POLITECHNIKA POZNAŃSKA

WYDZIAŁ AUTOMATYKI, ROBOTYKI I ELEKTROTECHNIKI
Instytut Robotyki i Inteligencji Maszynowej

Praca dyplomowa inżynierska

AUTOMATYZACJA PROCESU PLANOWANIA WYDATKÓW NA USŁUGI IT W VW POZNAŃ Z WYKORZYSTANIEM MICROSOFT POWER PLATFORM

Remigiusz Wolniak, 151192
Michał Gajdzis, 151066

Promotor
dr hab. inż. Piotr Kaczmarek

POZNAŃ 2025

Zastrzeżenie dotyczące treści pracy dyplomowej

Niniejsza praca inżynierska zawiera treści, informacje itp. udostępnione przez spółkę Volkswagen Poznań Sp. z o.o. z siedzibą przy ulicy Warszawskiej 349, 61-060 w Poznaniu, mogące stanowić tajemnice przedsiębiorstwa tej spółki i mogące być wykorzystane wyłącznie dla potrzeb napisania niniejszej pracy. Wobec powyższego niedozwolone jest wykorzystywanie całości lub części niniejszej pracy, a także udostępnianie całości lub części pracy komukolwiek jak również kopiowanie, powielanie, publikowanie itp. bez pisemnej zgody spółki Volkswagen Poznań Sp. z o.o. -– zastrzeżenie to nie ma zastosowania do przypadku udostępnienia niniejszej pracy nauczycielom akademickim w celu oceny, recenzji i obrony ww. pracy. Podmioty, które naruszą powyższy zakaz ponoszą odpowiedzialność odszkodowawczą wobec spółki Volkswagen Poznań Sp. z o.o.

Tutaj będzie skan karty pracy dyplomowej.

Spis treści

Rozdział 1

Wstęp

Współczesny świat biznesu stawia coraz większe wymagania wobec przedsiębiorstw, zarówno w zakresie wydajności procesów, jak i precyzji podejmowanych decyzji. Tradycyjne metody zarządzania i przetwarzania danych, oparte na pracy manualnej i mało efektywnych narzędziach, stają się niewystarczające w obliczu rosnącej skali operacji oraz konieczności szybkiego i niezawodnego podejmowania decyzji. W odpowiedzi na te wyzwania coraz większą rolę odgrywają rozwiązania z zakresu automatyzacji biurowej, które pozwalają na oszczędność czasu i zasobów, usprawnienie kluczowych procesów organizacyjnych oraz minimalizację ryzyka błędów ludzkich.

Jednym z obszarów, w którym automatyzacja znajduje zastosowanie, jest zarządzanie usługami IT i powiązanymi kosztami. W dużych organizacjach o rozbudowanej strukturze, konieczność gromadzenia, analizy oraz weryfikacji danych finansowych stanowi poważne wyzwanie. Dzięki wdrożeniu odpowiednich narzędzi, procesy te mogą być prowadzone w sposób uporządkowany i efektywny, umożliwiając jednocześnie bieżącą kontrolę nad wydatkami oraz lepsze planowanie budżetowe.

Ustandaryzowany i zautomatyzowany przepływ informacji ogranicza ryzyko powielania błędów i pozwala na skrócenie czasu potrzebnego na wykonanie poszczególnych zadań. Dodatkowo, wdrożenie automatyzacji zapewnia większą przejrzystość i ułatwia dostęp do informacji każdemu uczestnikowi procesu.

W dobie intensywnej cyfryzacji przedsiębiorstw oraz dynamicznego rozwoju technologii, automatyzacja biurowa staje się konieczna, aby sprostać wymaganiom współczesnego rynku. Odpowiednio zaprojektowane systemy i narzędzia wspierają nie tylko wydajność operacyjną, ale także strategiczne zarządzanie zasobami, umożliwiając rozwój w innych obszarach swojej działalności.

Celem pracy jest opracowanie aplikacji usprawniającej proces podejmowania decyzji dotyczących zakupu *usług IT*[1] na najbliższy rok kalendarzowy. Praca została wykonana z wykorzystaniem *Power Platform* oraz *SharePoint*, które są integralną cześcią pakietu *Microsoft 365*. Zdecydowano się na wybór tego rozwiązania, ponieważ pozwala ono na prostą integrację między programami wchodzącymi w skład pakietu. Ponadto, każdy z uczestników procesu ma dostęp do wspomnianych serwisów, co pozwala uniknąć dodatkowych kosztów.

DOPISAĆ:

Struktura pracy jest następująca. W rozdziale 2. przedstawiono powód wykonywanego zadania, wraz z jego wyjaśnieniem oraz opisem wykorzystanych kompontenów

[1] *Usługi IT* należy rozumieć jako licencje oraz klucze dostępu do używanych systemów informatycznych.

Rozdział 3 jest poświęcony założeniom projektowym i architekturze . . . (kilka zdań).
Rozdział 4 zawiera implementację . . . (kilka zdań) . . . itd.
Rozdział X stanowi podsumowanie pracy.

W przypadku prac inżynierskich zespołowych lub magisterskich 2-osobowych, po tych dwóch w/w akapitach musi w pracy znaleźć się akapit, w którym będzie opisany udział w pracy poszczególnych członków zespołu. Na przykład:

Jan Kowalski w ramach niniejszej pracy wykonał projekt tego i tego, opracował . . . Grzegorz Brzęczyszczykiewicz wykonał . . . , itd.

Rozdział 2

Podstawy teoretyczne

2.1 Struktura procesu

Przedmiotem omawianego procesu jest podjęcie decyzji o zakupie usług IT w zakładzie Volkswagen Poznań. Proces ten polega na wielokrotnej wymianie uwag dotyczących wcześniej używanego lub nowego oprogramowania między oddziałem Volkswagena w Poznaniu a zakładem z siedzibą w Wolfsburgu.

W wyniku wymiany zdań zapada decyzja o zakupie lub rezygnacji z wybranego produktu. Procedura, zazwyczaj podzielona na cztery *indykacje*[1], rozpoczyna się na początku czerwca i trwa do końca roku.

Efektem podejmowanych działań jest nabycie odpowiedniej liczby potrzebnych uprawnień licencyjnych. Przy podejmowaniu decyzji kluczowymi aspektami są:

- liczba użytkowników danego oprogramowania,
- cena zakupu w porównaniu z rokiem poprzednim,
- określenie, czy dana usługa zostanie w pełni wykorzystana biorąc pod uwagę poprzednie kryteria.

Dotychczas analiza i przetwarzanie danych odbywały się przy użyciu arkuszy kalkulacyjnych programu Excel, a wymiana informacji między jednostkami była realizowana za pomocą wiadomości e-mail.

2.1.1 Gromadzenie danych dotyczących ofert usługodawców

Informacje na temat serwisów są zbierane na początku roku, przed rozpoczęciem cyklu procesu. W tym czasie, prowadzone są rozmowy między menadżerami odpowiedzialnymi za dane rozwiązanie (*BSM*, ang. *Business Service Manager*) a firmami świadczącymi usługi, w celu otrzymania zaaktualizowanych wiadomości związanych z ich produktami. Na podstawie danych od usługodawców oraz menadżerów, powstaje arkusz, który jest przekazywany do zakładu w Poznaniu.

2.1.2 Przygotowanie danych

Otrzymany arkusz kalkulacyjny, zawiera tabelę o strzukturze kolumn podobnej do tabeli 2.1. Brakuje w nim jednak informacji kluczowych do rozpoczęcia cyklu. Dlatego pierwszym krokiem

[1] *indykacja* – wstępne głosowanie

jest przygotowanie danych przez osobę nadzorującą proces ze strony odziału w Poznaniu. Jej zadaniem jest manualne przypisanie numeru określającego miejsce powstawania kosztów, wewnętrznie nazywanego *MPK*. Numer ten definiuje konkretną jednostkę należącą do obszaru IT, która decyduje o zakupie danego produktu. Ponadto, dodawana jest kolumna, w której znajduje się wyliczona różnica cen między rokiem obecnym a poprzednim, w celu określenia czy koszt wzrósł lub zmalał. Tak przetworzony plik zostaje umieszczony we wspólnej przestrzeni dyskowej, co umożliwia pozostałym uczestnikom procesu przystąpienie do analizy oraz dalszego przetwarzania zawartych w nim informacji.

Tabela 2.1: Nagłówki kolumn z arkusza kalkulacyjnego z roku 2022

Service group	Service main group	Service sub group	Business Service	ID	Business Service Manager	Unit of Measurement	PL70 2022 PLAN EUR w KVA	QTY	PL71 2023 PLAN EUR w KVA	QTY

2.1.3 Przebieg Iteracji

W trakcie trwania iteracji analizowane są kluczowe informacje, takie jak:

- jednostka miary (ang. *Unit of Measurement*),
- decyzja podjęta w roku poprzednim,
- cena oraz liczba użytkowników w roku obecnym,
- cena oraz liczba użytkowników w roku przyszłym.

Po analizie i porównaniu danych z wcześniejszych lat, w arkuszu powstają kolejne kolumny. Ich struktura nie jest określona przez żaden standard, ale zazwyczaj zawierają one:

- Komentarz wewnętrzny,
- Status,
- Komentarz klienta.

Komentarz wewnętrzny nie jest wymagany dla każdego serwisu. Jest on zapisywany w celu skonsultowania decyzji ze współpracownikami.
Status określa wstępną, wymaganą decyzję (Zaakceptowany/Niezaakceptowany).
Komentarz klienta zawiera uzasadnienie podjętej decyzji ze strony Volkswagen Poznań.
Tak uzupełniony arkusz zostaje przekazany pośrednio przez zakład w Wolfsburgu, do zarządu firmy.

Kolejnym etapem jest analiza tych informacji przez wcześniej wymienione podmioty. Ich zadaniem jest konfrontacja podjętej decyzji. Dodawane są kolejne kolumny:

- Komentarz BSM,
- Komentarz K-DES.

Komentarz BSM jest to odpowiedź ze strony menadżera usługi.
Komentarz K-DES (tutaj by się przydało rozszyfrować co to K-DES z niemieckiego) natomiast jest odpowiedzią międzynarodowego zarządu firmy.

Zaaktualizowany plik powraca do Volkswagen Poznań, rozpoczynając tym samym kolejną iterację procesu.

Jak wcześniej wspomniano, proces składa się zazwyczaj z czterech iteracji. Etapem kończącym cykl jest sporządzenie wymaganych dokumentów oraz faktur.

2.2 Wykorzystane technologie

Aby usprawnić przebieg procesu, zabiezpieczyć go przed błędami i usystematyzować dane, zdecydowano się na stworzenie aplikacji do jego obsługi. Głównym kryterium przy doborze technologii była powszechna dostępność do powstałego systemu wśród pracowników. Dlatego też zdecydowano się na wykorzystanie komponentów pakietu *Microsoft 365*. Pakiet ten jest bardzo rozbudowany i powszechnie wykorzystywany w firmie Volkswagen. Zawiera on programy pozwalające na stworzenie kompletnego systemu bez konieczności użycia dodatkowych serwisów.

2.2.1 Skrypty pakietu Office

Skrypty pakietu Office umożliwiają automatyzację zadań w arkuszach kalkulacyjnych programu Excel. Jedną z dostępnych funkcji jest *Action Recorder*, który pozwala na "nagranie"sekwencji kroków wykonanych przez użytkownika, a następnie przekształcenie ich na skrypt wielokrotnego użytku.

Skrypty pakietu Office są wyposażone w wbudowany *edytor kodu* (ang. *Code Editor*), oparty na języku *TypeScript*, który jest rozszerzeniem *JavaScript*. Pomimo tego, że edytor jest stosunkowo ograniczony, umożliwia stosowanie konstrukcji niedostępnych w *Action Recorder*, takich jak instrukcje warunkowe czy pętle.

Dodatkowo, program Excel pozwala na zapis skryptu w skoroszycie. Oznacza to, że każdy użytkownik mający dostęp do pliku może również uruchomić kod powiązany z danym skoroszytem.

2.2.2 SharePoint

SharePoint to platforma należąca do pakietu Microsoft 365, umożliwiająca tworzenie aplikacji webowych, takich jak witryny i strony internetowe. Jej głównym celem jest usprawnienie współpracy zespołowej poprzez dostarczenie narzędzi do publikowania informacji i raportów, które mogą być skierowane do określonych grup odbiorców.

Jednym z kluczowych zastosowań SharePointa jest zarządzanie danymi. Platforma oferuje przestrzeń do przechowywania różnego rodzaju plików, dokumentów i informacji, pełniąc funkcję serwera danych. Dzięki dostępności wbudowanych konektorów[2] (ang. *connectors*), umożliwia również wykorzystanie przechowywanych danych w procesie tworzenia aplikacji czy witryn.

Istotnym elementem środowiska SharePoint jest możliwość tworzenia list, często nazywanych *listami sharepointowymi*. Listy te mogą być wykorzystywane jako proste bazy danych, które umożliwiają dynamiczne aktualizowanie i synchronizowanie danych w czasie rzeczywistym.

SharePoint oferuje zaawansowane zarządzanie uprawnieniami. Administratorzy mogą precyzyjnie definiować dostęp użytkowników do poszczgólnych zasobów witryny co pozwala na skuteczne zabezpieczenie wrażliwych informacji.

[2] *Konektor* (ang. *connector*) -– moduł umożliwiający integrację aplikacji z usługami lub źródłami danych w celu wymiany informacji i synchronizacji systemów.

Platforma jest silnie zintegrowana z innymi usługami pakietu Microsoft 365, takimi jak Teams, Outlook czy OneDrive. Dzięki temu użytkownicy mogą współdzielić dane, pracować nad nimi w czasie rzeczywistym i korzystać z jednego spójnego środowiska pracy.

Ważnym aspektem SharePointa jest możliwość dostosowania wyglądu i funkcjonalności witryn do potrzeb użytkowników. Personalizacja obejmuje m.in. konfigurację interfejsu, dodawanie aplikacji webowych czy tworzenie dedykowanych formularzy.

W kontekście współczesnych modeli pracy, takich jak praca hybrydowa czy zdalna, SharePoint oferuje wsparcie dla użytkowników korzystających z różnorodnych urządzeń. Dostęp do danych jest możliwy za pośrednictwem przeglądarki internetowej oraz aplikacji mobilnych.

2.2.3 Power Automate

Power Automate to narzędzie wchodzące w skład pakietu Microsoft 365, które umożliwia automatyzację procesów biznesowych (*RPA*, ang. *Robotic Process Automation*). Pozwala ono na tworzenie przepływów pracy (ang. *flows*), automatyzujących powtarzalne zadania i integrujących różne systemy, zwiększając efektywność procesów biznesowych.

Flow w Power Automate jest odpowiednikiem funkcji w standardowych językach programowania. Na przykład, przepływ może automatycznie wysyłać powiadomienia e-mail po aktualizacji rekordu w SharePoint. Różnica polega na tym, że jest ono tworzone w wizualnym środowisku Low-Code i działa na zasadzie logicznego ciągu akcji wyzwalanych po sobie przez określone instrukcje.

Za pomocą flow można tworzyć własne procesy, które przy odpowiedniej implementacji, dorównują tym znanym z pełnych środowisk kodowych pod względem logiki i efektywności. Do dyspozycji są instrukcje warunkowe, pętle, zmienne, operacje na danych czy integracje z API poprzez konektory.

