek 4.26 Zestaw wykresów opisujących zaraportowany czas przez pracowników

Dzięki lewej kolumnie osoba zarządzająca jest w stanie zareagować, jeśli któraś z osób pracujących przekroczy pewne standardy związane z czasem poświęconym na zadania. Jeden z dwóch wykresów mówi o zaraportowanym czasie przez danych użytkowników w aktualnym miesiącu, co pozwala na zidentyfikować najbardziej zapracowane osoby i rekonfigurować przypisanie zadań. Drugi wykres pozwala obserwować aktualnie wykonywane zadania, dając szybki sygnał, że jakieś zadanie przekracza swój zaplanowany wcześniej czas. Za to cała prawa kolumna pozwala śledzić statystyki miesięczne, która z osób w danym miesiącu pracowała najwięcej.

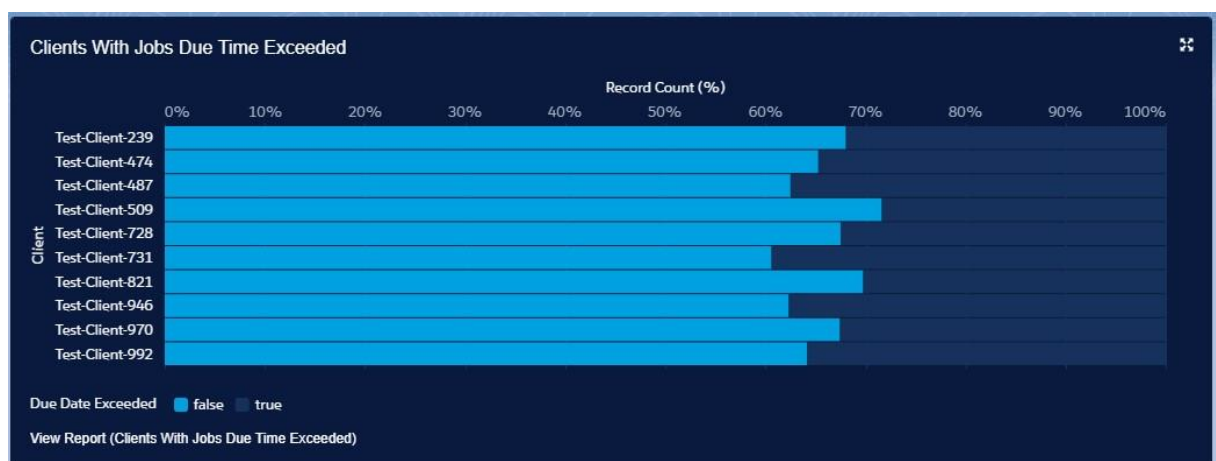
Raporty opóźnień

Ostatni zestaw raportów jaki przedstawię dotyczy się zagrożeń związanych z przekroczeniem zaplanowanego czasu na zadanie i przekroczeń ostatecznych dat zakończeń zadań. Sekcja ta pozwala na zidentyfikowanie niebezpieczeństw w danym miesiącu, sprawdzeniu który z klientów został ma największy procent opóźnionych zadań czy też podobnie do Rysunku 4.16 procent niedoszacowanych zadań w całej organizacji. Analogicznie do raportów kosztów przedstawię jedynie kilka z nich.

Dzięki grafowi na rysunku 4.28 menadżer jest w stanie zdecydować na którym kliencie należy się skupić, aby nie stracić jego zaufania. Takie szybkie reagowanie jest jak najbardziej elementem przedsiębiorstwa zwinnego i pozwala na długotrwałe utrzymanie współpracy. Rysunki 4.27 oraz 4.29 pozwalają śledzić całłościowe funkcjonowanie firmy i monitorowanie czy jej wydajność jest na odpowiednim poziomie.



Rysunek 4.27 Procent wszystkich niedoestymowanych zadań



Rysunek 4.28 Procent zadań niedostarczonych w terminie z podziałem na klienta



Rysunek 4.29 Sumaryczna liczba godzin ponad estymacje zadań

Proces akceptacji o przedłużenie ostatecznego terminu zakończenia zadania

W pewnym momencie pracownik organizacji może ocenić, że najprawdopodobniej nie zdąży wykonać przypisanego do siebie zadania w przed terminem ostatecznego terminu jego zakończenia. W tym celu zaprojektowana została funkcjonalność służąca do wysłania prośby do menadżera pracownika (zdefiniowanego na rekordzie użytkownika) o przełożenie tego terminu (UC16). Zwykły pracownik nie jest w stanie po prostu zmienić tego terminu z poziomu strony rekordu zadania. Aby ją zrealizować został zaimplementowany niestandardowy komponent z formularzem do wysłania wyżej wymienionego wniosku. Element ten został przedstawiony na rysunku poniżej.

The screenshot shows a job record for 'Test-Job-136' in an 'IN PROGRESS' state. A sidebar on the right contains a form titled 'Submit New Due Date For Manager Approval'. The form has two input fields: 'Date' with the value '31-Aug-2019' and 'Time' with the value '15:32'. Below these fields is a blue button labeled 'Submit Due Date'. The main content area shows job details: Job Name 'Test-Job-136', Stream 'Test-Stream-673', Status 'IN PROGRESS', Time Planned (h) '11.00', Created By 'Maciej Manager Employee, 17/08/2019 12:37', and Due Date '02/08/2019 05:41'.

Rysunek 4.30 Komponent do wysłania prośby o zmianę daty ostatecznego zakończenia zadania

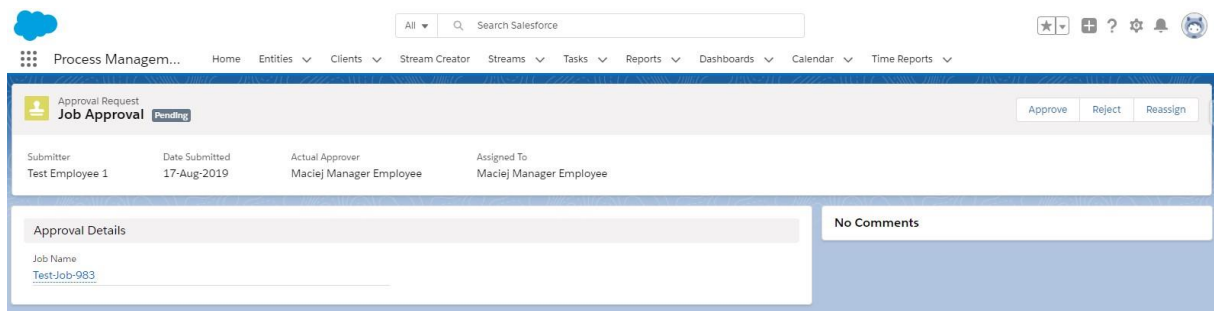
Po użyciu tego elementu pracownikowi wyświetli się informacja potwierdzająca uruchomienie procesu akceptacji. Następnie menadżer otrzyma powiadomienie o wysłanym procesie przez odpowiedniego pracownika.

The screenshot shows the Salesforce interface. A notification banner at the top right states: 'Test Employee 1 is requesting approval for job Job Name: Test-Job-983 3 minutes ago'. Below the notification are links for 'Email', 'Test-Client-968', 'Check if you can make it on time', 'Test-Job-911', 'Other', and 'Test-Client-622'. The main content area features a table titled 'Closest due date jobs you are assigned to' with columns: NAME, STREAM, STATUS, TIME PLANNED (H), TIME SPENT (H), and DUE DATE. The table contains three rows of job data. Below the table is a 'Your time reports' section with a table for tracking time spent on jobs.

NAME	STREAM	STATUS	TIME PLANNED (H)	TIME SPENT (H)	DUE DATE
Test-Job-468	Test-Stream-212	TO DO	12	0	Sep 03 2019 02:10
Test-Job-649	Test-Stream-652	TO DO	2	0	Sep 10 2019 02:18
Test-Job-938	Test-Stream-480	IN PROGRESS	4	7	Sep 13 2019 21:28

Rysunek 4.31 Powiadomienie o wysłaniu procesu akceptacji

Po kliknięciu lewym przyciskiem myszy w to powiadomienie, menadżer zostaje przekierowany do strony procesu. Z poziomu tej strony widzi wszystkie niezbędne informacje na temat zadania i zaproponowanej daty. Przy pomocy jednej z opcji jest w stanie zdecydować, czy zaakceptować prośbę, odrzucić ją czy przekierować do innej osoby, która powinna ją zaakceptować.



Rysunek 4.32 Strona procesu akceptacji

Przy wybraniu jednej z opcji menadżer zostanie poproszony o napisaniu komentarza motywującego swoją decyzję w wyskakującym oknie. Jeśli użytkownik zaakceptował proces, data wysłana w prośbie zostanie automatycznie podmieniona na instancji zadania i niezależnie od tego czy wniosek został zaakceptowany czy odrzucony, osoba inicjująca zostanie poinformowana o wyniku odpowiednim powiadomieniem.

4.5 Ograniczenia platformy

Przez to że platforma Salesforce.com działa w całości w chmurze, w celu utrzymywania stałej wydajności firma ta postanowiła wprowadzić pewne ograniczenia. W tym rozdziale przedstawię ograniczenia powiązane z projektowaniem rozwiązania przedstawionego w tej pracy dyplomowej. Ograniczenia związane ze sposobem implementacji pewnych elementów zostaną przedstawione w rozdziale 5.6.

Każda osoba chcąca skonstruować prototypową aplikację na platformie Salesforce.com lub przetestować pojedyncze rozwiązanie jest w stanie skorzystać z darmowego środowiska deweloperskiego. Właśnie na tego typu środowisku budowałem swoją aplikację w ramach implementacji przedstawionych wcześniej wymagań. Jednak aby nie przeciążyć serwerów firma zdecydowała się nałożyć szereg ograniczeń na tego typu środowiska, by móc swobodnie utrzymać inne środowiska na których działają opłacone instancje produkcyjne, testowe i deweloperskie ich klientów. Poniższa lista opisuje główne obostrzenia [14]:

- 2 aktywnych użytkowników (w moim przypadku jest to jeden z pracowników i menadżer)
- 5 MB do przechowywania danych (obiektów)
- 20 MB do przechowywania plików (załączników)
- 5000 wywołań API co 24 godziny
- 500 MB przepustowości i 10-minutowe limity czasu zgłoszeń serwisowych (na 24 godziny) dla aplikacji na stronie Force.com – są to strony, które można wystawić „na zewnątrz” platformy (na przykład do pozyskiwania klientów)
- 400 niestandardowych obiektów
- 10 niestandardowych aplikacji

Największą przeszkodą przy projektowaniu aplikacji była liczba aktywnych użytkowników i ograniczenie związane z rozmiarem przechowywanych danych. W celu sporządzenia statystyk byłem zmuszony stworzyć po kolei kilku użytkowników, a następnie odebrać im licencję platformy (użytkownik może istnieć bez licencji – jednak pozostaje nieaktywny). Na szczęście można przypisywać relację między obiektami wyłączonym użytkownikom, dzięki temu statystyki nie opierają się jedynie na 1 menadżerze i 1 pracowniku. Przez możliwość korzystania jedynie z 5MB miejsca, zaimplementowany generator danych należało zaprojektować w taki sposób, aby zasymulować w pseudolosowy sposób funkcjonowanie małego przedsiębiorstwa i zostawić pewien bufor bezpieczeństwa, aby dało się dalej funkcjonować z systemem. Gdyby limit został przekroczony – nie dałoby się wykonać większości akcji w aplikacji.