2.2.4 Power Apps

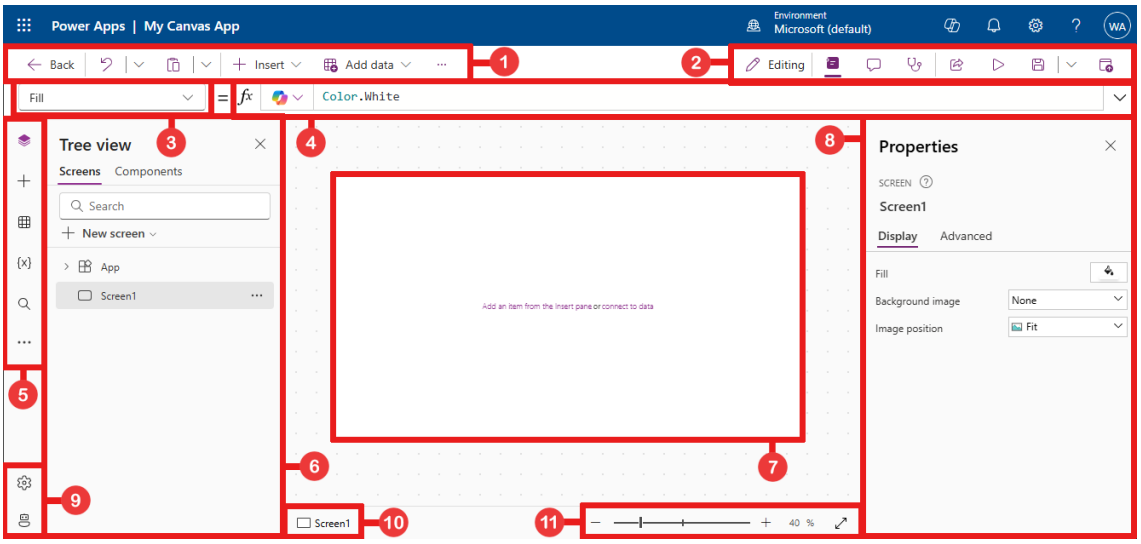
Power Apps to środowisko typu Low-Code, wchodzące w skład pakietu Microsoft 365, które jest dedykowane do tworzenia aplikacji biznesowych. Dzięki intuicyjnemu interfejsowi graficznemu umożliwia łatwą implementację mechanizmów działania, nawet osobom bez zaawansowanej wiedzy programistycznej. Jest zintegrowane z innymi usługami pakietu Microsoft 365, takimi jak SharePoint czy Power Automate, co znacznie rozszerza możliwości tworzonych aplikacji.

Power Apps pozwala na stworzenie spersonalizowanej aplikacji, dostosowanej do motywu organizacji, a przy połączeniu z innymi serwisami daje możliwość tworzenia zaawansowanych rozwiązań, minimalizując przy tym czas potrzebny na ich zaimplementowanie.

Ekrany aplikacji, komponowane za pomocą tego rozwiązania, porównywalne są z tymi, które można stworzyć w standardowych środowiskach programistycznych (jak np. JavaScript czy .NET), jednak proces ich tworzenia jest prostszy, ze względu na obecność edytora wizualnego. Umożliwia on korzystanie z gotowych komponentów w aplikacji, takich jak przyciski, pola danych wejściowych, listy, tabele, grafiki etc.

Dodawanie elementów do ekranów aplikacji odbywa się poprzez przeciąganie ich z biblioteki i upuszczanie w wybranym miejscu. Każdy komponent może zostać skonfigurowany według potrzeb użytkownika poprzez edycje *właściwości*. Możemy określić między innymi wypełnienie czy pozycję X i Y na ekranie, ale niektóre obiekty mają też unikalne właściwości takie jak *OnSelect*[3] dla przycisku.

[3] OnSelect – określa akcje, które zostaną wykonane po naciśnięciu elementu



Rysunek 2.1: Edytor Power Apps [?]
https://learn.microsoft.com/en-us/power-apps/maker/canvas-apps/power-apps-studio

Rysunek 2.1 przedstawia edytor programu. Zawiera on następujące elementy:

1. **Pasek poleceń:** wyświetla inny zestaw poleceń w zależności od wybranego kontrolki.
2. **Akcje aplikacji:** Opcje wyświetlania właściwości, dodawania komentarzy, sprawdzania błędów, udostępniania, podglądu, zapisu lub publikowania aplikacji.
3. **Lista właściwości:** Lista właściwości wybranego obiektu.
4. **Pasek formuł:** Tworzenie lub edycja formuły dla wybranej właściwości z użyciem jednej lub więcej funkcji.
5. **Menu tworzenia aplikacji:** Panel wyboru umożliwiający przełączanie się między źródłami danych oraz wstawianie dodatkowych opcji.
6. **Lista elementów aplikacji:** Pokazuje lementy obecne na ekranie w postaci drzewa.
7. **Płótno/ekran:** Główne płótno do komponowania struktury aplikacji.
8. **Panel właściwości:** Lista właściwości wybranego obiektu.
9. **Ustawienia i wirtualny agent:** Ustawienia aplikacji lub uzyskanie pomocy od wirtualnego agenta.
10. **Selektor ekranu:** Przełączanie się między różnymi ekranami w aplikacji.
11. **Zmiana rozmiaru płótna:** Zmienianie rozmiaru wyświetlanego płótna podczas tworzenia aplikacji.

Rozdział 3

Architektura rozwiązania

Niniejszy rozdział przedstawia architekturę rozwiązania, obejmującą zarówno założenia projektowe, jak i koncepcję opracowywanego systemu. Założenia projektowe określają podstawowe wymagania oraz wytyczne, stanowiąc punkt wyjścia dla opracowywanego rozwiązania. Natomiast koncepcja rozwiązania, uwzględniająca zasadnicze założenia, strukturę i logikę działania, stanowi podstawę implementacji rozwiązania.

3.1 Założenia projektowe

Założenia projektowe definiują podstawowe wytyczne dotyczące funkcjonalności i wymagań technicznych tworzonego rozwiązania. Obejmują one m.in. systematyzację danych, wykorzystanie platformy Microsoft 365 oraz optymalizację procesów decyzyjnych.

3.1.1 Systematyzacja danych

Jedną z zasadniczych funkcji omawianej aplikacji jest systematyzacja danych. Arkusze kalkulacyjne przesyłane przez oddział w Wolfsburgu nie posiadają ustandaryzowanej struktury, co negatywnie wpływa na ich czytelność oraz czas potrzebny na analizę.
Tabela 3.1 przedstawia różnice w nazwach kolumn na przestrzeni trzech lat.

Tabela 3.1: Zestawienie nagłówków kolumn w latach 2022-2024

	2022	**2023**	**2024**
Nazwy kolumn na przestrzeni lat	Service group	Service group	Service group
	Service main group	Service main group	Service main group
	Service sub group	Service sub group	Service sub group
	Business Service	Business Service	Business Service
	ID	ID	ID
	Business Service Manager	Business Service Manager	Business Service Manager
	Unit of Measurement	Unit of Measurement	Resource Unit
		Settlementtype	Settlementtype
	PL70 2022 PLAN EUR w KVA	PL71 2023 PLAN EUR w KVA	PL72 2024 PLAN EUR w KVA
	QTY	QTY	QTY
	PL71 2023 PLAN EUR w KVA	PL72 2024 PLAN EUR w KVA	PL73 2025 PLAN EUR w KVA
	QTY	QTY	QTY

Brak jednolitego formatu danych uniemożliwia również stworzenie spójnej bazy, co ogranicza możliwość ich wykorzystania w systemach automatyzacji procesów biznesowych. Dzięki wdrożeniu omawianego rozwiązania, możliwe jest ujednolicenie danych, pozwalając na ich efektywne zarządzanie i automatyczne przetwarzanie.

3.1.2 Archiwizacja danych

Utworzenie bazy danych gromadzącej informacje o wcześniejszych działaniach realizowanych w ramach projektowanego systemu stanowi istotny element zapewniający ciągłość procesów decyzyjnych. Dzięki systematycznej archiwizacji nowi użytkownicy mogą szybko zapoznać się z przebiegiem procedur i lepiej zrozumieć kontekst dotychczas podejmowanych decyzji. Dostęp do zasobów historycznych nie tylko skraca czas potrzebny na pełne wdrożenie w funkcjonowanie systemu, lecz także usprawnia przetwarzanie danych bieżących.

3.1.3 Interfejs przyjazny dla użytkownika

Dzięki dedykowanemu narzędziu z prostym i intuicyjnym interfejsem, nawigacja po bazie danych jest znacznie łatwiejsza, a problemy związane z używaniem arkuszy kalkulacyjnych zostają wyeliminowane. Przyjazny dla użytkownika interfejs oznacza:

- **Prostotę:** nieskomplikowany układ umożliwia szybkie odnalezienie potrzebnych informacji.
- **Przejrzystość:** dane są zaprezentowane w sposób czytelny, z jasno określonymi polami i etykietami.
- **Przydatne funkcje:** filtrowanie i wyszukiwanie danych wspierają efektywność pracy.

Klarowny układ i czytelność interfejsu pozwalają użytkownikowi skupić się na konkretnej usłudze, co minimalizuje ryzyko pomyłek, takich jak błędne interpretowanie danych lub wybór niewłaściwego wiersza.

Takie podejście nie tylko zwiększa efektywność pracy, ale również poprawia komfort użytkowników, dzięki czemu procesy związane z analizą i zarządzaniem danymi stają się bardziej zrozumiałe i mniej podatne na błędy.

3.1.4 Użycie pakietu Microsoft 365

Wykorzystanie platformy Power[1] w połączeniu z Sharepoint, pozwala na utworzenie w pełni funkcjonalnego rozwiązania, zachowując spójność danych dzięki integracji poszczególnych składników pakietu.

Aby korzystanie z aplikacji było możliwe, użytkownicy muszą mieć dostęp do potrzebnych usług oraz licencji. W przypadku omawianego pakietu, każdy z pracowników ma do niego dostęp, co pozwala uniknąć dodatkowych kosztów.

Wykorzystany pakiet nie jest dostępny w najbardziej rozbudowanym wariancie, co wprowadza pewne ograniczenia, ponieważ nie zawiera oprogramowania do tworzenia i zarządzania rozbudowanymi bazami danych o złożonej strukturze (takie możliwości daje między innymi *Microsoft Azure*). Sharepoint pozwala jedynie na utworzenie prostej bazy danych opierającej się o wcześniej opisane listy.

[1] Platforma Power (ang. *Power Platform*) – składowa pakietu Microsoft 365, w skład której wchodzą takie programy jak Power Apps, Power Automate czy Power BI.

3.1.5 Optymalizacja

Głównym celem implementowanego rozwiązania jest usprawnienie procesu decyzyjnego poprzez zwiększenie efektywności analizy i przetwarzania danych. Dzięki automatyzacji, czas potrzebny na podjęcie decyzji zostaje znacząco skrócony, co przekłada się na większą wydajność całego procesu.

Dzięki wprowadzeniu mechanizmów automatyzacji biurowej możliwe jest zmniejszenie liczby osób zaangażowanych w realizację procesu, co może przyczynić się do ograniczenia kosztów operacyjnych i lepszej organizacji personelu w przedsiębiorstwie.

3.2 Koncepcja rozwiązania

W niniejszym podrozdziale przedstawiona została koncepcja rozwiązania problemu automatyzacji procesu decyzyjnego. Na podstawie przeprowadzonej analizy wymagań oraz istniejących ograniczeń, zaproponowano kompleksowe podejście do realizacji systemu.

Całość rozwiązania podzielono na cztery główne etapy:

- utworzenie dedykowanej bazy danych,
- opracowanie mechanizmu importu danych z arkuszy kalkulacyjnych,
- przygotowanie formularzy do obsługi procesu,
- automatyzacja generowania raportów.

Przyjęte rozwiązanie ma na celu usprawnienie procesu przy zachowaniu jego dotychczasowej logiki biznesowej. Szczegółowy opis realizacji poszczególnych etapów został przedstawiony w kolejnych podrozdziałach.

3.2.1 Baza danych

Do przechowywania danych wykorzystano listy programu SharePoint. Pomimo tego, że nie jest to dedykowane rozwiązanie bazodanowe, wybór ten wynika z wymogu integracji z istniejącą infrastrukturą.

Struktura bazy danych

W wyniku analizy danych historycznych zidentyfikowano elementy kluczowe dla procesu indykacji. Na tej podstawie zaprojektowano strukturę składającą się z trzech powiązanych ze sobą list:

- **Lista usług** – zawierająca podstawowe, niezmienne informacje o serwisach,
- **Lista kwot** – przechowująca dane odnośnie cen i liczbie licencji, które zmieniają się raz do roku,
- **Lista indykacji** – gromadząca informacje w obrębie jednej indykacji.

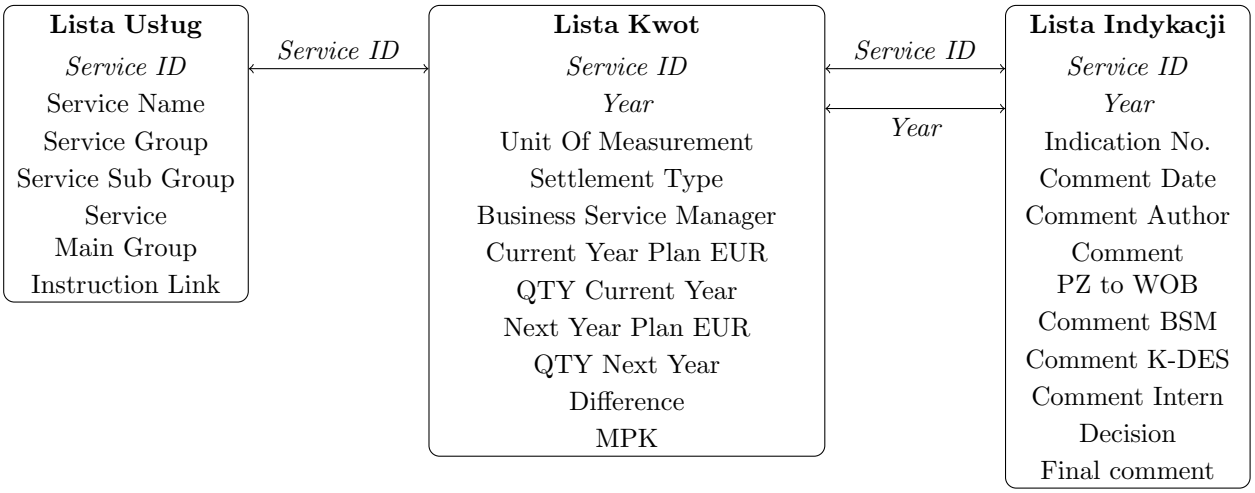
Atrybuty danych

Na podstawie analizy wymagań oraz dotychczasowego procesu, zdefiniowano następujący zestaw atrybutów, które powinna zawierać baza danych:

- Service group
- Service main group
- Service sub group
- Business Service
- Instruction link
- ID
- Business Service Manager
- Unit Of Measurement
- Settlement Type
- Current Year Plan EUR
- Quantity Current Year
- Next Year Plan EUR
- Quantity Next Year
- Year
- MPK
- Difference
- Indication Number
- Comment Intern
- Comment Date
- Comment Author
- Comment PZ to WOB
- Comment BSM
- Comment K-DES
- Decision
- Final comment

Powyższy zestaw atrybutów został opracowany na podstawie analizy danych historycznych z poprzednich lat (przedstawionych w Tabeli 3.1). Wybrane pola reprezentują najczęściej występujące informacje w procesie indykacji, uzupełnione o dodatkowe atrybuty niezbędne do efektywnego funkcjonowania procesu, takie jak pola komentarzy czy decyzji.

Model powiązań



Rysunek 3.1: Schemat relacji między listami.

Model danych przedstawiony na Rysunku 3.1 został zaprojektowany z uwzględnieniem następujących założeń:

- *Lista usług* pełni rolę centralnego rejestru serwisów, zawierając ich podstawową charakterystykę,
- *Lista kwot* umożliwia śledzenie zmian w wymiarze finansowym na przestrzeni lat,
- *Lista indykacji* przechowuje historię procesu decyzyjnego wraz z towarzyszącymi komentarzami i ustaleniami.

3.2.2 Dodawanie informacji do bazy danych

Po ustaleniu struktury danych wykorzystywanych przez system, kolejnym etapem jest określenie sposobu importu informacji z arkuszy kalkulacyjnych do bazy danych. Postanowiono wykorzystać

program Power Automate w celu automatyzacji tego procesu, jednakże z uwagi na nieschematyczność danych wymaga on asysty użytkownika.

W celu dostosowania danych do struktury bazy oraz ich importu zaplanowano zaimplementowanie formularza, który pozwoli na:

1. Walidację nazw kolumn w arkuszu kalkulacyjnym:
    - System pobiera nazwy istniejących kolumn z arkusza.
    - Użytkownik mapuje nazwy kolumn z predefiniowaną listą nagłówków z list SharePoint.
2. Określenie roku oraz numeru indykacji dla importowanego arkusza.
3. Przekazanie danych do *flow* w programie *Power Automate*, który przypisze je do odpowiednich list w bazie danych, jednocześnie zapobiegając duplikacji rekordów.

Interfejs jest dodatkowo wyposażony w formularz służący do przypisywania numerów *MPK* nowym serwisom, ze względu na to, iż numer *MPK* determinuje obszar odpowiedzialny za obsługę danej usługi. Dla serwisów występujących w poprzednich latach, system automatycznie przypisuje istniejące numery *MPK*, redukując ilość danych do wprowadzenia. Jednocześnie zachowana będzie możliwość modyfikacji wcześniej przypisanych numerów.

3.2.3 Interfejs procesu decyzyjnego

Interfejs obsługi procesu decyzyjnego został podzielony na dwa współpracujące ze sobą ekrany. Takie rozwiązanie pozwala na zachowanie przejrzystości prezentowanych informacji przy jednoczesnym zapewnieniu dostępu do wszystkich niezbędnych funkcjonalności.

Ekran nawigacyjny

Pierwszy ekran pełni rolę panelu nawigacyjnego, prezentując najważniejsze informacje dotyczące usługi:

- Service Name – nazwa serwisu,
- Service ID – unikalny identyfikator usługi,
- MPK – numer określający miejsce powstawania kosztów,
- Decision – aktualny status decyzji.

Ponadto, użytkownicy będą mieli możliwość filtrowania i wyszukiwania serwisów według następujących kryteriów:

- wyszukiwanie serwisów względem ID,
- wyszukiwanie serwisów względem nazwy,
- filtrowanie według przypisanych numerów MPK,
- filtrowanie według statusu decyzji (*Accepted*, *Not Accepted*, *No Status*).

Ekran szczegółowy

Po wybraniu serwisu z listy, użytkownik zostaje przekierowany do ekranu szczegółowego, który składa się z trzech głównych sekcji:

- **Podgląd danych historycznych** – prezentuje on zarówno ogólne informacje o serwisie zbierane na przestrzeni lat, jak i szczegóły dotyczące poszczególnych indykacji.
- **Formularz decyzyjny** – zestaw pól do wprowadzenia informacji o bieżącej indykacji. Składają się na niego:
  - rok (wartość domyślna: bieżący),
  - numer indykacji (wartość domyślna: kolejny wolny numer),
  - autor (wartość domyślna: zalogowany użytkownik),
  - komentarze (wewnętrzny, BSM, K-DES),
  - decyzja (wartość domyślna: poprzednia decyzja).

3.2.4 Generowanie raportu

Ostatnim etapem cyklu obsługi procesu jest generowanie raportu, który jest następnie przesyłany do zakładu w Wolfsburgu w celu dalszych konsultacji. Raport jest tworzony na podstawie danych przechowywanych na listach SharePoint, co zapewnia spójność i aktualność informacji.

W dedykowanym oknie aplikacji użytkownik ma możliwość wyboru odpowiedniego roku oraz etapu (numer indykacji). Na podstawie tych informacji, system przetworzy podane kryteria, aby zgromadzić odpowiednie dane z różnych źródeł.

Kolekcja[2] danych, utworzona na podstawie wybranych kryteriów, łączy dane z trzech list sharepointowych. Dzięki temu możliwe jest skonsolidowanie danych w jedną, spójną strukturę, która zawierać będzie wszystkie niezbędne informacje do sporządzenia raportu. **TO TRZEBA ROZWINĄĆ W IMPLEMENTACJI BO MOGĄ SIĘ POJAWIĆ JAKIE MECHANIZMY (JA NA PRZYKŁAD NIE WIEM XD):** Mechanizmy wyszukiwania umożliwią powiązanie identyfikatorów usług z odpowiadającymi im rekordami, co zapewni integralność zgromadzonych danych.

Dodatkowo, w tym samym oknie aplikacji użytkownik ma dostęp do podglądu zgromadzonych danych w formie tabeli. Umożliwi to weryfikację poprawności i kompletności informacji przed wygenerowaniem raportu. Po zatwierdzeniu danych, system przekaże zgromadzoną kolekcję do Power Automate, gdzie zostaną one zmodyfikowane, w celu do przygotowania raportu w odpowiednim formacie (tj. w formacie arkusza kalkulacyjnego *Excel*) do odesłania.

[2]Kolekcja (Power Apps) – tymczasowy zbiór danych, przechowywanych lokalnie w aplikacji, umożliwiający zarządzanie rekordami podczas jej działania.

Rozdział 4

Implementacja

W niniejszym rozdziale przedstawiono szczegóły techniczne wdrożonego rozwiązania oraz przybliżono aspekty implementacyjne systemu, obejmujące wykorzystanie platformy Microsoft Power Platform – w szczególności Power Apps – do budowy interfejsu użytkownika oraz Power Automate do automatyzacji procesów biznesowych. Ponadto, omówiono integrację z platformą SharePoint oraz implementację skryptów usprawniających pracę z pakietem Microsoft Office.

W ramach analizy technicznej szczegółowo opisano wszystkie komponenty systemu oraz sposób ich integracji. Szczególną uwagę poświęcono mechanizmom przepływu danych, automatyzacji procesów oraz implementacji logiki biznesowej w środowisku Low-Code.

4.1 Utworzenie bazy danych na platformie Sharepoint

Zgodnie z koncepcją, baza danych utworzona została w środowisku SharePoint. Pierwszym krokiem jest stworzenie strony dedykowanej temu procesowi. W ekranie startowym programu należy wybrać opcję *Utwórz witrynę*. Następnie należy wybrać typ witryny *Witryna zespołu* oraz szablon określający jej wygląd. Na koniec należy określić nazwę tworzonej strony.

Kolejnym krokiem jest utworzenie struktury list oraz plików. Na potrzeby procesu utworzono dwa foldery – *TempFiles* przechowujący arkusze przed zapisaniem ich danych na listach oraz *ArchivedFiles* przechowujący pliki po zakończeniu procesu. Ponadto utworzono trzy listy:

- *Lista_Uslug*,
- *Lista_Kwot*,
- *Lista_Indykacji*.

Ich struktura jest taka sama jak opisana w podsekcji 3.2.1. W celu dodania tych elementów należy:

- wejść w utworzoną witrynę,
- wybrać *Zawartość witryny* z bocznego paska nawigacji,
- dodać element przy pomocy przycisku *Nowy*,
- wybrać typ elementu – *Biblioteka dokumentów* lub *Lista*,
- podać nazwę oraz opcjonalnie opis elementu.

Po wykonaniu tych kroków, folder jest gotowy do użycia. Jednakże w przypadku list należy jeszcze zdefiniować kolumny.
Aby to zrobić należy:

- wejść w utworzoną listę,
- wybrać przycisk *Dodaj kolumnę* znajdujący się po prawej stronie domyślnie utworzonych kolumn,
- następnie pojawi się okno z możliwością wyboru typu kolumny oraz jej nazwy,
- po zatwierdzeniu podanych opcji, kolumna zostanie dodana do listy.

Na koniec należy skonfigurować niektóre kolumny. Aby to zrobić należy wybrać ustawienia strony i nacisnąć pozycję *Ustawienia listy*. Tam wybieramy nazwę kolumny, którą chcemy skofigurować.
Tabela 4.1 przedstawia konfigurację kolumn listy:

Tabela 4.1: Konfiguracja listy na platformie Sharepoint

Nazwa kolumny	**Typ**
Service_ID	Liczba
Service_Name	Pojedynczy wiersz tekstu
Service_Group	Pojedynczy wiersz tekstu
Service_Sub_Group	Pojedynczy wiersz tekstu
Service_Main_Group	Pojedynczy wiersz tekstu
Instruction_Link	Pojedynczy wiersz tekstu
Unit_Of_Measurement	Pojedynczy wiersz tekstu
Settlement_Type	Pojedynczy wiersz tekstu
Business_Service_Manager	Pojedynczy wiersz tekstu
Current_Year_Plan_EUR	Liczba
QTY_Current_Year	Liczba
Next_Year_Plan_EUR	Liczba
QTY_Next_Year	Liczba
Difference	Obliczeniowa
MPK	Pojedynczy wiersz tekstu
Year	Liczba
IndicationNo	Liczba
Comment_Date	Pojedynczy wiersz tekstu
Comment_Author	Wiele wierszy tekstu
Comment_PZ_to_WOB	Wiele wierszy tekstu
Comment_BSM	Wiele wierszy tekstu
Comment_K-DES	Wiele wierszy tekstu
Comment_Intern	Wiele wierszy tekstu
Decision	Liczba
Final_comment	Wiele wierszy tekstu

Kolumna *Difference* jest kolumną obliczeniową co oznacza, że jej wartość jest obliczana względem podanej formuły. W tym przypadku oblicza ona różnicę cen między rokiem następnym a bieżącym.

Kolumna *MPK* pomimo wartości liczbowych, jest typu tekstowego gdyż znacznie upraszcza to filtrowanie oraz przepisywanie istniejących numerów do nowych danych.
Kolumna *Decision* jest kolumną liczbową ponieważ status decyzji konwertowany jest na kod liczbowy aby przyspieszyć proces filtrowania:

- *Accepted* → 1,
- *Not Accepted* → -1,
- *No Status* → 0.

4.2 Ekran dodawania danych

Głównym wyzwaniem okazał się brak systematycznej organizacji danych w arkuszach programu Excel, co skutkowało niekompatybilnością z zaprojektowaną bazą danych. W celu rozwiązania tego problemu, opracowano dedykowany interfejs w aplikacji, który wspomaga użytkownika w procesie przetwarzania danych, minimalizując ryzyko wystąpienia błędów. Składa się on z:

- sekcji pozwalającej użytkownikowi na zapisanie pliku w chmurze,
- formularza walidacji nazw kolumn,
- kontrolek umożliwiających wybór roku i numeru indykacji oraz przycisku do przesłania danych do bazy,
- formularza do uzupełniania lub edycji numerów MPK.

4.2.1 Zapis pliku w chmurze

Pierwszym etapem procesu jest tymczasowy zapis pliku Excel w chmurze, co umożliwia jego udostępnienie innym systemom. Do realizacji tego zadania wykorzystano kontrolkę[1] *Attachment Control*. Pozwala ona na zapisanie pliku w pamięci aplikacji. Odbywa się to przez naciśnięcie przycisku *"Attach file"* lub przy użyciu mechaniki *przeciągnij i upuść* (ang. *Drag And Drop*).

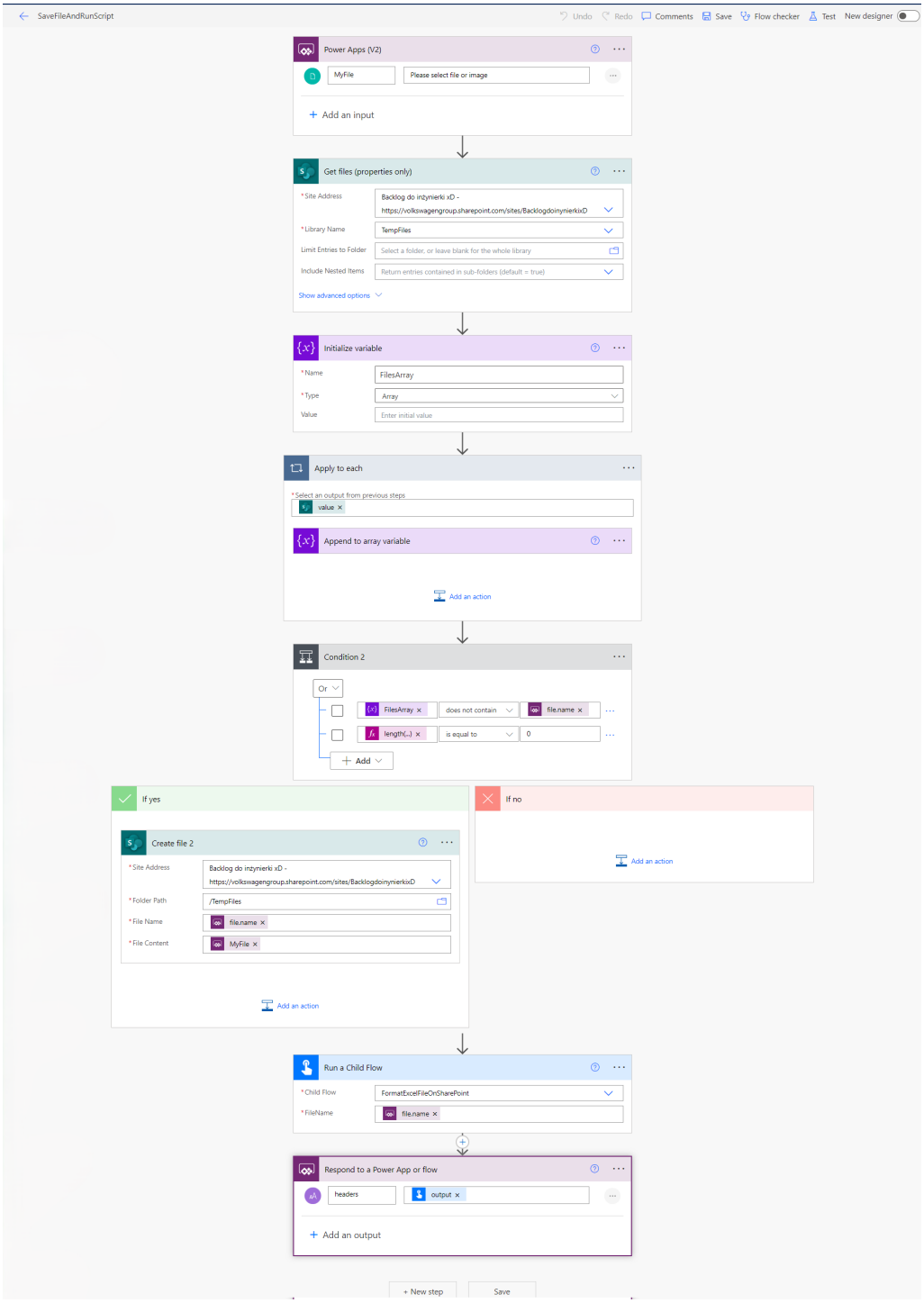
Pod kontrolką *Attachment Control* dodano przycisk *Save attachment*. Jego naciśnięcie skutkuje wywołaniem szeregu funkcji opisanych we właściwości *OnSelect*:

- spawdzenie czy plik został poprawnie załadowany,
- wywołanie przepływu *SaveFileAndRunScript*,
- zapisanie wyniku przepływu w zmiennej tablicowej *FlowOutput* (w Power Apps określanej jako *kolekcja*),
- po poprawnym zapisaniu pliku w folderze SharePoint, usunięcie go z pamięci aplikacji.

Przepływ SaveFileAndRunScript

Rysunek 4.1, przedstawia edytor programu Power Automate. Widoczny w nim przepływ nazwany *SaveFileAndRunScript* jest odpowiedzialny za zapisanie pliku w chmurze oraz wstępne przetworzenie. W momencie wywołania przepływu, plik jest przekazany jako parametr wejściowy. Przepływ ten składa się z kilku kroków, które zostaną omówione w kolejności ich wykonywania.

[1] Kontrolka – element służący do nawigacji, wyświetlania danych i obsługi aplikacji.



Rysunek 4.1: Widok przepływu SaveFileAndRunScript

1. **Funkcja: Power Apps (V2)**
   Przepływ rozpoczyna się od funkcji oczekującej na wywołanie przepływu bezpośrednio z aplikacji Power Apps. Jako parametry wejściowe przyjmuje:

   - nazwę pliku (*File Name*),
   - zawartość pliku (*File Content*) w formacie binarnym.