5 Implementacja

Jak już wcześniej zostało wspomniane aplikacje na platformie Salesforce.com zbudowane są ze standardowych i niestandardowych komponentów, obiektów, funkcjonalności i zależności, a samo programowanie aplikacji na serwerze różni się nieco od standardowej implementacji kompilowanej lokalnie na maszynie. Za każdym razem gdy zmienimy linie kodu, należy zamieścić ją na serwerze platformy, dopiero wtedy kod może być skompilowany, a następnie odpalony. W tym rozdziale mam zamiar przedstawić szczegóły dotyczące implementacji programu. Co oznacza programowanie komponentów i logiki między obiektami. Z czego się składają te komponenty oraz jakie narzędzia zostały użyte do ich implementacji. Ponad to przedstawię zaimplementowany algorytm walidacji acykliczności grafu skierowanego użytego w kreatorze i edytorze procesów oraz wzorzec delegacji w wyzwalaczach, dzięki któremu możliwe jest proste rozwijanie aplikacji i nie natrafienie przy tym na limity platformy, na których zakończę ten segment mojej pracy.

5.1 Technologie

Prace implementacyjne można podzielić na dwa główne aspekty. Komponenty nazwane przez platformę Lightning Web Components [15] realizowane przy pomocy wzorca Model-Widok-Kontroler (MVC) [12] i najnowszych standardów webowych oraz obsługi zachowań obiektów przy pomocy statycznych klas i metod pisane w języku APEX dające możliwość operowania na obiektach przy akcjach na wyzwalaczach.

5.1.1 Lightning Web Components (LWC) i biblioteka D3.js

Komponenty LWC korzystają z podstawowych standardów Internetowych i zapewnia tylko to, co jest niezbędne do uzyskania dobrej wydajności w przeglądarkach obsługiwanych przez Salesforce. Ponieważ jest oparty na kodzie, który działa natywnie w przeglądarkach, Lightning Web Components są lekkie i zapewniają wyjątkową wydajność. Większość pisanego kodu to standardowe skrypty JavaScript i HTML oraz stylowanie w CSS. Wykorzystuje niestandardowe elementy, szablony, dekoratory, moduły i inne nowe konstrukcje językowe dostępne w ECMAScript w wersji 7 i późniejszych [13].

Sam Salesforce jest zaangażowany w opracowywanie otwartych standardów internetowych i jest członkiem konsorcjum World Wide Web Consortium (W3C).

Wykorzystując wzorec MVC komponenty te dzielą się na 4 główne części przedstawione na poniższych listingach:

```
<template>
  <div class="buttonGroup">
    <div class="button">
      <lightning-button variant="destructive" name="delete" label="Delete All Data" onclick={deleteAllData} disabled={showSpinner}>
    </lightning-button>
    </div>
    <div class="button">
      <lightning-button variant="neutral" name="generateData" label="Generate Random Data" onclick={generateData} disabled=
{showSpinner}></lightning-button>
    </div>
  </div>
  <div if:true={showSpinner} class="spinnerClass">
    <lightning-spinner alternative-text="Loading" size="medium"></lightning-spinner>
  </div>
</template>
```

Listing 5.1 Widok komponentu

```
.buttonGroup {
  display: flex;
  justify-content: space-evenly;
  margin-bottom: 1%;
}

.spinnerClass{
  position: relative;
  margin: auto;
  width: 80px;
  height: 80px;
}
```

Listing 5.2 Stylowanie komponentu

```

import { LightningElement, api } from 'lwc';
import {
    ShowToastEvent
} from 'lightning/platformShowToastEvent';

import toastTitleSuccess from '@salesforce/label/c.TST_TITLE_Success';
import toastTitleError from '@salesforce/label/c.TST_TITLE_Error';

import generateRandomData from '@salesforce/apex/DataGenerator.generateRandomData';
import deleteAllData from '@salesforce/apex/DataGenerator.deleteAllData';

export default class DataGeneratorComponent extends LightningElement {

    @api showSpinner = false;

    deleteAllData() {
        this.showSpinner = true;
        deleteAllData({})
            .then(result => {
                if(result.isSuccess) {
                    this.fireToastEvent(toastTitleSuccess, result.msg, 'success');
                } else {
                    this.fireToastEvent(toastTitleError, result.msg, 'error');
                }
                this.showSpinner = false;
            })
            .catch(error => {
                this.fireToastEvent(toastTitleError, JSON.stringify(error), 'error');
                this.showSpinner = false;
            });
    }

    generateData() {
        this.showSpinner = true;
        generateRandomData({})
            .then(result => {
                if(result.isSuccess) {
                    this.fireToastEvent(toastTitleSuccess, result.msg, 'success');
                } else {
                    this.fireToastEvent(toastTitleError, result.msg, 'error');
                }
                this.showSpinner = false;
            })
            .catch(error => {
                this.fireToastEvent(toastTitleError, JSON.stringify(error), 'error');
                this.showSpinner = false;
            });
    }

    fireToastEvent(title, msg, variant) {
        this.dispatchEvent(
            new ShowToastEvent({
                title: title,
                message: msg,
                variant: variant
            })
        );
    }
}

```

Listing 5.3 Kontroler komponentu


```

@AuraEnabled
public static FrontResponseWrapper deleteAllData() {
    try {
        delete [SELECT Id FROM Task];
        delete [SELECT Id FROM Event];
        delete [SELECT Id FROM Account];
        delete [SELECT Id FROM Time_Report__c];
        delete [SELECT Id FROM Client__c];
        delete [SELECT Id FROM Job__c];
        delete [SELECT Id FROM Stream__c];
    } catch (Exception e) {
        return new FrontResponseWrapper(false, e.getMessage());
    }
    return new FrontResponseWrapper(true, Label.TST_MSG_DataDeleted);
}

```

Listing 5.4 Model komponentu

D3.js to biblioteka JavaScript do manipulowania dokumentami na podstawie wprowadzonych danych. Używa ona HTML, SVG i CSS. D3 daje pełne możliwości, łącząc potężne komponenty wizualizacji i oparte na danych podejście do manipulacji Document Object Model (DOM). D3.js pozwala powiązać dowolne dane z DOM [16], a następnie zastosować transformacje oparte na danych do dokumentu [17].

Jej głównym założeniem jest sprawna manipulacja dokumentami na podstawie danych. D3 jest niezwykle szybki, obsługuje duże zestawy danych i dynamiczne zachowania do interakcji i animacji. Jego funkcjonalny styl pozwala na ponowne użycie kodu poprzez różnorodną kolekcję oficjalnych i opracowanych przez społeczność modułów.

W moim projekcie biblioteka ta służy do wizualnej reprezentacji i manipulacji procesów, kreatora i edytora procesów.

5.1.2 Logika aplikacji w języku APEX

Apex to zorientowany obiektowo [18] język programowania, który umożliwia programistom wykonywanie instrukcji na serwerach Salesforce. Jego składnia wygląda jak język Java i działa podobnie do procedur przechowywanych w bazie danych. Język ten daje możliwość dodawanie logiki biznesowej do większości zdarzeń systemowych, takich jak:

- cykliczne procesy w tle (na przykład wysyłające powiadomienia email)
- kliknięcia przycisków i powiązane aktualizacje zapisów
- zachowania komponentów

Kod Apex może być inicjowany przez żądania usług internetowych i wyzwalaczy na obiektach. Zapewnia on również wbudowaną obsługę zapytań do zintegrowanej bazy danych platformy:

- wywołania języka manipulacji danymi (DML), takie jak INSERT, UPDATE i DELETE, które zawierają wbudowaną obsługę wyjątków manipulacji danymi (DmlException)

- bezpośrednie zapytania Salesforce Object Query Language (SOQL) do zintegrowanej bazy danych [19]
- bezpośrednie zapytania Salesforce Object Search Language (SOSL), które zwracają listy rekordów z różnych obiektów na podstawie podanych kryteriów [20]
- składnia pętli, umożliwiającą masowe przetwarzanie wielu rekordów jednocześnie
- składnia blokowania, zapobiegająca konfliktom aktualizacji rekordów, wywołuje niestandardowe publiczne wywołania API
- ostrzeżenia i błędy wydawane, gdy użytkownik próbuje edytować lub usunąć standardowy lub niestandardowy obiekt lub pole, do którego odwołuje się język

Apex opiera się na elementach Javy, takich jak zmienna i składnia wyrażenia, składnia bloku i instrukcji warunkowej, składnia pętli, obiekt i notacja tablicowa, ale wprowadza również dodatkowe elementy, dzięki którym można w prosty sposób odwoływać się do obiektów platformy i komponentów LWC.

5.2 Narzędzia implementacyjne

W tym rozdziale przedstawię narzędzia implementacyjne i migracyjne jakich użyłem do stworzenia programu. Wszystkie użyte narzędzia są aktualnym standardem w implementacji aplikacji na platformie Salesforce polecanym przez samą firmę.

5.2.1 Visual Studio Code

Do tworzenia i zarządzania zaimplementowanymi modułami wykorzystano Visual Studio Code. Jest to darmowy, nowoczesny edytor stworzony przez firmę Microsoft dla użytkowników systemów Windows, Linux i IOS. W porównaniu z konkurencją program wyróżnia się ciekawymi funkcjami, takimi jak dopracowany interfejs graficzny i szybkość działania.

Edytor został wyposażony w mechanizm kolorowania składni, wyświetlający wiele kart w jednym oknie lub wbudowany eksplorator plików, co znacznie usprawnia pracę nad bardziej złożonymi projektami. Dużym plusem jest również funkcja wyszukiwarki lub obsługa motywów graficznych.

5.2.2 Salesforce Extension Pack

Salesforce Extension Pack to rozszerzenie do Visual Studio Code, które jest wymagane do tworzenia oprogramowania na platformie Salesforce. Ten pakiet rozszerzeń zawiera narzędzia do programowania w edytorze VS Code. Zapewniają one funkcje do pracy ze środowiskami na którym implementuje się aplikacje, walidowanie komponentów LWC i języka Apex na poziomie składni [21].