2. **Zainicjowanie zmiennej**
   Element *Initialize variable* tworzy zmienną o nazwie *FileExists*, która przechowuje informa-

cję, czy plik o podanej nazwie znajduje się już na SharePoint.

3. **Sprawdzenie istniejących plików**
   Blok *Get files* pobiera listę wszystkich plików z wybranego folderu SharePoint wraz z ich metadanymi, takimi jak nazwa, ścieżka czy data modyfikacji. Wynik zostaje zapisany w zmiennej *FileExists*, która przyjmuje wartość *true*, jeśli plik został znaleziony, lub *false*, jeśli plik nie istnieje.

4. **Instrukcja warunkowa *If***
   Element *Condition* sprawdza wartość zmiennej *FileExists*. W zależności od wyniku:

   - jeśli zmienna ma wartość *true* – przepływ kończy działanie,
   - jeśli zmienna ma wartość *false* – przepływ kontynuuje proces zapisu.

5. **Utworzenie pliku**
   Blok *Create file* tworzy nowy plik w SharePoint, wykorzystując parametry:

   - adres witryny SharePoint,
   - ścieżkę do folderu docelowego,
   - nazwę pliku,
   - zawartość pliku.

6. **Uruchomienie przepływu podrzędnego**
   Po pomyślnym zapisaniu pliku przepływ wywołuje tzw. *child flow*, który inicjuje działanie skryptu Office. Skrypt ten odpowiada za przetworzenie pliku w sposób zgodny z założeniami aplikacji. Jego wynik w formacie JSON jest zwracany do strumienia nadrzędnego.

7. **Odpowiedź do aplikacji**
   Blok *Respond to Power Apps* kończy przepływ, zwracając do aplikacji dane w formacie JSON, przetworzone przez wspomniany skrypt.

4.2.2 Skrypt pakietu Office

Po utworzeniu pliku w SharePoint, w ramach przepływu następuje jego przetworzenie przez skrypt. Jest to niezbędne, jeśli chodzi o działanie procesu. Domyślnie otrzymane dane w pliku Excel są niewidoczne dla większości systemów, mogą one odczytać jedynie informacje zorganizowane w *tabele programu Excel*[2]. Dlatego też powstał skrypt, który działa bezpośrednio w arkuszu. Jego zadaniem jest automatyczne utworzenie tabeli oraz dostosowanie jej do wymagań systemu. Poniżej przedstawiono kroki działania skryptu:

1. **Wybór arkusza roboczego**
   Skrypt identyfikuje arkusz zawierający dane, analizuje zakres używanych komórek i usuwa ochronę hasłem, jeśli jest aktywna – krok ten jest wymagany, aby wprowadzanie zmian w arkuszu było możliwe.

2. **Analiza danych**
   Skrypt rozpoczyna analizę od wyszukiwania początku tabeli w arkuszu. Następnie:

   - usuwa puste kolumny, które nie zawierają żadnych danych,

[2]Tabela w programie Excel wymaga osobnej deklaracji poprzez zaznaczenie zakresu komórek i wybór opcji *Narzędzia główne→Formatuj jako tabelę*

- tworzy tabelę o dynamicznym rozmiarze, uwzględniając zakres danych znajdujących się w arkuszu,
- uzupełnia brakujące komórki w kluczowych kolumnach, korzystając z danych w poprzednich wierszach.

Takie podejście pozwala na uporządkowanie danych i przygotowanie ich do dalszego przetwarzania.

3. **Dopasowanie nazw kolumn**
   Skrypt porównuje istniejące nazwy kolumn z listą standardowych nagłówków, korzystając z algorytmu *Jaro-Winkler*. Algorytm ten:
   - analizuje podobieństwo tekstów, porównując wspólne znaki oraz ich kolejność,
   - przyznaje dodatkowe punkty za zgodność początkowych znaków (prefiksu),
   - zwraca wynik jako wartość z przedziału od 0 do 1, gdzie wartości bliższe 1 oznaczają większe podobieństwo.

   Wynik tego procesu jest wykorzystywany w dalszych etapach aplikacji, m.in. do walidacji struktury danych. Jeśli podobieństwo jest mniejsze niż 90%, skrypt sugeruje ręczne dopasowanie nazwy kolumny.

4. **Zwrócenie wyników**
   Skrypt generuje JSON zawierający mapowanie oryginalnych nazw kolumn z najlepszymi dopasowaniami z listy standardowych nagłówków.

Kolejnym elementem tej sekcji ekranu jest lista zapisanych plików znajdująca się obok kontrolki *Attach Control*. Umożliwia ona wybór pliku do dalszego przetwarzania. Poniżej umieszczono przycisk *"Click to open:..."*, który pozwala na otwarcie wybranego pliku w nowym oknie przeglądarki przy użyciu funkcji *Launch*. Możliwość podglądu ma na celu ułatwienie weryfikacji jego poprawności.

Algorytm Jaro-Winkler

Podstawą algorytmu, jest algorytm *Jaro*. Polega on na porównaniu dwóch ciągów znaków w celu określenia ich podobieństwa. Wynik obliczany jest na podstawie równania 4.1:

$$J = \frac{1}{3}\left(\frac{m}{|s_1|} + \frac{m}{|s_2|} + \frac{m-t}{m}\right) \text{ dla } m > 0 \tag{4.1}$$

gdzie:
m – liczba dopasowanych znaków,
t – liczba transpozycji,
$|s_1|$, $|s_2|$ – długości ciągów.

Wynikiem równania, jest wartość z przedziału od 0 do 1, gdzie 1 oznacza pełne dopasowanie.

Algorytm *Jaro-Winkler* jest rozszerzeniem algorytmu *Jaro*, które dodaje premię za zgodność początkowych znaków. Wynik obliczany jest na podstawie równania 4.2:

$$JW = J + l \cdot p \cdot (1 - J), \tag{4.2}$$

gdzie:
l – długość wspólnego prefiksu (maksymalnie 4 znaki),

p – współczynnik skalowania.

Rozszerzona wersja algorytmu, pozwala na bardziej precyzyjne porównanie ciągów znaków, które mają wspólny prefix, co jest szczególnie ważne w przypadku nagłówków kolumn rozpoczynających się od frazy *Service*.

LINK DO TEGO JARO_WINKLERA: https://crucialbits.com/blog/a-comprehensive-list-of-similarity-search-algorithms/ https://en.wikipedia.org/wiki/Jaro%E2%80%93Winkler_distance

4.2.3 Walidacja nazw kolumn

Kolejnym etapem przed zapisaniem danych do bazy jest walidacja nazw kolumn. W tym celu zaimplementowano formularz zawierający *galerię* – element umożliwiający wyświetlanie wielu rekordów danych o różnych typach. Pola wyświetlające dane w galerii mogą być dostosowywane w dowolny sposób w zależności od potrzeb użytkownika.
Galeria składa się z dwóch kolumn:

- Lewa kolumna prezentuje obecne nazwy kolumn, które są wyświetlane za pomocą kontrolki *Label*[3].
- Prawa kolumna zawiera kontrolkę *ComboBox*[4], która umożliwia wybór nazwy ze standardowej listy nagłówków. Wartości domyślne, widoczne w kontrolkach *ComboBox*, są generowane przez wcześniej opisany skrypt w taki sposób, aby do oryginalnej nazwy kolumny dopasowana została najbardziej podobna nazwa z predefiniowanej listy nagłówków. Ma to na celu minimalizację danych wprowadzonych przez użytkownika.

Po prawej stronie formularza umieszczono instrukcję użytkownika, zawierająca wskazówki dotyczące prawidłowego uzupełniania nazw kolumn. Poniżej instrukcji dodano przycisk *Update column names*, który umożliwia zapisanie zmian w strukturze danych.
Działanie tego mechanizmu opiera się na zastosowaniu skryptu pakietu Office, wywoływanego przy użyciu kolejnego przepływu. Skrypt jako parametr wejściowy przyjmuje zmienną tablicową w formacie JSON, zawierającą mapowanie oryginalnych nazw kolumn z poprawionymi wartościami wybranymi przez użytkownika. Następnie skrypt iteruje po wierszu zawierającym nagłówki kolumn i dokonuje ich zamiany zgodnie z otrzymaną mapą. Po zakończeniu działania skrypt zwraca nową strukturę nazw kolumn.

Pod przyciskiem *Update column names* umieszczone zostały kontrolki *Dropdown*[5] oraz przycisk *Upload data*, które są kluczowe dla kolejnego etapu przetwarzania danych, obejmującego ich integrację z listami SharePoint.

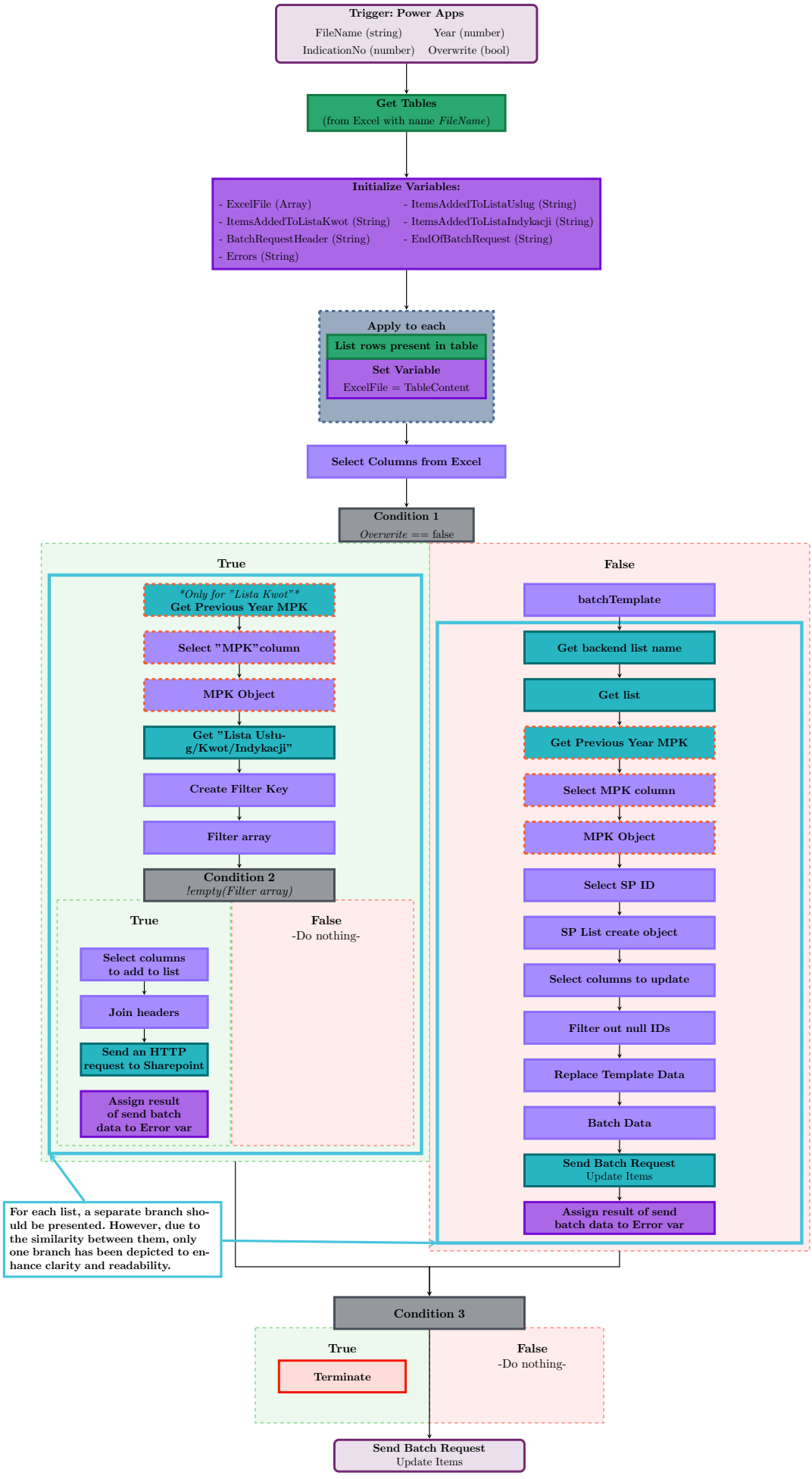
4.2.4 Integracja z listami SharePoint

Po zakończeniu walidacji nazw kolumn, kolejnym etapem jest zapisanie przetworzonych danych w utworzonej strukturze SharePoint. Proces ten rozpoczyna się od wyboru roku i numeru indykacji przy użyciu dedykowanych kontrolek *Dropdown*. Wybrane wartości są następnie wykorzystywane podczas importu danych do odpowiednich list, co odbywa się za pomocą przycisku *Upload data*.

[3] *Label* – kontrolka tekstowa umożliwiająca wyświetlanie statycznych wartości.
[4] *ComboBox* – rozwijana lista z możliwością wprowadzania tekstu
[5] *Dropdown* – kontrolka umożliwiająca wybór jednej z dostępnych wartości z rozwijanej listy, bez możliwości edycji.

Skutki kliknięcia przycisku mogą się różnić w zależności od wybranych wartości i tego czy nazwy kolumny zostały zmienione. Jeśli przed próbą wgrania danych, nie został wciśnięty przycisk *Update column names*, system wyświetla okno z zapytaniem o poprawność nazw kolumn w celu upewnienia się, że użytkownik nie wgra przypadkowo danych z niepoprawnymi nagłówkami. Kiedy jednak nazwy kolumn zostały zmienione, ale użytkownik wybrał rok oraz numer indykacji, istniejące w bazie danych, system wyświetla zapytanie czy użytkownik chce nadpisać dane, które już tam się znajdują czy też anulować operacje. W momencie kiedy nazwy kolumn nie zostaną zmienione oraz dane z wybranym rokiem i numerem indykacji istnieją w bazie danych, pojawiają się oba okna z informacjami.

Kiedy użytkownik upewni się, że wszystkie dane są poprawne i zatwierdzi operację, system przystępuje do importu danych. W tym celu wywołuje kolejny przepływ w programie Power Automate, który przypisuje informacji do odpowiednich list w bazie danych upewniając się jednocześnie, że nie zostaną dodane duplikaty rekordów.
Przepływ ten jest bardzo rozbudowany, dlatego zamiast widoku edytora Power Automate, na rysunku 4.2 pokazany został schemat blokowy, który reprezentuje kolejność wykonywania poszczególnych kroków. Został on uproszczony, ponieważ bloku umieszczone w błękitnych ramkach, powinny być powielone do trzech równoległych gałęzi. Wynika to z faktu, że dla wszystkich list procedura jest identyczna, a różnicą są m.in. nazwy list użyte strukturze żądań HTTP. Ponadto elementy, które mają przerywaną, pomarańczową ramkę to elementy, które znajdują się w głęzi dotyczącej listy kwot. Odpowiadają one za pobieranie i przepisywanie numeru MPK z roku wcześniej.



Rysunek 4.2: Schemat blokowy procesu importu danych z arkusza kalkulacyjnego do bazy danych

Poniżej wyjaśniono działanie poszczególnych bloków znajdujących się na schemacie. Elementy oznaczone symbolem (∗) odnoszą się do bloków znajdujących się w pomarańczowej, przerywanej ramce.

1. Blok **Trigger: Power Apps** jest wyzwalaczem działania przepływu. Uruchamia się kiedy użytkownik wciśnie przycisk w aplikacji. Jako parametry wejściowe przyjmuje nazwę pliku, rok i numer indykacji jakie mają być przypsiane do danych oraz informacje czy nadpisać istniejące rekordy czy też nie.

2. **Get Tables** pobiera nazwy wszystkich tabel z pliku o nazwie przekazanej do wyzwalacza (każdy plik powinien zawierać jedną tabelę, ale w Power Automate trzeba pobrać wszystkie możliwe).

3. Funkcja **Initialize variable** tworzy następujące zmienne:

   - *ExcelFile* – zmienna tablicowa, przechowująca dane z arkusza,
   - *ItemsAddedToListaUsług/Kwot/Indykacji* – te zmienne przechowują *ciało żądania HTTP*, które jest konstruowane w trakcie działania przepływu.
   - *BatchRequestHeader* oraz *EndOfBatchRequest* – przechowują one stałe nagłówki i stopkę żądania HTTP, które są wspólne dla wszystkich zapytań.

4. Pętla **Apply to each**, dodana automatycznie przez Power Automate, iteruje po nazwach tabel pobranych z pliku a następnie dla każdej z nich przepisuje dane do zmiennej *ExcelFile*.

5. Funkcja **Select Columns from Excel**, pozwala na kształtowanie danych. Jako wejście przyjmuje ona dane z *ExcelFile* a następnie mapuje wartości tej zmiennej do wybranych kluczy. Dzięki temu można odwołać się do dowolnej kolumny danych.

6. Blok **Condition 1** sprawdza czy wartość *Overwrite* jest równa *false*.
   Jeśli tak to wykonywane są instrukcje wewnątrz bloku *True*. Gałąź ta odpowiada za dopisanie nowych danych do bazy. W tym celu wykonane są następujące instrukcje:

   ∗ **Get Previous Year MPK** – Pobranie elementów z listy kwot dla roku niżej niż przekazany w parametrze wejściowym,

   ∗ **Select "MPK"column** jako wejście przyjmuje odpowiedź z bloku wyżej, ale tutaj zamiast przypisywania wartości do klucza, zawiera następujące wyrażenie:

   ```
   concat("",item()?['Service_ID'],'":',item())
   ```

   item() odwołuje się do pojedynczego elementu danych wejściowych. Zatem to wyrażenie tworzy strukturę obiektów, gdzie nazwą obiektu jest *Service_ID*, natomiast jako właściwości obiektu przypisane są dane z arkusza odpowiadające tej usłudze.

   ∗ **MPK Object** przekształca strukturę utworzoną w poprzednim kroku na listę obiektów JSON.

   - **Get "Lista Usług/Kwot/Indykacji"** – pobiera pełną listę rekordów z odpowiedniej listy w bazie danych.
   - **Create Filter Key** tworzy klucz filtrujący. Dla listy usług nie jest on wymagany. Dla listy kwot kluczem jest rok przekazany w parametrze wejściowym. Natomiast dla listy indykacji jest to rok oraz numer indykacji połączone w jeden ciąg znaków.

- **Filter array** blok ten wykorzystany jest do porównania elementów dla danego roku i indykacji na liście SharePoint z elementami w arkuszu. Ma on za zadanie zwrócić tablicę z elementami unikalnymi dla arkusza.
- **Condition 2** sprawdza czy tablica zwrócona przez *Filter array* nie jest pusta. Jeśli nie zawiera ona unikatowych elementów to przepływ kończy działanie. Jeśli natomiast tablica zawiera unikatowe elementy to przepływ przechodzi do kolejnego kroku w gałęzi *True*.
- **Select columns to add to list** – mapuje informacje z arkusza do kluczy odpowiadających strukturze każdej z list w bazie danych.
- **Join headers** – konwertuje tablicę powstałą w poprzednim kroku na ciąg znaków, będący ciałem żądania HTTP. Instrukcja ta zmienia separator między wierszami tabeli ze średnika na nagłówek, który musi znajdować się między każdym wysłanym zestawem danych. Dla każdej z list jest on inny.
- **Send an HTTP request to SharePoint** – wysyła kompletne żądanie HTTP do odpowiedniej listy w bazie danych. Wysyłane żądanie zawiera:
  - Nagłówek otwierający żądanie – *BatchRequestHeader*,
  - Ciało żądania powstałe w kroku wcześniej – wynik *Join headers*,
  - Stopkę żądania – *EndOfBatchRequest*.
- **Assign result of send batch data to Error var** – jak nazwa bloku wskazuje, przypisuje odpowiedź serwera na wysłane żądanie w celu późniejszej analizy.

Kiedy jednak wartość zmiennej *Overwrite* wynosi *True*, oznacza to, że istniejące rekordy mają zostać zaaktualizowane. Przeplyw przechodzi do gałęzi *False* i wykonuje następujące kroki:

- **batchTemplate** – tworzy wspólny szablon żądania HTTP.
- **Get backend list name** pobiera ona wewnętrzną nazwę listy SharePoint. Jest to konieczne, ponieważ w żądaniu HTTP należy ostrożnie używać znaków specjalnych.
- **Get list** odczytuje dane z każdej z list.

* Funkcje **Get Previous Year MPK**, **Select "MPK"column** oraz **MPK Object** wykonują te same zadania co w wcześniej omawianym scenariuszu.

- Kroki **Select SP ID** i **SP List create object** działają podobnie jak mechanizm pobierania numerów MPK z roku poprzedniego z tym, że mapują one identyfikatory wewnętrzne elementów listy SharePoint. Jest to konieczne ponieważ aby zaktualizować rekord, należy odwołać się do niego względem tego właśnie identyfikatora a nie np. nazwy lub *Service_ID*.
- **Select columns to update** – przydziela informacje z odpowiednich kolumn do odpowiednich list.
- **Filter out null IDs** – odfiltrowuje elementy, które nie mają przypisanego identyfikatora SharePoint. Gdyby nie ten krok, próba aktualizacji rekordu bez identyfikatora zakończyłaby się błędem.
- **Replace Template Data** – wstawia wybrane informacje do szablonu żądania HTTP.

- **batchData** – w kroku tym, znaki specjalne są zakodowane procentowo[6] (znane również jako *kodowanie URL*). Jest to wymagane aby uniknąć błędów.
- **Send Batch Request** – wysyła żądanie aktualizacji danych do SharePoint.
- **Assign result of send batch data to Error var** – przypisuje odpowiedź serwera na wysłane żądanie w celu późniejszej analizy.

7. **Condition 3** sprawdza czy zmienna *Errors* zawiera jakiekolwiek kody błędów. Jeśli tak to przepływ zostaje przerwany. W przeciwnym wypadku zwracana jest informacja do aplikacji, że zapis danych zakończył się powodzeniem.

4.2.5 Uzupełnianie numerów MPK

Ostatnią funkcją tego ekranu jest możliwość uzupełniania lub edycji numerów MPK. W tym celu ponownie wykorzystano galerię, która tym razem składa się z trzech kolumn. Dwie pierwsze kolumny zawierają pola tekstowe (*Label*), które prezentują nazwę usługi oraz odpowiadający jej identyfikator. Ostatnia kolumna zawiera pole danych wejściowych (*TextInput*), do którego użytkownik wprowadza odpowiedni numer MPK. Nad galerią znajduje się dodatkowe pole, w którym można wpisać kryteria filtrowania, takie jak nazwa, identyfikator bądź numer MPK. Obok pól filtracji, znajduje się przełącznik (*Toggle*), który umożliwia wyświetlenie usług z przypisanym już numerem MPK. Istniejące numery MPK wyświetlają się jako domyślny tekst kontrolki *TextInput* i mogą być edytowane. Ostatnim elementem ekranu jest pole tekstowe informujące użytkownika o liczbie usług bez przypisanego miejsca powstawania kosztów.

4.3 Ekran startowy aplikacji i przygotowanie danych

Kiedy dane zostały dostosowane do działania systemu, przystąpiono do implementacji pozostałych części rozwiązania. Kolejnym elementem jest ekran startowy aplikacji, zawierający przyciski, które przekierowują użytkownika do odpowiednich sekcji aplikacji.

Dodatkowo, podczas uruchomienia aplikacji, pobierane są dane z list SharePoint a następnie odpowiednio przetwarzane w celu płynnego wyświetlania ich w aplikacji.
Kod w języku *Power Fx* wywoływany podczas uruchamiania aplikacji został przedstawiony w listingu 4.1.

```
Set(varDownloadingData;  true );;
ClearCollect(colYears;
    {Value: Text(Now(); "yyyy") - 2};
    {Value: Text(Now(); "yyyy") - 1};
    {Value: Text(Now(); "yyyy") + 0};
    {Value: Text(Now(); "yyyy") + 1};
    {Value: Text(Now(); "yyyy") + 2}
);;
Set(VWBlue; ColorValue("#002e5f"));;
ClearCollect(colNumbers;
    {Value: 1};
    {Value: 2};
    {Value: 3};
```

[6] *Kodowanie procentowe* – metoda reprezentowania znaków specjalnych w adresach URL w formie zgodnej z protokołem HTTP. Polega na zastępowaniu niebezpiecznych lub niedozwolonych znaków ich odpowiednikami w postaci procentowego kodu, który składa się z symbolu "%"i dwóch cyfr szesnastkowych reprezentujących kod ASCII danego znaku.

```
    {Value: 4};
    {Value: 5}
);;
Set(UserVar; UżytkownicyusługiOffice365.MyProfile());;
ClearCollect(LocalServiceData; Lista_Uslug);;
ClearCollect(LocalCostData; Lista_Kwot);;
ClearCollect(LocalIndicationsData; Lista_Indykacji);;
ClearCollect(MergedData;
    AddColumns(LocalServiceData;
        Kwoty; LookUp(LocalCostData;
            Service_ID = LocalServiceData[@Service_ID] &&
            Year = Max(Filter(LocalCostData;
                Service_ID = LocalServiceData[@Service_ID]
            ); Year)
        );
        Indykacje; LookUp(LocalIndicationsData;
            Service_ID = LocalServiceData[@Service_ID] &&
            Year = Max(Filter(LocalIndicationsData;
                Service_ID = LocalServiceData[@Service_ID]
            ); Year) &&
            IndicationNo = Max(Filter(LocalIndicationsData;
                Service_ID = LocalServiceData[@Service_ID] &&
                Year = Max(Filter(LocalIndicationsData;
                    Service_ID = LocalServiceData[@Service_ID]
                ); Year)
            ); IndicationNo)
        )
    )
);;
Set(varDownloadingData; false);;
```

Listing 4.1: Kod wywoływany podczas uruchamiania aplikacji

W pierwszym kroku, zmiennej *varDownloadingData* przypisywana jest wartość *true* za pomocą funkcji *Set()*. Zmienna ta pełni kluczową rolę w zarządzaniu interfejsem użytkownika podczas procesu ładowania danych – aktywuje wskaźnik ładowania oraz blokuje możliwość wprowadzania zmian przez użytkownika, co zapobiega ewentualnym błędom wynikającym z prób modyfikacji danych w trakcie ich pobierania.

Następnie, funkcja *ClearCollect()* tworzy kolekcję *colYears*, która zawiera pięć elementów reprezentujących zakres lat: od dwóch lat wstecz do dwóch lat naprzód. Analogicznie, tworzona jest kolekcja *colNumbers*, zawierająca numery indykacji, które mogą być wykorzystywane w polach typu *Dropdown*. Kolekcje te są niezbędne do budowy dynamicznego i responsywnego interfejsu użytkownika, umożliwiając łatwe zarządzanie danymi w aplikacji.

W kolejnym kroku, za pomocą funkcji *Set()*, pobierane są informacje o aktualnie zalogowanym użytkowniku i przypisywane do zmiennej *UserVar*. Informacje te mogą być wykorzystywane do personalizacji interfejsu użytkownika lub kontroli dostępu do poszczególnych funkcji aplikacji, w zależności od uprawnień użytkownika.

Aplikacja tworzy również lokalne kopie trzech list danych: *Lista_Uslug*, *Lista_Kwot* oraz *Lista_Indykacji*, przy użyciu funkcji *ClearCollect()*. Lokalne kopie tych list, przechowywane odpowiednio w kolekcjach *LocalServiceData*, *LocalCostData* oraz *LocalIndicationsData*, pozwalają na

szybsze i bardziej efektywne filtrowanie oraz manipulację danymi podczas użytkowania aplikacji, redukując czas oczekiwania na odpowiedź systemu.

Kolejnym istotnym krokiem jest utworzenie kolekcji *MergedData*, która łączy dane z trzech lokalnych kopii list. W tym celu zastosowano funkcję *AddColumns()*, która dodaje dwie nowe kolumny: *Kwoty* oraz *Indykacje*. Dane w tych kolumnach są wyodrębniane przy użyciu funkcji *LookUp()* oraz *Filter()*, które umożliwiają precyzyjne filtrowanie i wyszukiwanie danych na podstawie określonych kryteriów. Funkcja *LookUp()* zwraca pierwszy rekord spełniający podane warunki, natomiast *Filter()* generuje zbiór rekordów spełniających zadane kryteria. W analizowanym kodzie funkcje te są zagnieżdżone, co pozwala na wyodrębnienie danych z list na podstawie trzech kluczowych kryteriów:

- *Service_ID* – identyfikator usługi,
- *Year* – najwyższy rok dla pasującego identyfikatora usługi,
- *IndicationNo* – maksymalny numer indykacji dla rekordów o zgodnym *Service_ID* i *Year*.

W efekcie, kolekcja *MergedData* zawiera dane dla najnowszego roku i najwyższego numeru indykacji dla każdej usługi, co umożliwia prezentację aktualnych informacji w interfejsie użytkownika.

Na końcu procesu, zmiennej *varDownloadingData* przypisywana jest wartość *false*, co sygnalizuje zakończenie pobierania danych i gotowość aplikacji do użytku. Lokalne kopie list (*LocalServiceData*, *LocalCostData*, *LocalIndicationsData*) zostały utworzone w celu przyspieszenia działania mechanizmu filtrowania oraz zwiększenia efektywności podczas wyboru usług do edycji.

4.4 Edycja danych

Po zapisaniu najnowszych danych, kolejnym krokiem jest wprowadzenie niezbędnych aktualizacji dotyczących usług. W tym celu zaimplementowano dwa ekrany: pierwszy służy do wyboru usługi z listy, a drugi umożliwia przeglądanie i edycję szczegółów związanych z wybraną usługą. Oba ekrany zostały zaprojektowane z myślą o intuicyjnej nawigacji i efektywnym zarządzaniu danymi.

4.4.1 Ekran wyboru usługi do edycji

Ekran wyboru usługi do edycji został zaprojektowany w sposób umożliwiający użytkownikom szybkie i precyzyjne wyszukiwanie oraz filtrowanie danych. Składa się z następujących elementów:

Lista wyboru serwisu

Lista prezentuje dane pochodzące z dynamicznej kolekcji *MergedData*, która została szczegółowo omówiona w poprzednim podrozdziale. Dzięki temu użytkownicy mają dostęp do aktualnych informacji bez konieczności ręcznego przeszukiwania list źródłowych.

Każdy element na liście posiada dodatkową funkcję interakcji. Po najechaniu kursorem na wybraną usługę (właściwość *Hover*), element wizualnie zmienia swój wygląd — zwęża się oraz zmienia kolor. Kliknięcie wybranego elementu, przenosi użytkownika do dedykowanego ekranu edycji, który umożliwia szczegółowe zarządzanie wybraną usługą.

Dodatkowo użytkownik ma możliwość zawężenia widocznych danych poprzez zastosowanie różnych kryteriów wyszukiwania. Pola obejmują:

- **Wyszukiwanie po nazwie usługi (*Service Name*)** — Obsługuje częściowe dopasowania dzięki wbudowanej funkcji *StartsWith*, która sprawdza, czy ciąg tekstowy zaczyna się od określonej frazy.
- **Filtrowanie po identyfikatorze usługi (*Service ID*)** — Umożliwia precyzyjne wyszukiwanie konkretnego elementu względem jego identyfkatora (*Service ID*).
- **Filtrowanie według miejsca powstawania kosztów (*MPK*)** — Pozwala na szybkie odnalezienie danych przypisanych do konkretnego obszaru finansowego.
- **Filtrowanie według statusu decyzji (*Accepted*, *Not Accepted*, *No Status*)**.

Filtry te mogą być stosowane jednocześnie, co umożliwia precyzyjne dopasowanie wyświetlanych danych.