5.2.3 Ant Migration Tool

Ant to biblioteka Java i narzędzie wiersza polecenia. Ant [22] był pierwotnie używany do tworzenia aplikacji Java [23], ale Salesforce.com dostarcza dodatkowe biblioteki, aby

umożliwić Antowi przenoszenie metadanych między katalogiem lokalnym a środowiskami na platformie Salesforce. Istnieje wiele metod migracji metadanych (na przykład przy użyciu zestawów zmian), ale narzędzie Ant Migration Tool jest szczególnie przydatne w następujących sytuacjach:

- projekty, w których trzeba wypełnić środowisko dużą ilością zmian w konfiguracji, a wprowadzanie tych zmian za pomocą interfejsu internetowego może zająć dużo czasu
- powtarzalne wdrożenia przy użyciu tych samych parametrów – można pobrać wszystkie metadane w swojej organizacji, wprowadzić zmiany i wdrożyć podzbiór składników
- ściągnięcie zmian wprowadzonych na platformie do lokalnych plików, a następnie do systemu kontroli wersji

5.3 System kontroli wersji

Do śledzenia zmian w plikach aplikacji wybrano system kontroli wersji Git z serwerem Github.com. Został on wykorzystany do zarządzania kodem źródłowym, co daje możliwość zachowania całej wersji aplikacji jako kopii zapasowej, strukturyzacji pracy i możliwości w przyszłości odpalenia programu na innych środowiskach bez klonowania projektu.

5.4 Walidacja acykliczności grafu skierowanego

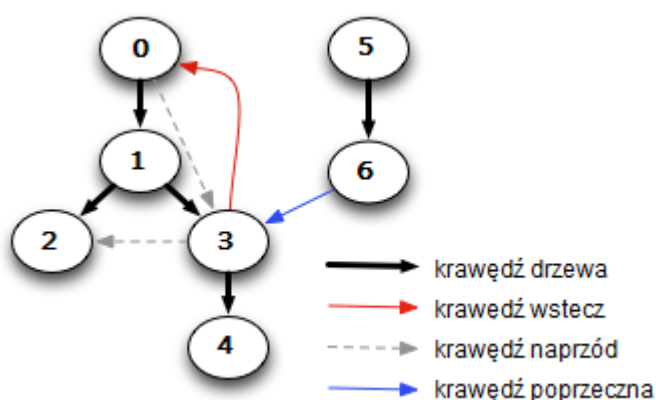
Wszystkie procesy w aplikacji reprezentowane są jako postępujące po sobie zadania w ustalonej hierarchii. Użytkownik przy pomocy kreatora tworzy graf skierowany, w którym zadanie reprezentowane jest jako wierzchołek, a kolejność zadań jako krawędzie tego grafu. Zachowując logiczną kolejność wykonywania w takim przebiegu, zaimplementowana została metoda walidacji acykliczności takiego grafu. Skierowany graf acykliczny to taki, w którym nie istnieje ścieżka prowadząca do wierzchołka, który był początkiem tej ścieżki. [24]

Gdyby istniała możliwość zdefiniowania pętli – kolejność wykonywania straciłaby sens. W wypadku rozwoju aplikacji w kierunku stworzenia bloków warunkowych walidacja taka powinna zostać dopracowana. W tego typu blokach istniała by możliwość powtórzenia pewnych czynności, jeśli pewne warunki nie zostały spełnione. Jednakże obecnie takie założenia nie zostały popelnione, więc badanie acykliczności grafu jest niezbędne do prawidłowego funkcjonowania.

Realizacja walidacji acykliczności grafu skierowanego została zrealizowana przy pomocy przeszukiwania w głąb. Jest to algorytm polegający na sprawdzaniu każdego wierzchołka połączonego krawędzią z początkowym węzłem, tak że sprawdzany wierzchołek staje się tym, z którego sprawdzane są połączenia. Po zbadaniu wszystkich krawędzi wychodzących do węzła, algorytm powraca do wierzchołka z którego dany węzeł został odwiedzony [25].

W algorytmie przeszukiwania w głąb (ang. DFS) można wyróżnić następujące rodzaje krawędzi [26]:

- krawędź naprzód – wskazująca od wierzchołka do jednego z jego niebezpośrednich potomków (potomków już odwiedzonych)
- krawędź drzewa – wskazująca na bezpośredniego potomka przy zwykłym przeszukiwaniu danego drzewa
- krawędź poprzeczna – wskazująca na poprzednio odwiedzany wierzchołek, który nie jest ani przodkiem, ani potomkiem
- krawędź wstecz – wskazująca wierzchołek, który jest jednym z przodków przy przeszukiwaniu w głąb



Rysunek 5.1 Rodzaje krawędzi przy przeszukiwaniu w głąb. [26]

Aby zrealizować walidację acykliczności grafu skierowanego należy sprawdzić czy nie istnieją żadne krawędzie wstecz powodujące cykle. W tym celu trzeba jeszcze określić jedno ważne pojęcie. Jest to czas odejścia (ang. departure time) danego wierzchołka w grafie przy przeszukiwaniu DFS [27]. Czas odejścia jest to (liczba przejść), który poświęciliśmy na zbadanie rodzeństwa wierzchołka i jesteśmy gotowi na powrót do rodzica.

Algorytm walidacyjny polega na wykonywaniu przeszukiwania w głąb po kolei na każdym nieodkrytym jeszcze wierzchołku. Jest to niezbędne, ponieważ musimy założyć że graf może być niespójny. Przy trafieniu na dany wierzchołek oznaczamy go jako już odkryty i wykonujemy przeszukiwanie w głąb na każdym z jego dzieci. Gdy natrafimy na pierwszy wierzchołek, który nie ma już dzieci oznaczamy jego czas odejścia jako 0 i przechodzimy do jego rodzeństwa oznaczając je kolejnymi liczbami. Oznaczanie czasu odejścia odbywa się w sposób rekurencyjny, więc będzie ono rosnąć również w stronę starszych generacji węzłów. Biorąc pod uwagę że przeszukujemy w głąb, na tej podstawie jesteśmy w stanie stwierdzić, że starsze generacje węzłów zawsze będą miały wyższy czas odejścia niż ich potomkowie. Na podstawie definicji krawędzi wstecz, która mówi o tym, że jest to krawędź od potomka do przodka przy przeszukiwaniu DFS, jesteśmy w stanie stwierdzić, że będzie to krawędź w której węzeł od którego ona idzie ma mniejszy czas odejścia, niż węzeł docelowy (przodek) [28].

Aby zrealizować ten algorytm, należy przeprosować opis grafu w formacie JSON, na obiekty odpowiadające za sam graf oraz krawędzie. Przeprowadzić przeszukiwanie w głąb z oznaczeniem odwiedzonych wierzchołków i czasem odejścia, a na końcu sprawdzić czy istnieją jakieś krawędzie wsteczne w tym grafie na podstawie zapisanych czasów odejścia oraz hierarchii zadań. Poniższy kod przedstawia część przykładowej implementacji algorytmu.

```
1. private static Boolean isDAG(Graph currentGraph, Integer numberOfNodes) {
2.     Map<Integer, Boolean> discovered = new Map<Integer, Boolean>();
3.     Map<Integer, Integer> departure = new Map<Integer, Integer>();
4.     for (Integer i = 0; i < numberOfNodes; i++) {
5.         discovered.put(i, false);
6.         departure.put(i, 0);
7.     }
8.     Integer timeConsumed = 0;
9.     for (Integer i = 0; i < numberOfNodes; i++) {
10.        if (!discovered.get(i)) {
11.            timeConsumed = DFS(currentGraph, i, discovered, departure, timeConsumed);
12.        }
13.    }
14.    for (Integer u = 0; u < numberOfNodes; u++) {
15.        for (Integer v : currentGraph.adjacencyList.get(u)) {
16.            if (departure.get(u) <= departure.get(v)) {
17.                return false;
18.            }
19.        }
20.    }
21.    return true;
22. }
```

Listing 5.5 Implementacja algorytmu walidującego acykliczność grafu skierowanego

```
1. private static Integer DFS(Graph graph, Integer v, Map<Integer, Boolean> discovered, Map<Integer, Integer> departure, Integer timeConsumed) {
2.     discovered.put(v, true);
3.     for (Integer u : graph.adjacencyList.get(v)) {
4.         if (!discovered.get(u)) {
5.             timeConsumed = DFS(graph, u, discovered, departure, timeConsumed);
6.         }
7.     }
8.     departure.put(v, timeConsumed++);
9.     return timeConsumed;
10. }
```

Listing 5.6 Implementacja przeszukiwania w głąb z wyznaczeniem czasu odejścia

5.5 Wzorzec delegacji w wyzwalaczach

Wzorzec delegacji, na wysokim poziomie abstrakcji, zakłada, że wykonanie przez obiekt pewnej operacji może zostać przekazane do realizacji innemu obiektowi. Delegowanie pozwala na dynamiczne modyfikowanie powiązań pomiędzy obiektami oraz modyfikowanie zachowań obiektów. Stanowi również (wraz z abstrakcją i wstrzykiwaniem zależności) ważny środek do budowania wzorców projektowych oraz budowania kodu podatnego na rozszerzenia a odpornego na modyfikacje.

Kod w języku Apex można wywoływać za pomocą wyzwalaczy (ang. triggers). Wyzwalacze Apex [29] umożliwiają wykonywanie niestandardowych działań przed lub po zmianach w rekordach Salesforce, takich jak wstawianie, aktualizacje lub usuwanie.

Wyzwalaczem jest kod Apex, który wykonuje się przed lub po następujących rodzajach

- insert
- update
- delete
- merge
- upsert
- undelete

Na przykład można uruchomić wyzwalacz przed wstawieniem rekordów obiektu do bazy danych, po ich usunięciu lub nawet po przywróceniu rekordu z Kosza.

Można zdefiniować wyzwalacze dla standardowych obiektów najwyższego poziomu, które obsługują wyzwalacze, takie jak Kontakt lub Konto i każdego niestandardowego obiektu.

Istnieje jeszcze jeden podział wyzwalaczy na:

- before
- after

Tak jak wskazują ich nazwy można wykonywać działania przed i po zadaniach opisanych w poprzedniej liście.

Wyzwalacze After służą do uzyskiwania dostępu do wartości pól ustawianych przez system (takich jak identyfikator rekordu lub pole LastModifiedDate) oraz do wpływania na zmiany w innych rekordach, takich jak logowanie do tabeli kontroli lub uruchamianie asynchronicznych zdarzeń w kolejce [29].

Biorąc pod uwagę do czego służą wyzwalacze i to, że wykonuje się na nich takie same typy operacji na każdym obiekcie, aby nie natrafiać na limity opisane w rozdziale 5.6 warto zastosować wzorzec delegacji. Dzięki temu mamy spójną strukturę dla każdej obsługi wyzwalacza. Struktura ta polega na początkowym tworzeniu odpowiednich kolekcji przy danym rodzaju wywołanego wyzwalacza, sprawdzenie masowo warunków logiki aplikacji, jeśli tak to obiekty są zbierane do wcześniej wspomnianych kolekcji. Po zebraniu wszystkich kolekcji następuje masowe wykonanie operacji na rekordach. Dzięki zbiorowemu traktowaniu obiektów transakcja wykonywana jest tylko raz na funkcjonalność. Takie podejście pozwala budować kolejne funkcjonalności w oparciu o wzorzec delegacji nie przejmując się natrafieniem na limity.

Na rysunku poniżej mamy przykład wyzwalacza (wyzwalacz JobTrigger), który deleguje prace do wykonawcy (klasa TriggerHandler). Wykonawca posiada w sobie implementację wzorca delegacji. Wzorzec ten zachowuje się w sposób jaki zostało opisane powyżej. Wyznacza tylko kolejność operacji, a same operacje wykonywane są w klasie podwykonawcy danego wyzwalacza (klasa TH_JobTrigger).

```

trigger JobTrigger on Job__c (
    before insert,
    before update,
    before delete,
    after insert,
    after update,
    after delete,
    after undelete)
{
    TriggerHandler.execute(new TH_JobTrigger());
}

```

Listing 5.7 Wyzwalacz delegujący zadanie

```

public with sharing class TriggerHandler{

    public interface Delegate {
        void prepareBefore();
        void prepareAfter();

        void beforeInsert(List<sObject> o);
        void beforeUpdate(Map<Id, sObject> old, Map<Id, sObject> o);
        void beforeDelete(Map<Id, sObject> o);

        void afterInsert(Map<Id, sObject> o);
        void afterUpdate(Map<Id, sObject> old, Map<Id, sObject> o);
        void afterDelete(Map<Id, sObject> o);
        void afterUndelete(Map<Id, sObject> o);

        void finish();
    }

    public abstract class DelegateBase implements Delegate {

        public virtual void prepareBefore() {}
        public virtual void prepareAfter() {}

        public virtual void beforeInsert(List<sObject> o) {}
        public virtual void beforeUpdate(Map<Id, sObject> old, Map<Id, sObject> o) {}
        public virtual void beforeDelete(Map<Id, sObject> o) {}

        public virtual void afterInsert(Map<Id, sObject> o) {}
        public virtual void afterUpdate(Map<Id, sObject> old, Map<Id, sObject> o) {}
        public virtual void afterDelete(Map<Id, sObject> o) {}
        public virtual void afterUndelete(Map<Id, sObject> o) {}

        public virtual void finish() {}
    }
}

```

Listing 5.8 Implementacja wzorca delegacji wyznaczającego kolejność działań 1/2

2

```
public static void execute(Delegate d) {
    if (Trigger.isBefore) {
        d.prepareBefore();
        if (Trigger.isInsert) {
            d.beforeInsert(Trigger.new);
        } else if (Trigger.isUpdate) {
            d.beforeUpdate(Trigger.oldMap, Trigger.newMap);
        } else if (Trigger.isDelete) {
            d.beforeDelete(Trigger.oldMap);
        }
    } else {
        d.prepareAfter();
        if (Trigger.isInsert) {
            d.afterInsert(Trigger.newMap);
        } else if (Trigger.isUpdate) {
            d.afterUpdate(Trigger.oldMap, Trigger.newMap);
        } else if (Trigger.isDelete) {
            d.afterDelete(Trigger.oldMap);
        } else if (Trigger.isUndelete) {
            d.afterUndelete(Trigger.newMap);
        }
    }
    d.finish();
}
```

Listing 5.9 Implementacja wzorca delegacji wyznaczającego kolejność działań 2/2

W klasie implementującej zachowanie logiki aplikacji na jednej z akcji opisanej już w tym rozdziale możemy zauważyć nadpisaną implementację metod, które są we wzorcu delegacji. Dzięki dziedziczeniu po wewnętrznej klasie naszego wzorca (klasa `TriggerHandler.DelegateBase`), kolejność wykonywania transakcji jest ściśle zdefiniowana. Na początku wykonywane są metody z przedrostkiem „prepare”, w których jedynie definiujemy kolekcje - dobra praktyka nakazuje definiowanie tylko tych kolekcji, których mamy zamiar używać, nie zużywamy dzięki temu dostępnej pamięci aplikacji.

Następnie, niezależnie od typu akcji, wykonywana jest metoda z przedrostkiem „before”, która odpowiada za zbieranie rekordów do zmian, które mają być wykonywane przed samą operacją, która ma zostać wykonana.

Dobrym przykładem jest zadanie „update”, w którym często chcemy zmienić wartość innych pól, oprócz tego co zostało zmienione, aby nie wykonywać operacji aktualizacji dwa razy. Inną funkcjonalnością może być walidacja wartości przed aktualizacją – jeśli wartość jest niepoprawna, jesteśmy w stanie przerwać aktualizację rekordu i ukazać odpowiedni błąd rzucając wyjątek. Po zebraniu rekordów wykonywane są same akcje do których te rekordy zbieraliśmy, za to (według naszej kolejności we wzorcu w metodzie

„execute”) odpowiada metoda „finish”. Jeśli nie zostaną rzucone żadne wyjątki, według przykładu który opisuje, po wyjściu z tej metody rekord zostanie zaktualizowany.

```
1: public without sharing class TH_JobTrigger extends TriggerHandler.DelegateBase {
2:     Map<Job__c, Decimal> jobsWithTimeReported;
3:     Map<Id, List<Job__c>> jobsToCheckTemplate;
4:     Map<Id, Job__c> jobsToUpdateStreamJSON;
5:     public override void prepareBefore() {
6:         jobsToCheckTemplate = new Map<Id, List<Job__c>>();
7:     }
8:     public override void prepareAfter() {
9:         jobsWithTimeReported = new Map<Job__c, Decimal>();
10:        jobsToUpdateStreamJSON = new Map<Id, Job__c>();
11:    }
12:    public override void beforeInsert(List<SObject> newObjectsMap) {
13:        List<Job__c> newJobsList = (List<Job__c>)newObjectsMap;
14:        collectJobsToCheckTemplate(newJobsList);
15:    }
16:    public override void afterInsert(Map<Id, SObject> newObjectsMap) {
17:        Map<Id, Job__c> newJobsMap = (Map<Id, Job__c>)newObjectsMap;
18:        for (Id key : newJobsMap.keySet()) {
19:            Job__c newJob = newJobsMap.get(key);
20:        }
21:    }
22:    public override void afterUpdate(Map<Id, SObject> oldObjectsMap, Map<Id, SObject> newObjectsMap) {
23:        Map<Id, Job__c> newJobsMap = (Map<Id, Job__c>)newObjectsMap;
24:        Map<Id, Job__c> oldJobsMap = (Map<Id, Job__c>)oldObjectsMap;
25:        for (Id key : newJobsMap.keySet()) {
26:            Job__c newJob = newJobsMap.get(key);
27:            Job__c oldJob = oldJobsMap.get(key);
28:            checkReportedTimeChanges(oldJob, newJob);
29:            checkForStreamJSONUpdate(oldJob, newJob);
30:        }
31:    }
32:    public override void beforeDelete(Map<Id, SObject> oldObjectsMap) {
33:        JobManager.updateRelatedJobs(oldObjectsMap.keySet());
34:    }
35:    private void collectJobsToCheckTemplate(List<Job__c> jobsToCollectByStreamId) {
36:        for (Job__c jobToCheck : jobsToCollectByStreamId) {
37:            if (jobToCheck.Stream__c != null && jobToCheck.RecordTypeId != CommonUtility.getRecordTypeId(CommonUtility.SBJECT_API_NAME_JOB, CommonUtility.JOB_TYPE_STANDARD_TEMPLATE)) {
38:                if (jobsToCheckTemplate.containsKey(jobToCheck.Stream__c)) {
39:                    jobsToCheckTemplate.get(jobToCheck.Stream__c).add(jobToCheck);
40:                } else {
41:                    jobsToCheckTemplate.put(jobToCheck.Stream__c, new List<Job__c>{jobToCheck});
42:                }
43:            }
44:        }
45:    }
46:    private void checkReportedTimeChanges(Job__c oldJob, Job__c newJob) {
47:        if (
48:            newJob.Assigned_To__c != null
49:            && (oldJob != null && oldJob.Time_Spent__c != newJob.Time_Spent__c) //afterUpdate
50:        ) {
51:            jobsWithTimeReported.put(newJob, oldJob != null ? newJob.Time_Spent__c - oldJob.Time_Spent__c : newJob.Time_Spent__c);
52:        }
53:    }
54:    private void checkForStreamJSONUpdate(Job__c oldJob, Job__c newJob) {
55:        if (
56:            newJob.Name != null
57:            && ((oldJob.Name != newJob.Name)
58:                || oldJob.Status__c != newJob.Status__c)
59:        ) {
60:            jobsToUpdateStreamJSON.put(newJob.Id, newJob);
61:        }
62:    }
63:    public override void finish() {
64:        if (jobsToCheckTemplate != null && !jobsToCheckTemplate.isEmpty()) {
65:            JobManager.checkJobToTemplateIfTemplateStream(jobsToCheckTemplate);
66:        }
67:        if (jobsWithTimeReported != null && !jobsWithTimeReported.isEmpty()) {
68:            TimeReportManager.createTimeReports(jobsWithTimeReported);
69:        }
70:        if (jobsToUpdateStreamJSON != null && !jobsToUpdateStreamJSON.isEmpty()) {
71:            JobManager.updateRelatedStreamJSON(jobsToUpdateStreamJSON);
72:        }
73:    }
74: }
75: }
```

Listing 5.10 Przykładowa implementacja obsługi zadań wyzwalacza

Następnie wszystkie operacje zostaną powtórzone, tylko dla akcji wyznaczonych po aktualizacji rekordu z przedrostkiem „after”.

Przykład opisany powyżej jest analogiczny dla tworzenia, usuwania czy też cofnięcia usuwania z tą różnicą, że w niektórych przypadkach nie mamy dostępu do niektórych wersji rekordów. Dostęp do wartości przed i po akcji jest dostępny tylko przy aktualizacji, co jest naturalne, ponieważ przy tworzeniu rekordów nie jesteśmy w stanie wiedzieć jakie wartości miał przed stworzeniem, gdyż jeszcze nie istniał. Analogicznie jest z opcją usuwania, tylko z wartościami po usunięciu.

5.6 Ograniczenia implementacyjne platformy

Salesforce.com jest to największa platforma CRM działająca w chmurze. Oznacza to tyle, że cały kod aplikacji, dane i wszystkie wykonania operacji wykonywane są na serwerach. W celu dbania o szybkość serwerów platforma Salesforce.com odgórnie narzuca pewne ograniczenia związane z częstotliwością wywołań do zintegrowanej bazy danych, liczby asynchronicznych wywołań kodu i innych. Poniżej zostały opisane najważniejsze z nich, przez które programiści zmuszeni są stosować pewne dobre praktyki – na przykład takie jak wzorzec delegacji w wyzwalaczach opisany w rozdziale 5.5. Limity dotyczą wywołań synchronicznych i asynchronicznych w pojedynczej transakcji (czyli przebiegu kodu wywołanej akcją użytkownika, mechanizmem w aplikacji na platformie lub zewnętrznego systemu), chyba że zostało to opisane inaczej [30].