Warto zauważyć, że informacje dotyczące podjętej decyzji, są przechowywane w bazie danych w postaci liczb całkowitych w celu szybszego przeszukiwania. Aby jednak w aplikacji widoczne były one w postaci tekstowej, zastosowano funkcję *Switch*, która przypisuje odpowiednią wartość tekstową na podstawie wartości liczbowej.

Ostatnim elementem tego ekranu jest wykres kołowy, który ilustruje podział danych według statusów decyzji. Dane widoczne na wykresie, pochodzą z listy wyboru usługi, co pozwala na uwzględnienie filtrów np. po wprowadzeniu numeru MPK użytkownik może sprawdzić jaka część usług należąca do danego obszaru, wymaga rozpatrzenia. Wartości przekazane są do właściwości *Items* i uwzględniają mapowanie wartości liczbowych na tekst.

4.4.2 Ekran edycji elementu

Ekran edycji elementu, prezentuje szczegółowe informacje dotyczące wybranej usługi, umożliwiając analizę danych historycznych oraz wprowadzanie nowych decyzji. Ekran składa się z kilku logicznie ułożonych sekcji, które są opisane poniżej.

Porównanie finalnych decyzji z lat poprzednich

W górnej części ekranu umieszczona została galeria, przedstawiająca porównanie danych finansowych oraz decyzji z trzech ostatnich lat. Dane, określone we właściwości *Items*, zawierają:

- **Rok (Year)** — okres, którego dotyczy dana decyzja.
- **Jednostka miary (Unit Of Measurement)** — zazwyczaj ilość licencji.
- **Zeszłoroczne i planowane na następny rok wartości finansowe** — *Current Year Plan* oraz *Next Year Plan*.
- **Różnice w finansach (Difference)** — różnica między zeszłorocznymi i potencjalnymi przyszłymi kosztami.
- **Status końcowej decyzji (Final Decision)** — decyzje dotyczące finalnej decyzji z danego roku (*Accepted, Not Accepted, No Status*).

TO Do INSTRUKCJI: Sekcja ta pozwala użytkownikowi przeanalizować ostatnie decyzje oraz ocenić trendy finansowe dla danej usługi w kolejnych latach.

Galeria ta jest interaktywna. Po kliknięciu na wybrany wiersz, wyświetlana jest kolejna galeria, która zawiera szczegółowe informacje na temat poszczególnych indykacji z wybranego roku.

Porównanie tegorocznych indykacji

Druga galeria wyświetla następujące informacje:

- **Numer indykacji (Indication Number)**,
- **Komentarze** — w tym *Internal Comment, Comment PZ to WOB, Comment K-DES*,
- **Data i autor komentarza**,
- **Status decyzji (Decision)**.

Aby uniknąć problemu z czytelnością długich komentarzy, które nie mieszczą się w przeznaczonych komórkach, zaimplementowano możliwość kliknięcia na element. Skutkuje to ustawieniem wartości zmiennej *ShowCommentPopup*, odpowiedzialnej za widoczność okna dialogowego, na wartość *True*. W oknie tym wyświetlany jest autor, data wprowadzenia oraz pełna treść komentarza. Dane dotyczące komentarza przekazywane są w zmiennej tablicowej *SelectedComment*, która aktualizowana jest w momencie wybrania komórki z komentarzem.

Formularz do uzupełnienia danych

Na samym dole strony znajduje się formularz umożliwiający wprowadzenie nowych danych lub aktualizację istniejących rekordów. Formularz zawiera pola takie jak:

- **Rok (*Year*)** — pole typu *Dropdown*, zawiera dane z kolekcji *colYears* tworzonej przy uruchomieniu aplikacji,
- **Numer indykacji (*Indication Number*)** — pole typu *Dropdown*, zawiera dane z kolekcji *colNumbers* tworzonej przy uruchomieniu aplikacji,
- **Komentarze** — pole *TextInput*,
- **Status decyzji (*Decision*)** — pole typu *Dropdown*. Dane we właściwości *Items*, zdefiniowane są jako *["Accepted", "No Status", "Not Accepted"]*.
- **Data** — kontrolka *DatePicker*, która umożliwia wybór daty poprzez wyświetlenia kalendarza,
- **Autor** — pole *ComboBox*, umożliwia wybór autora z listy pracowników. Lista ta pobierana jest poprzez connector *Office365Users*[7], który zawiera informacje na temat wszystkich użytkowników infrastruktury firmy.

TO DO INSTRUKCJI: \beginitemize \item \textbf{\Rok (\emph{Year})} — użytkownik może wybrać rok, którego dotyczy wpis. Domyślnie pole to jest ustawiane na bieżący rok. \item \textbf{\Numer indykacji (\emph{Indication Number})} — kolejny numer przypisany do decyzji. Numer ten jest automatycznie ustawiany na o jeden większy niż najwyższy numer indykacji dla danego roku w bazie danych. \item \textbf{\Komentarze} — pola do wprowadzenia uwag wewnętrznych, komentarzy między działami oraz końcowych komentarzy. \item \textbf{\Status decyzji (\emph{Decision})} — lista rozwijana, umożliwiająca wybór odpowiedniego statusu (\emph{Accepted, Not Accepted, No Status}). Domyślnie status jest ustawiany na podstawie decyzji z poprzedniego roku. \item \textbf{\Data i autor} — data oraz osoba odpowiedzialna za wprowadzenie wpisu. Pole autora jest automatycznie uzupełniane danymi aktualnie zalogowanego użytkownika. \enditemize W kodzie pod tą notatką jest zakomentowany kod do tej listy

[7] *Office365Users* – konektor pozwalający na dostęp do listy zawierającej informacje na temat użytkowników takie jak imie, nazwisko, dane kontaktowe lub dział.

Ostatnim elementem na tym ekranie jest przycisk *Save*, który umożliwia zapisanie wprowadzonych zmian. Mechanizm ten:

- Sprawdza istnienie wcześniejszych indykacji, aby upewnić się, że zachowana jest poprawna kolejność numeracji.
- W przypadku istniejącego wpisu — uzupełnia dane przy użyciu funkcji *Patch*.
- W przypadku nowego wpisu — tworzy nowy rekord przy użyciu funkcji *Defaults*.
- Resetuje pola formularza oraz odświeża dane na ekranie, aby uwzględnić ostatnie zmiany.
- Informuje użytkownika o powodzeniu lub błędach operacji za pomocą komunikatów (*Notify*).

4.4.3 Listing kodu

Listing 4.2 przedstawia fragment kodu odpowiedzialny za zapisywanie danych w formularzu edycji, który jest wywoływany po kliknięciu przycisku *Save*. Kod ten odpowiedzialny jest za zapisanie informacji z formularza w bazie danych.

```
1     If(
2         // Dla indykacji nr 1
3         IndicationNo_Dropdown.Selected.Value = 1;
4
5         // Sprawdź; czy istnieje już pierwsza indykacja
6         If(
7             !IsBlank(
8                 LookUp(
9                     Lista_Indykacji;
10                    Year = Year_Dropdown.Selected.Value &&
11                    IndicationNo = 1 &&
12                    Service_ID = ChosenServiceID
13                )
14            );
15            // Aktualizacja istniejącego rekordu
16            Set(
17                TempOutput;
18                Patch(
19                    Lista_Indykacji;
20                    LookUp(
21                        Lista_Indykacji;
22                        Year = Year_Dropdown.Selected.Value &&
23                        IndicationNo = 1 &&
24                        Service_ID = ChosenServiceID
25                    );
26                    {
27                        Comment_date: Text(CommentDateInput.SelectedDate);
28                        Comment_author: ComboboxCanvas1.Selected.
                              DisplayName;
29                        Comment_PZ_to_WOB: CommentToWOBInput.Value;
30                        Comment_Intern: InternalCommentInput.Value;
31                        Decision: Switch(
32                            DropdownCanvas1.Selected.Value;
```

```
                                "Accepted"; 1;
                                "No Status"; 0;
                                "Not Accepted"; -1;
                                0
                            )
                        }
                    )
                );;
                Notify("A record has been successfully updated!";
                    NotificationType.Success);

                // Jeśli rekord nie istnieje; utwórz nowy
                Set(
                    TempOutput;
                    Patch(
                        Lista_Indykacji;
                        Defaults(Lista_Indykacji);
                        {
                            Service_ID: ChosenServiceID;
                            Year: Year_Dropdown.Selected.Value;
                            IndicationNo: 1;
                            Comment_date: Text(CommentDateInput.SelectedDate);
                            Comment_author: ComboboxCanvas1.Selected.
                                DisplayName;
                            Comment_PZ_to_WOB: CommentToWOBInput.Value;
                            Comment_Intern: InternalCommentInput.Value;
                            Decision: Switch(
                                DropdownCanvas1.Selected.Value;
                                "Accepted"; 1;
                                "No Status"; 0;
                                "Not Accepted"; -1;
                                0
                            )
                        }
                    )
                );;
                Notify("The record has been successfully created!";
                    NotificationType.Success)
            )
        );;
```

Listing 4.2: Kod zapisywania danych w formularzu edycji

4.5 Generowanie raportu

Następnym ekran umożliwia wygenerowania raportu w formacie arkusza kalkulacyjnego na podstawie uzupełnionej bazy danych. W związku z tym w Power Apps przygotowano mechanizm łączący dane z trzech list SharePoint, które następnie są przesyłane do Power Automate, gdzie podlegają dalszemu przetwarzaniu. Wynikiem tego jest kompletny arkusz Excel, który jest zapisywany na platformie SharePoint.

4.5.1 Łączenie danych z list

W celu określenia, roku i indykacji których ma dotyczyć raport, wprowadzono kontrolki *Dropdown* po lewej stronie ekranu. Ich uzupełnienie skutkuje wygenerowaniem podglądu danych, który pozwala na walidację ich poprawności. Listing 4.3 przedstawia kod odpowiedzialny za utworzenie kolekcji *CombinedData*, która zawiera odpowiednio odfiltrowane informacje.

```
    ClearCollect(
    CombinedData;
    AddColumns(
        LocalServiceData;
        MPK;
        LookUp(
            LocalCostData;
            Service_ID = LocalServiceData[@Service_ID] && Year =
                Year_Dropdown_1.Selected.Value;
            MPK
        );
        Comment_PZ_to_WOB;
        LookUp(
            LocalIndicationsData;
            Service_ID = LocalServiceData[@Service_ID] && Year =
                Year_Dropdown_1.Selected.Value && IndicationNo =
                IndicationNo_Dropdown_1.Selected.Value;
            Comment_PZ_to_WOB);
    [...]
);;
```

Listing 4.3: Fragment kodu tworzącego kolekcję CombinedData

Kod ten korzysta z następujących funkcji:

- *ClearCollect* – przypisuje dane do kolekcji o podanej nazwie,
- *AddColumns* – pozwala na dodanie do kolekcji nowej kolumny i formuły definiującej jej wartości,
- *LookUp* – wyszukuje rekordy w tabeli na podstawie podanych kryteriów.

4.5.2 Przekazywanie danych do Power Automate

W lewym dolnym rogu interfejsu znajduje się przycisk *Generate file*, który po naciśnięciu uruchamia przepływ Power Automate, odpowiedzialny za stworzenie pliku na podstawie danych wejściowych. Dane te są przekazywane do usługi Power Automate w postaci ciągu tekstowego, sformatowanego jako JSON[8], wykorzystując połączone informacje z trzech źródeł zawarte w kolekcji *CombinedData*. Kod wywołujący funkcję *GenerateRaport* przedstawiono w listingu 4.4.

```
    GenerateRaport.Run(
        Substitute(
            "[" &
            Concat(
                CombinedData;
```

[8] JSON - format wymiany danych, który jest oparty na strukturze tekstowej

```
6                   "{""Service_ID"":""" & Service_ID & """," &
7                   """Service_Name"":""" & Service_Name & """," &
8                   """MPK"":""" & MPK & """," &
9                   """Comment_PZ_to_WOB"":""" & Comment_PZ_to_WOB & """," &
10                  """Decision"":""" & Decision & """},"
11              ) &
12              "]",
13              "},]";
14              "}]"
15          ),
16          IndicationNoCollect.Value,
17          YearNoCollect.Value
18      )
```

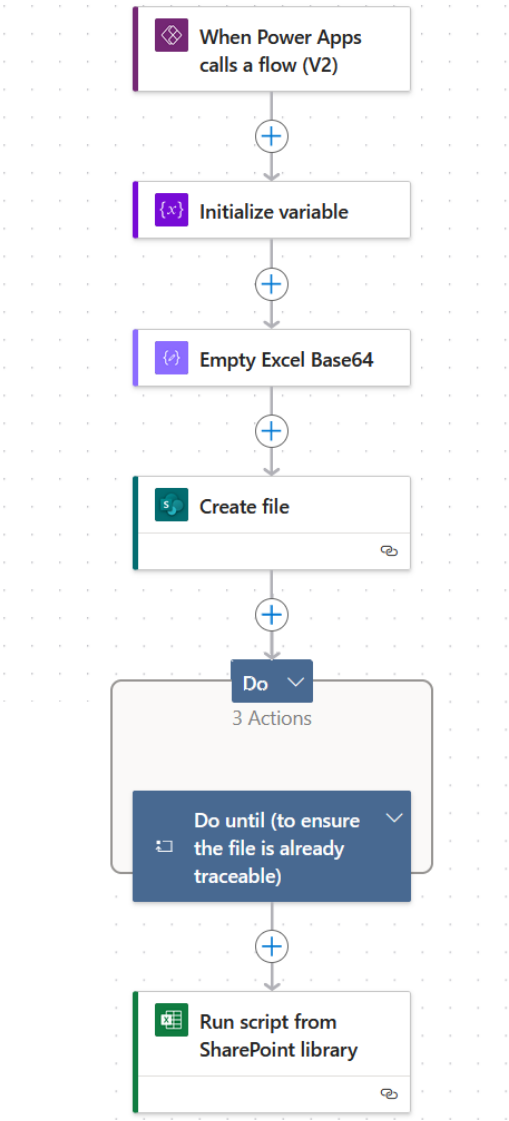
Listing 4.4: Kod wywołujący funkcję GenerateRaport

4.5.3 Generowanie raportu w Power Automate

Flow o nazwie *GenerateRaport* składa się z kilku komponentów, które pozwalają na stworzenie raportu w formacie arkusza kalkulacyjnego, w oparciu o dane z list SharePoint. Przepływ ten zawiera następujące instrukcje:

- **Wejście danych:** *Flow* przyjmuje trzy dane wejściowe: ciąg znaków w formacie JSON, zawierający dane przekazane z aplikacji do przetworzenia, wybraną indykację oraz wybrany rok.
- **Inicjalizacja zmiennej:** W tym kroku tworzona jest zmienna odpowiedzialna za nazwę generowanego pliku. Zmienna zawiera dynamicznie generowaną datę utworzenia pliku.
- **Generowanie pustego arkusza Excel:** Wykorzystywany jest statyczny plik Excel, zapisany w formacie Base64, który pełni rolę szablonu dla dalszego przetwarzania.
- **Tworzenie pliku na SharePoint:** Szablon arkusza zostaje zapisany w określonej lokalizacji w bibliotece SharePoint.
- **Sprawdzenie dostępności pliku:** W celu upewnienia się, że plik jest gotowy do dalszego przetwarzania, uruchamiana jest pętla *Do until*, która weryfikuje dostępność pliku na platformie.
- **Uruchomienie skryptu:** Na końcu przepływu wywoływany jest skrypt pakietu office, który generuje wypełniony arkusz Excel.

Schemat 4.3 przedstawia szczegółowy schemat przepływu, uwzględniający poszczególne kroki oraz przepływ danych między nimi.