- maksymalna liczba wywołań SOQL do bazy – 100 synchronicznych i 200 asynchronicznych
- maksymalna liczba wywołań SOSL do bazy – 20
- maksymalna liczba pobranych rekordów przy pomocy – 50,000
- maksymalna liczba rekordów, na których można jednorazowo wykonać operacje przetwarzania danych w ramach zintegrowanej bazy danych (np. insert, update, delete) – 10,000
- maksymalna głębokość stosu wywołań wyzwalaczy – 16
- maksymalny rozmiar stosu – 6MB dla synchronicznych i 12MB dla asynchronicznych
- maksymalny czas użytku procesora serwera w ramach transakcji – 10,000 milisekund dla synchronicznych i 60,000 milisekund dla asynchronicznych
- maksymalny czas przebiegu transakcji – 10 minut
- maksymalna liczba znaków w klasie – 1 milion

6 Testowanie

Etap ten, jest nieodłącznym elementem implementowania każdej funkcjonalności w każdym systemie. Nie tylko dlatego, że jest to dobrą praktyką, ale przede wszystkim dlatego, że bez dokładnych testów nie jesteśmy tak naprawdę w stanie stwierdzić czy nasz program działa w pożądanym sposób. Testowanie nie jest w stanie wykryć wszystkich defektów oprogramowania, jednak może dostarczyć informacji o jego zgodności z wymaganiami klienta, czy też z jego oczekiwaniami. Trzeba pamiętać, że testowanie nie sprawdza oprogramowania pod kątem wszelkich możliwych warunków początkowych, lecz jedynie w wyselekcjonowanych warunkach.

Testowanie można przedstawiać na kilku wyselekcjonowanych poziomach i typach [31]:

Testy jednostkowe – służą do testowania pojedynczych modułów lub komponentów oprogramowania. Dzięki nim sprawdza się najmniejsze części kodu (np. pojedyncze funkcje, klasy, moduły). Ich głównym założeniem jest wykonanie każdego testu w izolowanych warunkach, co przekłada się na przygotowanie danych pod każdy test. Ważne na tym poziomie jest testowanie wartości brzegowych, czyli. W tym rodzaju testowania powinniśmy sprawdzić jak najwięcej różnych ścieżek przejścia przez dany moduł.

Testy integracyjne – ich celem jest sprawdzenie poprawności integracji pomiędzy danymi komponentami. Dobrym przykładem jest sprawdzanie poprawności łączenia się z bazą danych. Na tym poziomie testowane mogą być mniejsze komponenty takie jak funkcje, klasy, a także całe programy lub systemy. Wyróżnić można tu zstępującą i wstępującą metodę testowania. Pierwsza z nich polega na testowaniu w pierwszej kolejności elementów najwyżej w hierarchii, a następnie niższych rzędów, kończąc na najniższym. Metoda wstępująca działa odwrotnie. Zaczynamy od modułów najniższych rzędów (ułatwia to testowanie kolejnych modułów na wyższych rzędach), a kończąc na najwyższym.

Testy akceptacyjne – ostateczne testy wykonywane są w celu akceptacji produktu przez klienta, lub inną uprawnioną do tego osobę. W przypadku tych testów najczęściej ustala się formę raportowania błędów przez osobę testującą, aby jak najszybciej programiści mogli je naprawić. Jest to kluczowy etap w końcowej fazie projektu. Testy przeprowadzane są na środowisku docelowym lub środowisku odtwarzającym docelowe w pełni i powinny sprawdzać, czy oprogramowanie spełnia oczekiwania i wszystkie założenia.

Testy manualne - są wykonywane osobiście przez testerów przechodzących przez kolejne elementy aplikacji lub korzystających z odpowiednich narzędzi w celu interakcji z oprogramowaniem i interfejsami. Wymagają one skonstruowania scenariuszy testowych opisujących listę kroków jakie należy wykonać wraz z oczekiwanymi rezultatami. Takie testowanie wymaga zatrudnienia testerów oraz skonfigurowania dodatkowego środowiska. Ich głównym minusem jest to, że mogą być także podatne na błędy ludzkie – na przykład pominięcie, któregoś z kroków lub niepoprawne przygotowanie danych.

Testy automatyczne – jest typ testów wykonywanych przez maszynę. Istnieją narzędzia pozwalające definiować pewne ścieżki użytkownika złożone z pojedynczych kroków, dzięki którym symulują przejście testera przez system w sposób automatyczny – sprawdzając jednocześnie oczekiwane rezultaty. Tego typu testów często bazują na wykrywaniu odpowiednich elementów stron (w przypadku aplikacji internetowych) i wypełnianiu, modyfikowaniu czy po prostu wybieraniu ich. Pozwalają one na częste testy, bez zużywania czasów pracowników – są szczególnie przydatne przy tak zwanych testach regresji, służących do sprawdzenia czy przy dalszym rozwijaniu funkcjonalności nie zmodyfikowały się elementy już istniejące w systemie.

W ramach pracy przygotowałem bardzo dokładne testy jednostkowe oraz najważniejsze scenariusze testowe przedstawione w następnych rozdziałach.

6.1 Testy jednostkowe

W swoim projekcie przeprowadziłem dokładne testy jednostkowe. W ramach Salesforce.com istnieje zintegrowane narzędzie do tworzenia tego typu testów, budując klasy i oznaczając je odpowiednimi tagami. Warto również nadmienić, że sama platforma, aby móc przenieść lub zainstalować aplikację na innym środowisku wymaga 75-cio procentowego pokrycia kodu. Oznacza to tyle, że testy jednostkowe przechodząc przez daną linię kodu oznaczają ją jako pokrytą, a więc procent pokrycia jest to stosunek pokrytych linii kodu do wszystkich istniejących w aplikacji. Na poniższym rysunku przedstawiłem raport z wykonanych testów jednostkowych wraz z procentowym i liczbowym pokryciem klas:

Status	Test Run	Enqueued Time	Duration	Failures	Total	Overall Code Coverage		
✓	7071t00001hrXNg	Thu Aug 22 2019 09:15:52 GM...		0	2	Class	Percent	Lines
✓	TimeReportManagerTest			0	4	Overall	98%	
✓	timeReportsModified		0:01			CommonUtility	100%	21/21
✓	timeReportsDeletion		0:01			DAGChecker	100%	61/61
✓	timeReportsCreation		0:00			DailyReminderBatch	100%	56/56
✓	timeReportsCounter		0:01			DataGenerator	96%	124/129
✓	ResetReminderSentBatchTest			0	1	DueDatesComingBatch	100%	60/60
✓	testBatchExecution		0:00			FrontResponseWrapper	100%	10/10
✓	ProcessCreatorControllerTest			0	9	JobManager	100%	93/93
✓	updateStreamJSONDescriptionSuccess		0:00			JobTrigger	100%	1/1
✓	updateStreamJSONDescriptionError		0:00			ManageApprovalProcess	100%	15/15
✓	saveStreamAsTemplateSuccess		0:00			ProcessCreatorController	100%	34/34
✓	saveStreamAsTemplateError		0:00			RandomUtility	100%	23/23
✓	retrieveJSONStreamDescription		0:00			ResetReminderSentBatch	100%	18/18
✓	deleteSelectedJobSuccess		0:00			TestHelper	100%	159/159
✓	deleteSelectedJobError		0:01			TH_JobTrigger	100%	46/46
✓	createConnectionBetweenJobsSuccess		0:00			TimeReportManager	100%	45/45
✓	createConnectionBetweenJobsError		0:00			TriggerHandler	65%	19/29
✓	ManageApprovalProcessTest			0	3			
✓	submitDueDateApprovalSuccess		0:00					
✓	submitDueDateApprovalManagerNotDefined		0:01					
✓	submitDueDateApprovalLocked		0:00					
✓	JobManagerTest			0	6			
✓	userJobsCount		0:01					
✓	updateRelatedJSONStream		0:03					
✓	updateLookupAfterRelatedJobDeletion		0:01					
✓	cloneStreamWithJobsSuccess		0:01					
✓	cloneStreamWithJobsError		0:00					
✓	changeJobToTemplateIfStreamIsTemplate		0:05					
✓	DueDatesComingBatchTest			0	1			
✓	testBatchExecution		0:02					
✓	DataGeneratorTest			0	2			
✓	DailyReminderBatchTest			0	1			
✓	testBatchExecution		0:01					
✓	DAGCheckerTest			0	2			
✓	notDAGGraph		0:00					
✓	DAGGraph		0:00					
✓	CommonUtilityTest			0	1			
✓	getRecordTypeIdErrorHandling		0:00					

Rysunek 6.1 Raport z testów jednostkowych i pokrycia klas

Jedynie nieznaczne braki w pokryciu istnieją jedynie w dwóch klasach (DataGenerator oraz TriggerHandler). Pierwsza z nich to generator danych. Jej niecałociowe pokrycie wynika z linii kodu obsługujące specjalne wyjątki bazodanowe, które są niemożliwe do odtworzenia w testach. Drugi z nich opisuje obsługę wzorca delegacji w wyzwalaczu. Został on zaimplementowany tak, aby był łatwo rozszerzalny przy nowych funkcjonalnościach, więc elementy które nie zostały wykorzystane przy aktualnych funkcjonalnościach nie są pokryte – tylko te były testowane w testach jednostkowych.

6.2 Scenariusze testowe

W celu poprawnego wykonania testów manualnych należy się posłużyć tak zwanymi scenariuszami testowymi. Scenariusze (oznaczone skrótem ST i liczbą porządkową) budowane są z pojedynczych przypadków testowych (oznaczoną skrótem P, liczbą porządkową scenariusza i własną liczbą porządkową oddzieloną kropką). Każdy z nich powinien składać się z kilku istotnych elementów: oznaczenia przeprowadzonego testu, celu testu, warunków początkowych, akcji jakie ma wykonać użytkownik, powiązanych z nim oczekiwanych rezultatów oraz oceny każdego kroku. Dzięki temu, osoba jest w stanie ocenić czy dana funkcjonalność działa w poprawny sposób. Głównym atutem tworzenia tego typu testów jest to, że może go wykonać każda osoba mająca dostęp do scenariusza. Poprzez jasną strukturę i opis nie wymagane jest doświadczenie testerskie, przez co akcję tą może wykonać każdy członek zespołu. Co więcej mając przypadki opisane w strukturyzowany sposób, same scenariusze mogą służyć jako ważny element dokumentacji. Jednak trzeba zaznaczyć, że istotną wadą jest trudne utrzymywanie, jeśli aplikacja jest ciągle rozwijana – w takim przypadku scenariusze testowe muszą być za każdym razem aktualizowane. Inną dużą niedogodnością jest samo konstruowanie przypadków. Przez to, że każdy musi mieć bardzo szczegółowy opis zajmuje to stosunkowo dużo czasu, uwagi i dokładności, przez to aby ułatwić pracę często stosuje się scenariusze testowe z jednym przypadkiem wykonywania, ale kilkoma wejściowymi parametrami i różnymi oczekiwanymi wynikami. W ten sposób otrzymujemy listę kontrolną z poprzednimi krokami. W ramach zaimplementowanej pracy przedstawię kilka zaprojektowanych scenariuszy złożonych z przypadków testowych związanych z najważniejszymi funkcjonalnościami aplikacji.

ST1: Stworzenie procesu z zadań

P1.1	Zainicjowanie procesu z zadaniem	
Cel:	Stworzenie instancji procesu z jednym poprawnie skonfigurowanym zadaniem	
Warunki początkowe:		
<ul style="list-style-type: none">• użytkownik powinien być zalogowany• w systemie powinien być przynajmniej jeden klient		
Krok	Oczekiwany rezultat	Status
1. Przejdź do zakładki „Stream Creator”.	Pojawił się formularz konfiguracji podstawowych informacji o procesie.	OK
2. Wypełnij podane pola formularza i kliknij przycisk „Create Standard”.	Formularz znika, odsłaniając ekran kreatora procesów oraz wyświetla się komunikat o poprawnie zdefiniowanym procesie.	OK
3. Kliknij w dowolnym miejscu w strefie wyznaczonej niebieską przerywaną linią.	Na ekranie ukazało się szare koło symbolizujące zadanie w procesie.	OK
4. Przenieś zadanie w dowolne miejsce w wyznaczonym obszarze.	Przeciągnięte zadanie przeniosło się w wyznaczone miejsce.	OK
5. Dwukrotnie kliknij lewym przyciskiem myszy na zadaniu.	Zadanie zmieniło kolor na żółty, oznaczając się jako wybrane.	OK

6. Wybierz przycisk „Configure selected job”.	Wyświetlił się formularz konfiguracji zadania.	OK
7. Uzupełnij formularz wymaganymi danymi. Zatwierdź wybierając przycisk „Save”.	Formularz zniknął, pod zadaniem pojawiła się jego nazwa, a kolor zmienił się w zależności od statusu. (TO DO – czerwony, IN PROGRESS – niebieski, DONE – zielony).	OK

Tabela 6.1 I Przypadek testowy

P1.2	Raportowanie czasu na zadanie przy tworzeniu procesu	
Cel:	Poinformowanie użytkownika o braku możliwości raportowania na statusie innym niż IN PROGRESS	
Warunki początkowe: <ul style="list-style-type: none">• przejście przez przypadek P1.1 do kroku nr. 6		
Krok	Oczekiwany rezultat	Status
1. Uzupełnij formularz wymaganymi danymi, w polu „Time Spent” zmień wartość na inną niż 0, a w polu status wybierz wartość „TO DO” i kliknij „Save”.	Wyświetlił się komunikat błędu o braku możliwości raportowania na zadanie które jest w statusie innym niż „IN PROGRESS”.	OK
2. W polu status wybierz wartość „DONE” i kliknij „Save”.	Wyświetlił się komunikat błędu o braku możliwości raportowania na zadanie które jest w statusie innym niż „IN PROGRESS”.	OK
3. W polu status wybierz wartość „IN PROGRESS” i kliknij „Save”.	Formularz zniknął, pod zadaniem pojawiła się jego nazwa, a kolor zadania zmienił się na niebieski.	OK

Tabela 6.2 II Przypadek testowy

P1.3	Hierarchia zadań w procesie	
Cel:	Stworzenie hierarchii zadań w instancji procesu	
Warunki początkowe:		
<ul style="list-style-type: none">przejęcie przez przypadek P1.1powtórzenie kilkakrotnie kroków od 3 do 7 z P1.1		
Krok	Oczekiwany rezultat	Status
1. Kliknij w dowolnym pustym miejscu w strefie wyznaczonej niebieską przerywaną linią.	Na ekranie ukazało się nowe szare koło.	OK
2. Kliknij przycisk „Building Edges”.	Przycisk zmienił się na zielony, a dwa pozostałe wyszarzały.	OK
3. Kliknij w dowolne skonfigurowane zadanie (nie szare).	Wybrane zadanie to zmieniło kolor na fioletowy.	OK
4. Kliknij w to samo zadanie.	Wyświetlił się komunikat błędu.	OK
5. Kliknij w zadanie szare.	Wyświetlił się komunikat błędu.	OK
6. Kliknij w inne skonfigurowane	Stworzyło się połączenie w postaci	OK

zadanie.	strzałki.	
7. Kliknij w zadanie do którego prowadzi strzałka.	Wybrane zadanie to zmieniło kolor na fioletowy.	OK
8. Kliknij w zadanie z którego prowadzi strzałka.	Wyświetlił się komunikat błędu.	OK
9. Kliknij w inne skonfigurowane zadanie.	Stworzyło się połączenie w postaci strzałki.	OK
10. Kliknij w zadanie do którego prowadzi ostatnia strzałka.	Wybrane zadanie to zmieniło kolor na fioletowy.	OK
11. Kliknij w zadanie z którego prowadzi pierwsza strzałka, tak aby stworzyć cykl.	Wyświetlił się komunikat błędu.	OK

Tabela 6.3 III Przypadek testowy

P1.4	Tworzenie procesu z szablonu	
Cel:	Sklonowanie instancji procesu w celu automatycznego stworzenia szablonu	
Warunki początkowe: <ul style="list-style-type: none">• stworzony szablon procesu ze skonfigurowaną hierarchią zadań		
Krok	Oczekiwany rezultat	Status
1. Wejść w zakładkę „Streams”.	Wyświetlił się widok list procesów.	OK
2. Przy użyciu czarnej strzałki w dół u góry ekranu wybierz listę „Template List”.	Wyświetliła się lista szablonów dostępnych w aplikacji.	OK
3. Wybierz szablon, z którego chcesz stworzyć proces.	Wyświetlił się widok strony szablonu.	OK
4. Kliknij przycisk „Create Stream From Template”.	Wyświetlił się widok strony nowo powstałego procesu z nowo powstałymi zadaniami z szablonu. Komunikat poinformował o poprawnym stworzeniu procesu.	OK

Tabela 6.4 IV Przypadek testowy

ST2: Edycja procesu

P2.1	Usunięcie elementów procesu	
Cel:	Usunięcie zadania i hierarchii w kreatorze procesów.	
Warunki początkowe: <ul style="list-style-type: none">stworzona instancja procesów składająca się z kilku połączonych zadań		
Krok	Oczekiwany rezultat	Status
5. Przejść na zakładkę „Streams” zawierającą procesy	Wyświetliła się przypięta lista procesów.	OK
6. Wybierz czarną strzałkę skierowaną w dół nad widokiem listy.	Rozwinęła się lista dostępnych list procesów.	OK
7. Wybierz listę „Stream List”	Wyświetliła się lista wszystkich procesów.	OK

8. Wybierz proces mający kilka zadań i hierarchię.	Przeglądarka przekierowała na ekran instancji procesu.	OK
9. Wybierz przycisk „Edit Using Creator”	Wyświetlił się komponent z kreatorem procesów.	OK
10. Kliknij w dowolne zadanie.	Zadanie zmieniło kolor na żółty, a powiązane krawędzie na kolor niebieski.	OK
11. Kliknij przycisk „Delete selected element”	Z komponentu zniknął zaznaczony proces i powiązane z nim krawędzie. Wyświetlił się komunikat o poprawności usunięcia.	OK
12. Kliknij w dowolną krawędź.	Krawędź zmieniła kolor na niebieski.	OK
13. Kliknij przycisk „Delete selected element”	Z komponentu zniknęła zaznaczona krawędź.	OK

Tabela 6.5 V Przypadek testowy

P2.2	Edycja widoku hierarchii w procesie	
Cel:	Możliwość rekonfiguracji wizualnej reprezentacji zadań i ich hierarchii	
Warunki początkowe: <ul style="list-style-type: none">przejsięcie przez przypadek P2.1 do punktu 5 włącznie		
Krok	Oczekiwany rezultat	Status
1. Przeciągnięcie kilku dowolnych zadań w dowolne miejsce w celu zmiany wyglądu procesu.	Zadanie przesunęły się.	OK
2. Wyjdź z edytora procesów klikając „Cancel” lub krzyżyk u góry komponentu.	Komponent zniknął.	OK
3. Odśwież stronę.	Strona odświeżyła się ukazując ułożony proces.	OK

Tabela 6.6 VI Przypadek testowy

P2.3	Edycja zadania	
Cel:	Edycja szczegółów zadania w edytorze procesów	
Warunki początkowe: <ul style="list-style-type: none">przejsście przez przypadek P2.1 do punktu 5 włącznie		
Krok	Oczekiwany rezultat	Status
1. Kliknij w dowolne skonfigurowane zadanie (nie szare).	Kolor zadania zmienił się na żółty.	OK
2. Kliknij w przycisk „Configure selected job”.	Wyświetlił się formularz zadania.	OK
3. Wyedytuj element zadania i kliknij przycisk „Save”	Formularz zamknął się i wyświetlił się komunikat o poprawnym zapisaniu procesu.	OK
4. Zamknij edytor procesów klikając przycisk „Cancel” lub czarny krzyżyk u	Edytor procesu zamknął się.	OK

góry komponentu.		
5. U dołu strony przejdź do zakładki „Related”.	Dolna część układu strony zmieniła się na widok powiązanych elementów.	OK
6. Kliknij w nazwę zmodyfikowanego zadania.	Aplikacja przekierowała do widoku strony zadania.	OK
7. Sprawdź na widoku strony czy zmiany zostały naniesione.	Szczegóły zadania zaktualizowały się.	OK

Tabela 6.7 VII Przypadek testowy

ST3: Prośba o przedłużenie ostatecznej planowanej daty zakończenia zadania

P3.1	Prośba o przedłużenie ostatecznej planowanej daty zakończenia zadania – akceptacja	
Cel:	Wysłanie i zaakceptowanie procesu o przedłużeniu ostatecznej planowanej daty zakończenia zadania	
Warunki początkowe: <ul style="list-style-type: none">• skonfigurowane zadanie z pracownikiem jako osoba przypisana• dostęp do konta menadżera i pracownika		
Krok	Oczekiwany rezultat	Status
1. Zaloguj się do systemu jako pracownik.	Wyświetlił się główny ekran aplikacji.	OK
2. Wybierz zadanie do którego jest przypisany pracownik.	Wyświetliła się strona zadania.	OK
3. Używając panelu po prawej stronie wypełnij nową datę ostatecznej planowanej daty zakończenia zadania i kliknij przycisk „Submit Due Date”	Wyświetlił się komunikat o wysłaniu procesu akceptacji do menadżera użytkownika.	OK
4. Przeloguj się na konto menadżera.	Zalogowano jako menadżer.	OK
5. W prawym górnym rogu pojawiło się powiadomienie z oczekującą decyzją procesu akceptacji. Kliknij na nie.	Przekierowano do ekranu procesu akceptacji,	OK
6. Kliknij przycisk „Approve” w celu akceptacji zmiany ostatecznej daty zakończenia zadania.	Wyświetliło się okno z prośbą o komentarz do akceptacji.	OK
7. Wpisz przykładowy komentarz do akceptacji i kliknij przycisk „Approve”.	Przekierowano do strony procesu akceptacji. Proces akceptacji ma teraz znacznik „Approved”.	OK
8. Wejdź na zadanie, które dotyczyło procesu akceptacji i sprawdź czy ostateczna planowana data zakończenia została zmieniona.	Ostateczna planowana data zakończenia zadania zmieniła się.	OK