Rysunek 4.3: Schemat przepływu *GenerateRaport*

Opis działania skryptu generującego raport

Skrypt odpowiedzialny za generowanie raportu wykonuje kilka operacji. Jego główne funkcje to:

- **Analiza danych wejściowych:** Przyjęcie danych w formacie JSON oraz roku. W przypadku otrzymania niepoprawnych dancyh, w arkuszu umieszczony jest odpowiedni komunikat.
- **Tworzenie arkusza:** Sprawdzenie, czy arkusz wynikowy już istnieje – jeśli tak, to usuwa go i tworzy nowy.
- **Dynamiczne wstawianie danych:** Kolumny zawierające informacje na temat cen, mają w nagłówkach podany rok, których one dotyczą.
- **Generowanie tabeli:** Tworzenie nagłówków tabel (np. `Service_ID`, `Service_Name`) oraz wprowadzenie danych wypełniające tabelę, pobierane z JSON.
- **Formatowanie:** Skrypt aktywuje opcję filtracji kolumn oraz zmienia ich szerokość. Dzięki temu arkusz jest czytelny i nie wymaga formatowania ze strony użytkownika.

4.6 Składniki

Składniki to niestandardowe elementy, które można implementować w środowisku Power Apps. Są one wykorzystywane do tworzenia złożonych komponentów, które nie są dostępne w standardowych kontrolkach. Przydatne są szczególnie w momencie kiedy chcemy stworzyć element, który będzie wykorzystywany w wielu miejscach aplikacji. W zależności od potrzeb, można stworzyć zarówno proste elementy, jak i bardziej skomplikowane, które będą spełniały określone wymagania. Ponieżej omówiono dwa składniki stworzone w ramach aplikacji.

4.6.1 Nagłówek ekranu

Pierwszym niestandardowym komponentem jest wstążka znajdująca się na górze ekranów. Składa się ona z następujących elementów:

- logo firmy – jest to element dekoracyjny,
- przycisk powrotu do poprzedniego ekranu – oznaczony strzałką w lewo, dzięki funkcji *Back()* przenosi użytkownika do poprzedniego ekranu,
- przycisk powrotu do ekranu głównego – reprezentowany poprzez ikonę domu, przenosi użytkownika do ekranu głównego aplikacji (*Navigate(HomeScreen, ScreenTransition.Cover)*),
- dane użytkownika – wyświetlane są dane zalogowanego użytkownika, takie jak imię, nazwisko oraz adres email. Pobierane są one przy pomocy *Office365Users.MyProfile()*[9].
- awatar użytkownika – również pobierany z użyciem *Office365Users.MyProfile()*. W przypadku braku zdjęcia, wyświetlane są inicjały użytkownika.

Rysunek 4.4: Nagłówek ekranu

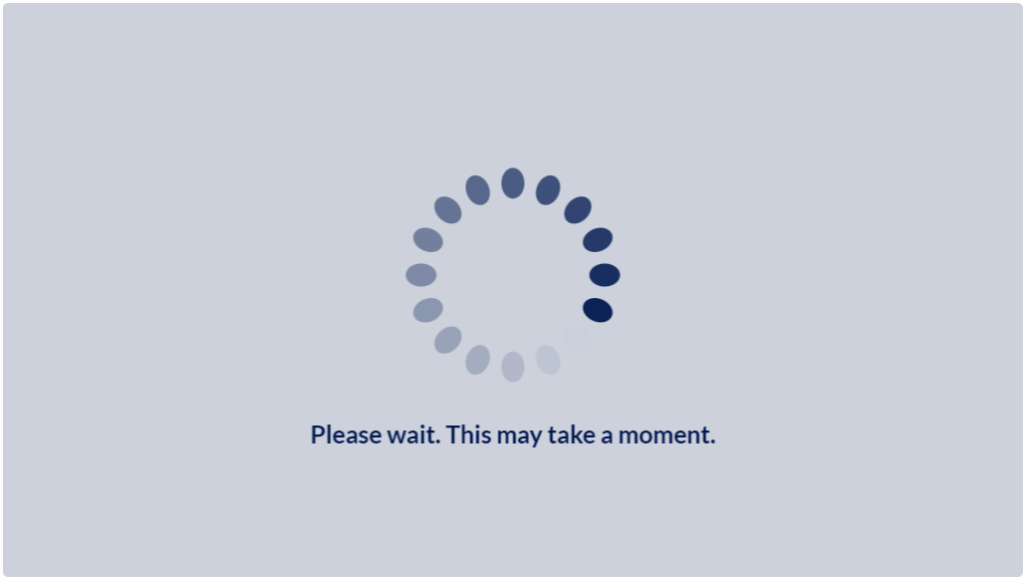
Warto zauważyć, że tytuły widoczne na ekranach, nie są integralną częścią wstążki. Wynika to z faktu, że po dodaniu kontrolki *label*, pozwalającej na wyświetlenie tekstu, nie jest możliwe zmienienie jej tekstu we właściwościach utworzonego składnika. W związku z tym, tytuły są dodawane na każdym ekranie osobno.

4.6.2 Indykator ładowania

Drugim niestandardowym składnikiem jest kontrolka infromująca o ładowaniu się aplikacji. Składa się ona z trzech elementów:

- koło ładowania – animowana grafika w formacie *SVG*(*Scalable Vector Graphics*).
- tekst – informacja dla użytkownika, że aplikacja ładuje dane,
- tło – półprzezroczyste tło z filtrem powodującym rozmycie elementów ekranu. Tło wykonane przy użyciu *tekst HTML*.

[9] Atrybut *MyProfile* pozwala na dostęp do informacji o bieżącym użytkowniku.



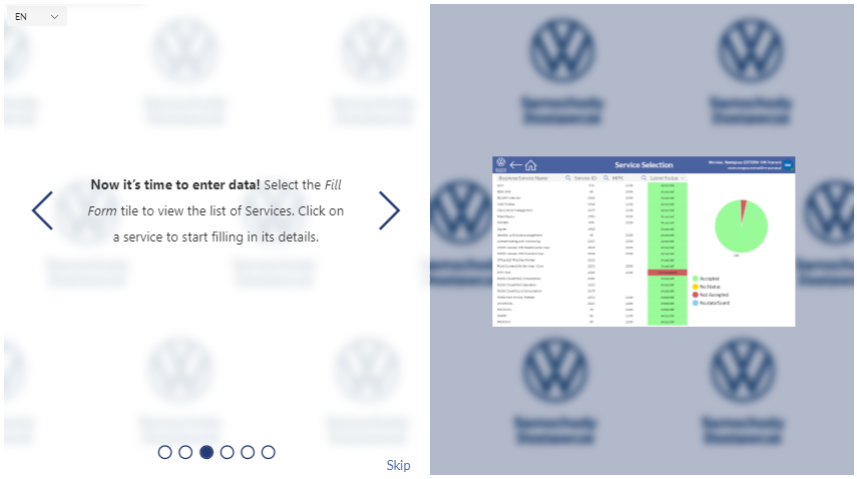
Rysunek 4.5: Kontrolka ładowania

4.7 Samouczek

Ostatnim z ekranów aplikacji jest samouczek, do którego bezpośredni dostęp został zapewniony z poziomu ekranu domowego.

Do jego wykonania wykorzystano wbudowany szablon *TemplateScreen*, który składa się z dwóch pionowych pól - tekstowego z lewej strony oraz multimediów z prawej. Na widok całości składają się dodatkowo ikony strzałek, umożliwiających nawigację po kolekcji tekstów oraz nawigatora, prezentującego aktualny postęp prezentacji samouczka.

Obraz 4.6 przedstawia przykładowy widok działania samouczka.



Rysunek 4.6: Prezentacja widoku samouczka *GenerateRaport*

4.7.1 Nawigacja po samouczku

```
1 Set(guideStep; Min(guideStep+1; Last(TutorialNavigator1.AllItems).Step))
```

Listing 4.5: Kod wywoływany podczas realizacji przycisku nawigacji po samouczku – przycisk następnego elementu

Listing 4.5 przedstawia fragment kodu użytego do nawigacji. Funkcja *Set()* jest wykorzystana do aktualizacji zmiennej *guideStep*, która przechowuje wartość aktualnego kroku samouczka. Funkcja

Min() zapewnia, że wartość zmiennej kroku nie przekroczy ostatniego możliwego kroku. Analogicznie działanie wykorzystano dla nawigacji wstecz (strzałka w lewo).

4.7.2 Prezentacja zawartości w samouczku

W samouczku opisano wszystkie ekrany w kolejności postępu procesu. Każdy z opisów znajduje się w kolekcji tekstów, jako osobny element.

```
1 If(IsBlank(guideStep); First(TutorialNavigator1.AllItems).Text; LookUp(
    TutorialNavigator1.AllItems; Step = guideStep).Text);
```

Listing 4.6: Kod pobierający elementy tekstowe z kolekcji

W kodzie widocznym w listingu 4.6, funkcja *If()* sprawdza, czy zmienna *guideStep* jest pusta – jeśli tak, to pobierany jest pierwszy element z kolekcji *TutorialNavigator1.AllItems* za pomocą *First()*. W przeciwnym razie, funkcja *LookUp()* wyszukuje element o pasującym numerze kroku, zwracając odpowiadający tekst.

Analogicznie jak w przypadku wyświetlania tekstu, grafiki ilustracyjne aplikacji są dynamicznie pobierane zgodnie z postępem nawigacji. Każdy z etapów samouczka zawiera odpowiedni obrazek, który wyświetlany jest razem z przypisanym do niego tekstem.

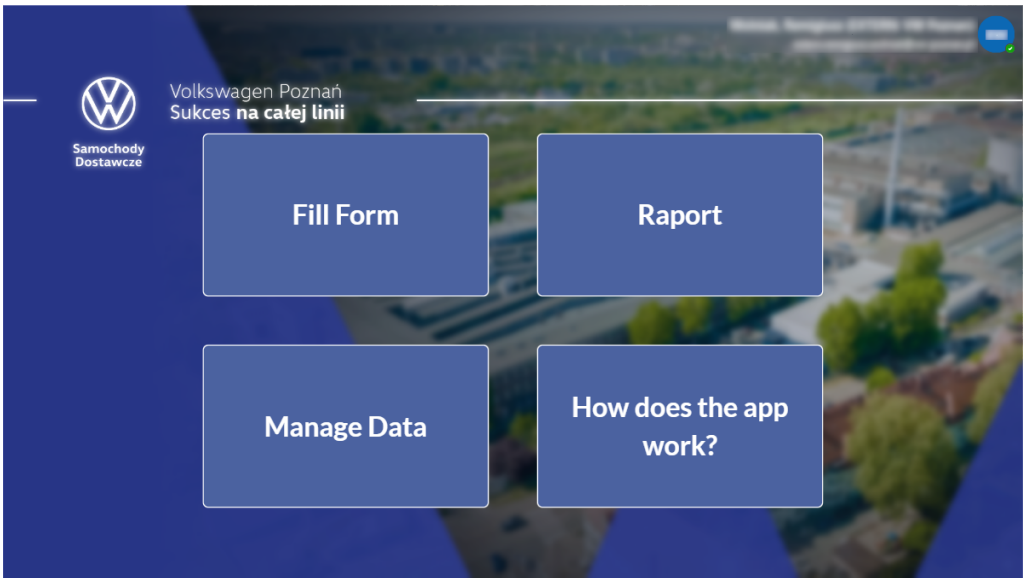
Rozdział 5

Prezentacja działania

W nieniejszym rozdziale przedstawiono działanie systemu. Opisano w nim poszczególne ekrany aplikacji oraz ich funkcjonalność.

5.1 Ekran startowy

Po uruchomieniu aplikacji, pierwszym ekranem widocznym dla użytkownika, jest ekran startowy, przedstawiony na rysunku 5.1. Składa się on z czterach przycisków, które po naciśnięciu przekierowują użytkownika do odpowiednich sekcji aplikacji.



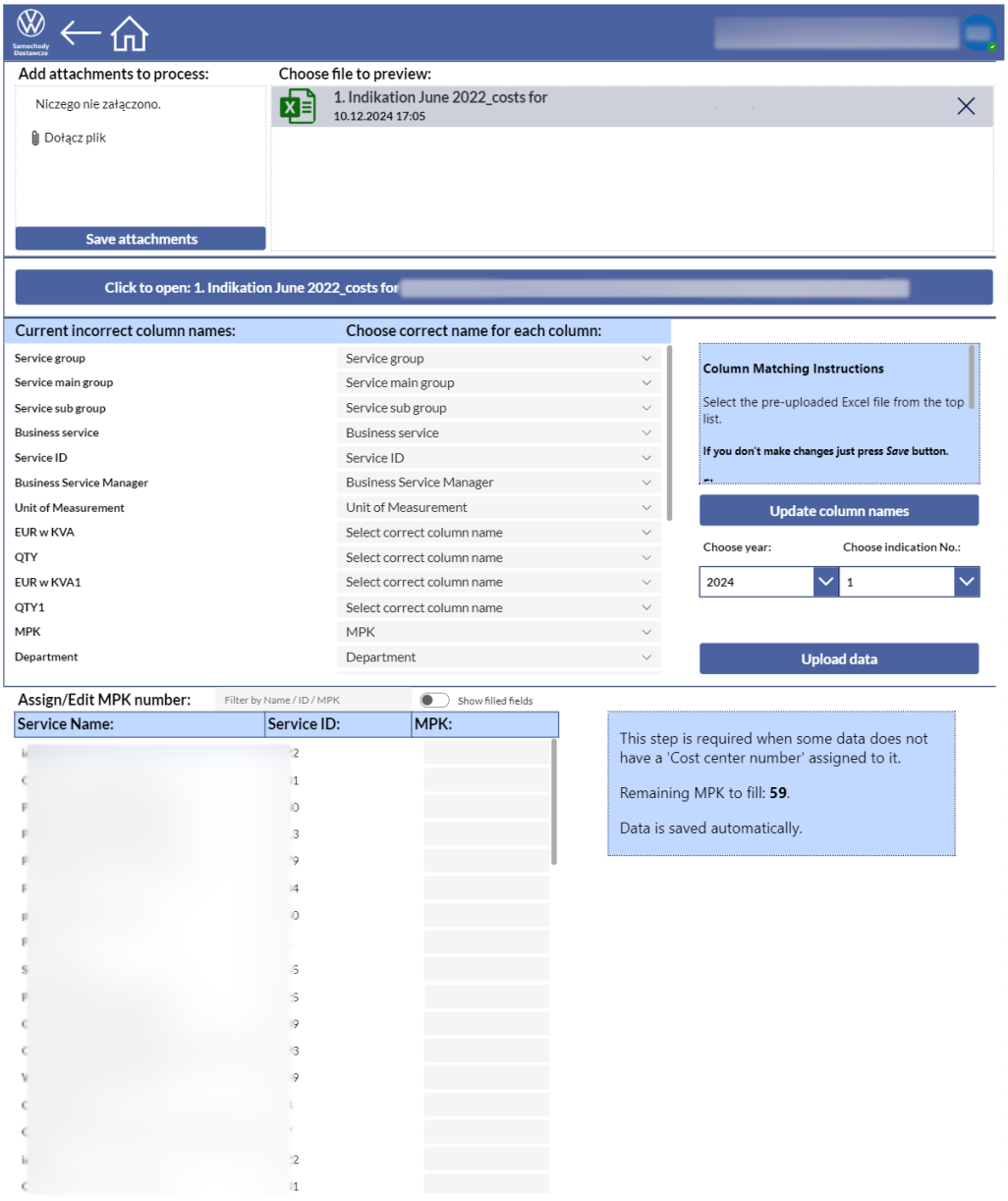
Rysunek 5.1: Ekran startowy aplikacji

Ten ekran aplikacji, pełni funkcję panelu nawigacji dla użytkownika, umożliwiając intuicyjne poruszanie się po innych sekcjach systemu. Poniżej przedstawiono, który przycisk odpowiada za przekierowanie do danej sekcji aplikacji:

- *Manage Data* → Ekran dodawania danych,
- *Fill Form* → Ekran wypełniania formularza,
- *Raport* → Ekran generowania raportu,
- *How does tha app work?* → Ekran samouczka.

5.2 Ekran dodawania danych

Rysunek 5.2 przedstawia ekran dodawania danych do systemu. Przeznaczony jest on do wprowadzenia nowych informacji o serwisach, które pochodzą z arkusza kalkulacyjnego.