Tabela 6.8 VIII Przypadek testowy

P3.2	Prośba o przedłużenie ostatecznej planowanej daty zakończenia zadania – odrzucenie	
Cel:	Wysłanie i odrzucenie procesu o przedłużeniu ostatecznej planowanej daty zakończenia zadania	
Warunki początkowe: <ul style="list-style-type: none">• Przejdźcie przez przypadek P3.1 do punktu 5 włącznie		
Krok	Oczekiwany rezultat	Status
1. Kliknij przycisk „Reject”.	Wyświetliło się okno z prośbą o komentarz do decyzji.	OK
2. Wpisz przykładowy komentarz do akceptacji i kliknij przycisk „Reject”.	Przekierowano do strony procesu akceptacji. Proces akceptacji ma teraz znacznik „Rejected”.	OK
3. Wejdź na zadanie, które dotyczyło procesu akceptacji i sprawdź czy ostateczna planowana data zakończenia została zmieniona.	Ostateczna planowana data zakończenia zadania nie zmieniła się.	OK

Tabela 6.9 IX Przypadek testowy

ST4: Powiadomienie o nadchodzących zadaniach

P4.1	Prośba o przedłużenie ostatecznej planowanej daty zakończenia zadania – akceptacja		
Cel:	Wysłanie i zaakceptowanie procesu o przedłużeniu ostatecznej planowanej daty zakończenia zadania		
Warunki początkowe: <ul style="list-style-type: none">• dostęp do adresu email pracownika ze skonfigurowanym zadaniem z dzisiejszą datą jako planowana ostateczna data zakończenia• dostęp do konta menadżera			
Krok	Oczekiwany rezultat	Status	
1. Zaloguj się do systemu jako menadżer.	Wyświetlił się ekran główny menadżera.	OK	
2. Kliknij w zębatkę w prawnym górnym rogu, a następnie przejdź do opcji „Developer console”.	Wyświetliło się nowe okno z konsolą developerską.	OK	
3. Pod zakładką przejdź do „Debug” -> „Open Execute anonymous window”	Wyświetliło się okno do wpisania kodu.	OK	
4. Zasymuluj odpalenie mechanizmu powiadomień wpisując w puste miejsce „Database.executeBatch(new DailyReminderBatch(), 90);” i kliknij przycisk „Execute”	Okno zniknęło i zostałeś przekierowany do okna konsoli.	OK	
5. Sprawdź email pracownika.	Pracownik otrzymał email z powiadomieniem o zadaniach na dziś.	OK	

Tabela 6.10 X Przypadek testowy

7 Podsumowanie

Wielu drobnych przedsiębiorców słysząc o metodykach zwinnych w prowadzeniu działalności od razu spotyka się ze ścianą złożoną z niedostatecznych umiejętności analitycznych, nieznajomości podstawowych notacji czy też niewiedzy w jaki sposób można rekonfigurować posiadane zasoby oraz procesy w taki sposób, aby zmaksymalizować przychody i zadowolenie pracowników. Nie zdając sobie sprawy, że prób wejścia do wprowadzenia elementów organizacji zwinnej może być na wyciągnięcie ręki. Wirtualizacja przedsiębiorstwa będąca kluczowym elementem tego procesu jest w stanie poprawić działalność takiej organizacji, poprawiając komunikację między pracownikami i z klientami, monitorując przebieg prac i szybko reagować na zmieniający się rynek czy też w sytuacjach kryzysowych. Systemy związane z zarządzaniem relacjami z klientami (CRM) wydają być się idealnym rozwiązaniem dla tego typu osób prowadzących działalność. Dodatkowo mając narzędzie do budowania, rekonfiguracji, śledzenia i wyciągania statystyk z prowadzonych przez każdego pracownika procesów – daje to duże możliwości kontroli nad tym co aktualnie dzieje się w firmie.

7.1 Efekt końcowy

W ramach tej pracy udało się zaprojektować, a następnie zaimplementować i przetestować aplikację na jednej z największych platform CRM – zapewnia nam ona przejrzysty interfejs, możliwość prostej integracji z zewnętrznymi systemami przy rozwijaniu oraz elementy przydatne przy wirtualizacji przedsiębiorstwa (tworzenie grup dyskusyjnych, komunikacja wewnątrz firmy i z klientem bezpośrednio z poziomu platformy). Spełnione zostały wymagania przedstawione w analizie potrzeb użytkowników. Pracownicy logują się do systemu, w którym mają wszystkie potrzebne informacje dotyczące klientów i zadań jakie mają wykonać, a wszystko to z poziomu telefonu, tabletu lub komputera. Wspomniane we wstępie elementy związane z wizualizacją procesów i ich edytowaniem reprezentujących wykonywane zadania w firmie zostały spełnione przy pomocy funkcjonalności związanych z konfiguratorem procesów. Dzięki tworzeniu instancji przebiegów z tak zwanych szablonów (wcześniej zdefiniowanych, powtarzalnych procesów) nie ma potrzeby każdorazowego jego budowania, co przyspiesza pracę i poprawia wydajność. Sam proces buduje się przy pomocy prostego edytora graficznego. Pozwala to łatwo, niezależnie od wykształcenia technicznego i znajomości specjalistycznych notacji, zrozumieć jego działanie. Jest to istotny aspekt przy adaptacji mniejszych przedsiębiorstw niezaznajomionych z podobnymi narzędziami co przekłada się na mniejsze koszty szkoleń pracowników. Dodatkowo, dzięki powiadomieniom, raportom z wykonywanych zadań, pracownicy są w stanie reagować na zaniechania dotyczące poszczególnych klientów. Mając taki mechanizm przedsiębiorstwo ma wgląd na to jak funkcjonuje i jakie elementy można w nim poprawić. Należy nadmienić, że system dając większą niezależność pracownikom, dzięki możliwościom samoorganizacji pracy przy pomocy kalendarza czy też przedstawionych tak zwanych akcji powiązanych, daje możliwość obserwowania postępów prac kadrze menadżerskiej za pomocą przygotowanych raportów. Obie te rzeczy w połączeniu ze sobą dają możliwość szybkiego reagowania przy sytuacjach wyjątkowych. Wraz z implementacją programu zostały napisane również testy jednostkowe sprawdzające większość funkcjonalności aplikacji od strony przygotowanych danych, wywołań kodu a następnie oczekiwanych

rezultatów. Napisane zostały również scenariusze testowe złożone z przypadków najważniejszych elementów systemu.

7.2 Możliwości dalszego rozwoju

Dzięki wzorcu delegacji w wyzwalaczach dopisywanie kolejnych funkcjonalności w aplikacji nie powinno sprawić dużej trudności. Dodatkowo jego konstrukcja chroni programistów przed natrafieniem na limity wywołań zintegrowanej bazy danych platformy przez stworzenie mało wydajnej transakcji.

W zależności od potrzeb przedsiębiorstwa i jego dynamiczności mogłaby nastąpić potrzeba implementacji integracji z serwisem odpowiadającym za powiadomienia SMS, dzięki którym pracownicy mogliby jeszcze szybciej reagować zbliżające się obowiązki czy też pilnować dzisiejszych zadań.

Modelowanie procesów wydaje się być stosunkowo prymitywne. Realizacja prostych procesów z pewnością jest niewystarczająca biorąc pod uwagę złożoność funkcjonowania przedsiębiorstw. Takie rzeczy jak przebiegi warunkowe, zależne od rezultatu zadania, czy też możliwość dodania większej ilości następników do zadania wydają się być niezbędnym kierunkiem. Co więcej bardzo atrakcyjną funkcjonalnością mogła by być aktualizacja zadań na podstawie statusów powiązanych elementów w zewnętrznych narzędziach – na przykład Trello. Firmy korzystają wielu narzędzi, dzięki integracji ich praca byłaby na pewno lepiej ustrukturyzowana.

Elementy związane z monitorowaniem procesów mogłyby być rozwinięte w stronę większej automatyzacji. Obecnie menadżer musi obserwować zmieniające się raporty, aby wyciągać wnioski z realizowanych zadań, co z pewnością jest uciążliwe. Na przykład automatyczny mechanizm bazujący na przekroczeniu pewnej liczby godzin lub pieniędzy w miesiącu lub wyliczający różnice w efektywności byłby na pewno przydatny.

Rozwijając kolejne elementy systemu należy jednak pamiętać o tym, że jego istotną zaletą jest prostota działania. Zbytnie skomplikowanie części funkcjonalności może przełożyć się na zwiększenie progu wejścia dla mniejszych przedsiębiorców nie mających doświadczenia z podobnymi narzędziami.

8 Bibliografia

- [1] CompaTIA, IT Industry outlook 2019, Dostęp: <https://www.comptia.org/resources/it-industry-trends-analysis> [10.07.2019]
- [2] Andrzej Olak, Organizacja zwinna – wyznaczniki oraz kierunki strategii prowadzące do zwinności przedsiębiorstwa, „e-mentor” 2017, nr 1(68), s. 48–54 Dostęp: http://www.e-mentor.edu.pl/pdf/68/art_48-54_Olak_Ementor%201_68_2017_with_metadata.pdf [10.07.2019]
- [3] Zeszyty Naukowe Politechniki Poznańskiej. Wydawnictwo Politechniki Poznańskiej, Seria: Organizacja i Zarządzanie, nr.71, Hanna Włodarkiewicz-Klimek, Koncepcja i modele

zwinnego przedsiębiorstwa, rok wydania: 2016, Identyfikator cyfrowy dokumentu DOI: 10.21008/j.0239-9415.2016.071.19 [10.07.2019]

[4] Encyklopedia Zarządzania, Dostęp: https://mfiles.pl/pl/index.php/System_CRM [11.07.2019]

[5] Salesforce, Dostęp: <https://www.salesforce.com> [11.07.2019]

[6] The benefits of the Salesforce AppExchange for SMB and enterprise businesses, Dostęp: <https://blog.pandadoc.com/salesforce-appexchange-for-smb-and-enterprise-businesses/> [11.07.2019]

[7] Microsoft Dynamics AppSource, Dostęp: <https://appsource.microsoft.com/en-us/> [11.07.2019]

[8] Salesforce Appexchange, ProcessClick - Your Agile BPM for Salesforce, Dostęp: <https://appexchange.salesforce.com/appxListingDetail?listingId=a0N30000000rJ1OEAU> [11.06.2019]

[9] Oracle Process Cloud Service, Dostęp: <https://cloud.oracle.com/process> [11.07.2019]

[10] Salesforce Chatter, Dostęp: <https://www.salesforce.com/products/chatter/overview/> [11.07.2019]

[11] Low-Code Process Modeling and Execution, demo aplikacji IBM Business Automation Workflow, Dostęp: <https://www.youtube.com/watch?v=WgaFvGjMH8c> [11.07.2019]

[12] Implementation of Model-View-Controller Architecture Pattern for Business Intelligence Architecture, International Journal of Computer Applications Volume 102– No.12 , Dostęp: <https://pdfs.semanticscholar.org/41a3/3127ac74f126d6ae13a49431983fbda6f7cf.pdf> [11.07.2019]

[13] LWC Documentation, Dostęp: <https://developer.salesforce.com/docs/component-library/documentation/lwc> [11.07.2019]

[14] Limitation of Developer Edition, Dostęp: <https://www.oreilly.com/library/view/salesforcecom-customization-handbook/9781849685986/ch01s10.html> [12.07.2019]

[15] LWC Introduction, Dostęp: <https://developer.salesforce.com/blogs/2018/12/introducing-lightning-web-components.html> [13.07.2019]

- [16] Document Object Model Documentation, Dostęp: https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model [12.07.2019]
- [17] D3.js Documentation, Dostęp: <https://d3js.org/> [12.07.2019]
- [18] Projektowanie i programowanie obiektowe, Roman Simiński, Dostęp: http://programowanie.siminskionline.pl/resource/wp/ood_strategy.pdf [15.07.2019]
- [19] Salesforce Object Query Language (SOQL), Dostęp: https://developer.salesforce.com/docs/atlas.en-us.soql_sosl.meta/soql_sosl/sforce_api_calls_soql.htm [13.07.2019]
- [20] Salesforce Object Search Language (SOSL), Dostęp: https://developer.salesforce.com/docs/atlas.en-us.soql_sosl.meta/soql_sosl/sforce_api_calls_sosl.htm [13.07.2019]
- [21] Salesforce Extension Pack, Dostęp: <https://marketplace.visualstudio.com/items?itemName=salesforce.salesforcedx-vscode> [14.07.2019]
- [22] Ant Migration Tool Deployment, Dostęp: https://developer.salesforce.com/docs/atlas.en-us.apexcode.meta/apexcode/apex_deploying_ant.htm [13.07.2019]
- [23] Java Documentation, Dostęp: <https://docs.oracle.com/javase/8/docs/> [13.07.2019]
- [24] Rodzaje grafów, Dostęp: <http://student.krakow.pl/026-Ciosek-Grybow/rodzaje.html> [17.07.2019]
- [25] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein: Wprowadzenie do algorytmów.. Wyd. 8. Wydawnictwa Naukowo-Techniczne, 2007, s. 549-558. ISBN 978-83-204-3328-9
- [26] Type of edges involved in DFS, Dostęp: <https://www.techiedelight.com/types-edges-involved-dfs-relation/> [17.07.2019]
- [27] Arrival and Departure Time of Vertices in DFS, Dostęp: <https://www.techiedelight.com/arrival-departure-time-vertices-dfs/> [17.07.2019]
- [28] DAG validation, Dostęp: <https://www.techiedelight.com/check-given-digraph-dag-directed-acyclic-graph-not/> [17.07.2019]
- [29] Trigger, Apex Developer Guide, Dostęp: https://developer.salesforce.com/docs/atlas.en-us.apexcode.meta/apexcode/apex_triggers.htm [14.07.2019]

[30] Salesforce Developer Limits and Allocations Quick Reference, Dostęp:

[https://developer.salesforce.com/docs/atlas.en-](https://developer.salesforce.com/docs/atlas.en-us.salesforce_app_limits_cheatsheet.meta/salesforce_app_limits_cheatsheet/salesforce_app_limits_platform_apexgov.htm)

[us.salesforce_app_limits_cheatsheet.meta/salesforce_app_limits_cheatsheet/salesforce_app_limits_platform_apexgov.htm](https://developer.salesforce.com/docs/atlas.en-us.salesforce_app_limits_cheatsheet.meta/salesforce_app_limits_cheatsheet/salesforce_app_limits_platform_apexgov.htm) [16.07.2019]

[31] Testy poziomy i typy, Agnieszka Pieszczyk, Dostęp:

<https://codecouple.pl/2016/02/17/testy-poziomy-i-typy/> [16.07.2019]

9 Spis rysunków, tabel oraz listingów

Spis rysunków

Rysunek 1.1 Relacje między czynnikami kształtującymi zwinność przedsiębiorstwa. [3].....	9
Rysunek 2.1 Materiał reklamowy przedstawiający fragment procesu w ProcessClick. [8]	13
Rysunek 2.2 Wygląd aplikacji Oracle Process Cloud na różnych urządzeniach. [9]	14
Rysunek 2.3 Klatka z filmu prezentującego demo aplikacji IBM Business Automation Workflow. [11]	16
Rysunek 4.1 Model danych aplikacji	26
Rysunek 4.2 Widok zakładek aplikacji z oznaczeniem kreatora	28
Rysunek 4.3 Formularz inicjujący proces	28
Rysunek 4.4 Dodanie wierzchołka do procesu	29
Rysunek 4.5 Zainicjowanie konfiguracji zadania w procesie	30
Rysunek 4.6 Porównanie formularzy zadań dla instancji procesu oraz szablonu	30
Rysunek 4.7 Konfigurator procesu w trybie dodawania krawędzi	31
Rysunek 4.8 Zainicjowanie tworzenia krawędzi między wierzchołkami	31
Rysunek 4.9 Proces z utworzoną hierarchią zadań	32
Rysunek 4.10 Strona rekordu procesu	32
Rysunek 4.11 Edytor rekordu procesu przy pomocy konfiguratora	33
Rysunek 4.12 Opcje znalezienia szablonu procesu	33
Rysunek 4.13 Komponent z nadchodzącymi zadaniami użytkownika	34
Rysunek 4.14 Komponenty z dzisiejszymi wydarzeniami i akcjami powiązanymi użytkownika	35
Rysunek 4.15 Komponent z raportami czasowymi pracy użytkownika	35
Rysunek 4.16 Statystyka o procencie niedoszacowanych zadań	36
Rysunek 4.17 Wskaźnik niedosetymowanych godzin w danym miesiącu	36
Rysunek 4.18 Statystyka miesięcznych zarobków użytkownika	37
Rysunek 4.19 Tablica z wykresami dotyczącymi klientów	38
Rysunek 4.20 Przykładowa wiadomość email opisująca stworzenie akcji powiązanej	38
Rysunek 4.21 Wiadomość email opisująca listę zadań z prawdopodobnym przekroczeniem ostatecznej zaplanowanej daty zakończenia	39
Rysunek 4.22 Wiadomość email opisująca listę zadań zaplanowanych na dziś	39
Rysunek 4.23 Wykres kosztów miesięcznych z wyróżnieniem klientów	40

Rysunek 4.24 Miesięczne koszty poszczególnych pracowników.....	41
Rysunek 4.25 Sumaryczne koszty zadań z podziałem na branże klientów	41
Rysunek 4.26 Zestaw wykresów opisujących zaraportowany czas przez pracowników	42
Rysunek 4.27 Procent wszystkich niedoestymowanych zadań	43
Rysunek 4.28 Procent zadań niedostarczonych w terminie z podziałem na klienta.....	43
Rysunek 4.29 Sumaryczna liczba godzin ponad estymacje zadań.....	43
Rysunek 4.30 Komponent do wysłania prośby o zmianę daty ostatecznego zakończenia zadania	44
Rysunek 4.31 Powiadomienie o wysłaniu procesu akceptacji	44
Rysunek 4.32 Strona procesu akceptacji	45
Rysunek 5.1 Rodzaje krawędzi przy przeszukiwaniu w głąb. [26]	52
Rysunek 6.1 Raport z testów jednostkowych i pokrycia klas	60

Spis tabel

Tabela 2.1 Ocena rozwiązania ProcessClick	14
Tabela 2.2 Ocena rozwiązania Oracle Process Cloud	15
Tabela 2.3 Ocena rozwiązania IBM Business Automation Workflow.....	16
Tabela 3.1 I przypadek użycia	18
Tabela 3.2 II przypadek użycia	18
Tabela 3.3 III przypadek użycia	18
Tabela 3.4 IV przypadek użycia	19
Tabela 3.5 V przypadek użycia	19
Tabela 3.6 VI przypadek użycia	19
Tabela 3.7 VII przypadek użycia	19
Tabela 3.8 VIII przypadek użycia	20
Tabela 3.9 IX przypadek użycia	20
Tabela 3.10 X przypadek użycia	20
Tabela 3.11 XI przypadek użycia	21
Tabela 3.12 XII przypadek użycia	21
Tabela 3.13 XIII przypadek użycia	22
Tabela 3.14 XIV przypadek użycia.....	22
Tabela 3.15 XV przypadek użycia.....	22

Tabela 3.16 XVI przypadek użycia	22
Tabela 3.17 XVII przypadek użycia	23
Tabela 3.18 XVIII przypadek użycia	23
Tabela 4.1 Legenda znaczeń koloru wierzchołków w konfiguratorze procesów	29
Tabela 4.2 Pola z dostępnymi wartościami na obiekcie akcji powiązanej.....	34
Tabela 6.1 I Przypadek testowy	62
Tabela 6.2 II Przypadek testowy	62
Tabela 6.3 III Przypadek testowy	63
Tabela 6.4 IV Przypadek testowy.....	63
Tabela 6.5 V Przypadek testowy.....	64
Tabela 6.6 VI Przypadek testowy.....	64
Tabela 6.7 VII Przypadek testowy.....	65
Tabela 6.8 VIII Przypadek testowy.....	65
Tabela 6.9 IX Przypadek testowy.....	66
Tabela 6.10 X Przypadek testowy.....	66

Spis listingów

Listing 5.1 Widok komponentu	47
Listing 5.2 Stylowanie komponentu	47
Listing 5.3 Kontroler komponentu.....	48
Listing 5.4 Model komponentu	49
Listing 5.5 Implementacja algorytmu walidującego acykliczność grafu skierowanego	53
Listing 5.6 Implementacja przeszukiwania w głąb z wyznaczeniem czasu odejścia	53
Listing 5.7 Wyzwalacz delegujący zadanie.....	55
Listing 5.8 Implementacja wzorca delegacji wyznaczającego kolejność działań 1/2.....	55
Listing 5.9 Implementacja wzorca delegacji wyznaczającego kolejność działań 2/2.....	56
Listing 5.10 Przykładowa implementacja obsługi zadań wyzwalacza	57

10 Załącznik A. Zawartość płyty CD

W skład płyty CD wchodzi:

- plik MaciejSuchocki.pdf zawierający niniejszą pracę
- folder Aplikacja zawierający kod programu
- folder Instrukcje zawierający opis dotyczący zalogowania się do aplikacji na platformie jako menadżer lub jeden z pracowników