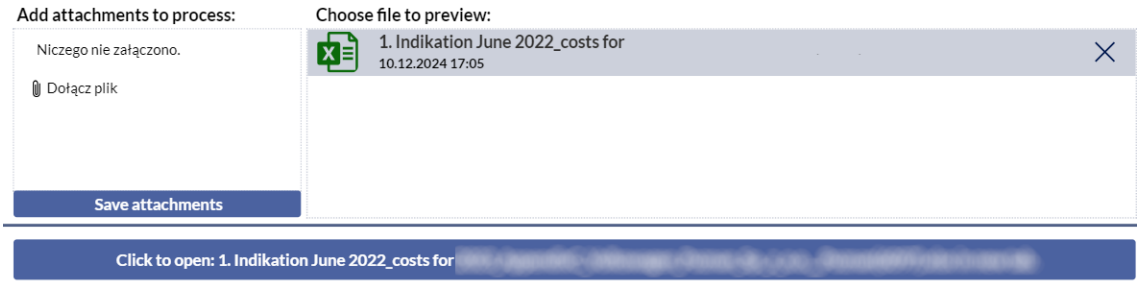


Rysunek 5.2: Ekran dodawania danych do systemu

Ekran ten podzielony został na trzy główne sekcje:

- Sekcja dodawania pliku do systemu,
- Formularz walidacji kolumn,
- Formularz uzupełniania numerów MPK.

Sekcja dodawania pliku do systemu



Rysunek 5.3: Formularz zapisu plików do systemu

Rysunek 5.3 zawiera elementy służące do dodawaniu plików do systemu oraz nawigacji po nich.

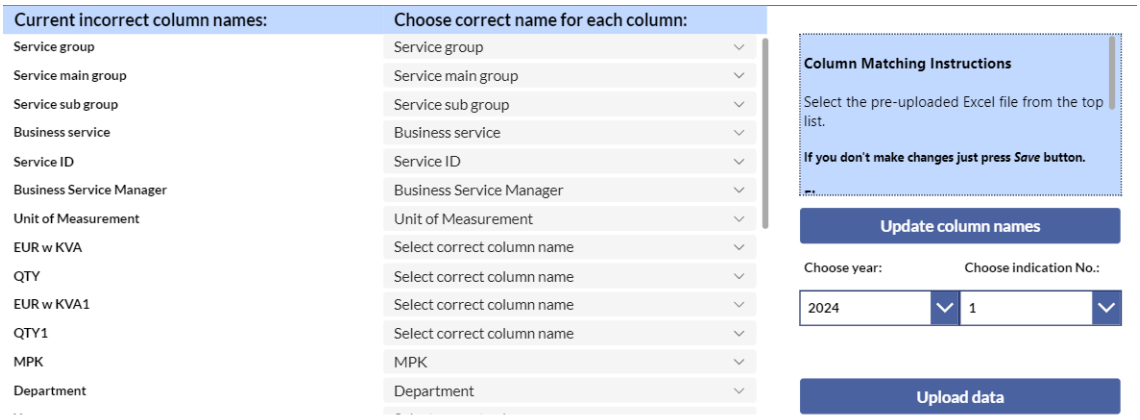
Kontrolka opisana jako *Add attachments to process*, pozwala na dodanie nowego arkusza kalkulacyjnego do systemu. Wybór pliku odbywa się przy pomocy przycisku *Add attachments* lub z użyciem mechaniki *Drag&Drop*. Kiedy plik zostanie pomyślnie dodany do pamięci aplikacji, wyświetli się on na liście.

Aby plik został dodany do bilbioteki SharePoint, należy nacisnąć przycisk *Save attachments* znajdujący się podniżej. Naciśnięcie przycisku skutkuje rozpoczęciem procesu zapisu oraz wstępnego formatowania danych. W tym czasie na ekranie widoczny będzie wskaźnik informujący użytkownika o trwającym procesie.

Po prawej stronie znajduje się lista plików, które zostały poprawnie zapisane w folderze tymczasowym Sharepoint. Lista ta pozwala na wybór pliku, który zostanie przetworzony w kolejnych krokach. Dodatkowo krzyżyk przy każdym pliku pozwala na usunięcie go z systemu – w tym z folderu tymczasowego.

Ostatnim elementem tej sekcji jest przycisk *Clock to open:...*. Jego naciśnięcie skutkuje otwarciem pliku wybranego z listy w nowej karcie przeglądarki. Pozwala to na weryfikację poprawności zapisanego pliku oraz jest pomocne w kolejnym kroku procesu.

5.2.1 Formularz walidacji kolumn

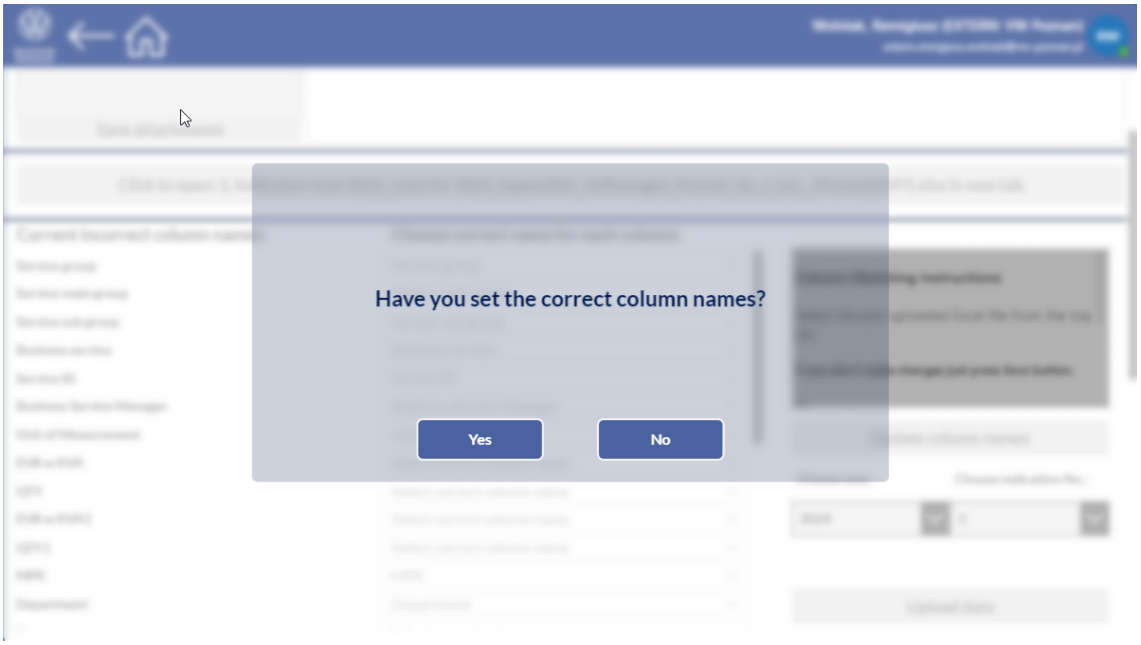


Rysunek 5.4: Formularz walidacji nazw kolumn

Głównym elementem formularza jest dwukolumnowa galeria, dzięki której użytkownik może dostosować nazwy kolumn w arkuszu kalkulacyjnym do nazw kolumn w bazie danych. W lewej kolumnie znajdują się nagłówki pobrane z arkusza natomiast w prawej kolumnie znajdują się rozwijane listy pozwalające na dopasowanie odpowiedniej nazwy. Na rysunek 5.4 widać, że niektóre

nazwy zostały już przypisane. Wynika to z działania skryptu, który automatycznie podpowiada ich nazwę co pozwala przyspieszyć ten krok. Tutaj również przydatna jest funkcja podglądu pliku, ponieważ pozwala ona na określenie prawidłowej nazwy na podstawie zawartości kolumny.

Po prawej stronie umieszczono dodatkowe kontrolki. Pierwszą z nich, jest krótka instrukcja dla użytkownika, umieszczona na niebieskim tle. Pod nią znajduje sie przycisk *Update column names*, który pozwala na zapisanie zmian w nazwach kolumn. Po jego naciśnięciu, wyświetlone zostanie okno dialogowe (rysunek 5.5) z zapytaniem o potwierdzenie zmian. Zmiana nazw kolumn jest bardzo ważny dla dalszego działania aplikacji, dlatego zaleca się dokładne sprawdzenie poprawności wybranych nazw. Zatwierdzenie zmiany, skutkuje rozpoczęciem procesu nadpisania nagłówków,



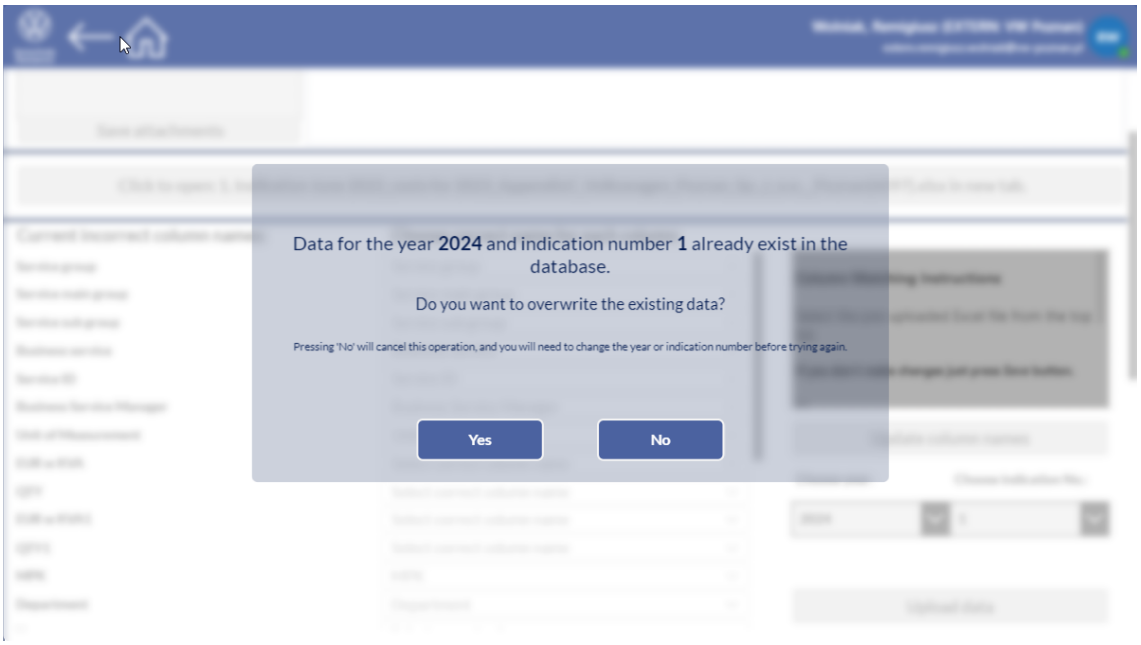
Rysunek 5.5: Zapytanie o poprawność nazw kolumn

oczym informuje odpowiedni indykator. Po zakończeniu procesu, galeria z nazwami kolumn zostaje zaaktualizowana.

Formularz ten pozwala również na przesłanie danych z arkusza do bazy danych. Pierwszym krokiem jest wybór roku oraz numeru indykacji, których dotyczą dane przy użyciu rozwijanych list znajdujących się pod przyciskiem aktualizującym nazwy kolumn. Kiedy wybrano odpowiednie wartości, można zainicjować proces zapisu danych poprzez użycie przycisku *Upload data*. Naciśnięcie go skutkuje trzema scenariuszami:

- Kiedy nie zmieniono nazw kolumn, wyświetlone zostaje okno dialogowe z rysunku 5.4. Ma to na celu upewnienie się, że nagłówki są poprawne.
- Kiedy wybrano rok i numer indykacji dla których informacje istnieją już w bazie danych, wyświetlone zostaje okno dialogowe widoczne na rysunku 5.6. Informuje ono użytkownika, o istniejących danych i daje możliwość ich nadpisania lub anulowania operacji.
- Kiedy nazwy zostału uprzednio zmienione i kiedy wybrano unikalne wartości dla roku i numeru indykacji, proces zapisu danych zostaje rozpoczęty bez dodatkowych zapytań.

Występuje również przypadek kiedy omówione okna dialogowe wyświetlają się jedno o drugim. W takiej sytuacji, użytkownik musi najpierw potwierdzić zmianę nazw kolumn, a następnie zdecydować czy chce nadpisać dane.

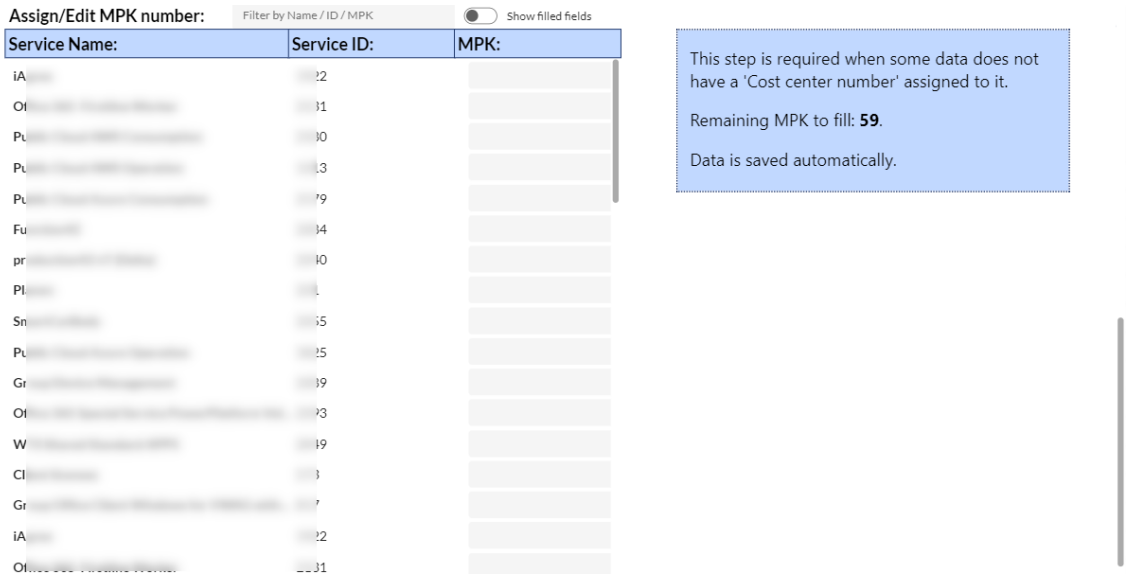


Rysunek 5.6: Zapytanie o nadpisanie danych

Proces zapisu danych jest najdłużej trwającym etapem. Czas zapisu wynosi od 30 do 60 sekund w zależności od ilości danych oraz jakości połączenia z serwerem. Z uwagi na wykorzystanie mechanizmu zapisu z użyciem żądań zbiorowychm, nie jest możliwe poinformowanie użytkownika o postępie operacji.

Po ukończeniu operacji wyświetlana jest informacja o zakończeniu procesu. Użytkownik zostanie również poinformowany w razie wystąpienia błędów.

5.2.2 Formularz uzupełniania numerów MPK



Rysunek 5.7: Formularz wypełniania/edycji numerów MPK

Ostatnią sekcją tego ekranu jst formularz edycji numerów MPK widoczny na rysunku 5.7. Składa się on z galerii zawierającej następujące kolumny:

- *Service Name* – nazwa serwisu,

- *Service ID* – unikalny identyfikator serwisu,
- *MPK* – miejsce powstawania kosztów.

Ostatnia z nich jest edytowalna, co pozwala na nadanie lub zmianę numeru MPK. Zmiany są automatyczie zapisywane.

Nad galerią umieszczono pole tekstowe, które pozwala na filtrowanie wyników względem każdej z kolumn.
Obok znajduje się przełącznik, kótry pozwala na przełączenie się miedzy widokiem wypełnionych oraz pustych numerów MPK.
Należy pamiętać, że system automatycznie przypisuje numer MPK na podstawie informacji z poprzedniego roku co pozwala na minimalizację wprowadzanych danych.

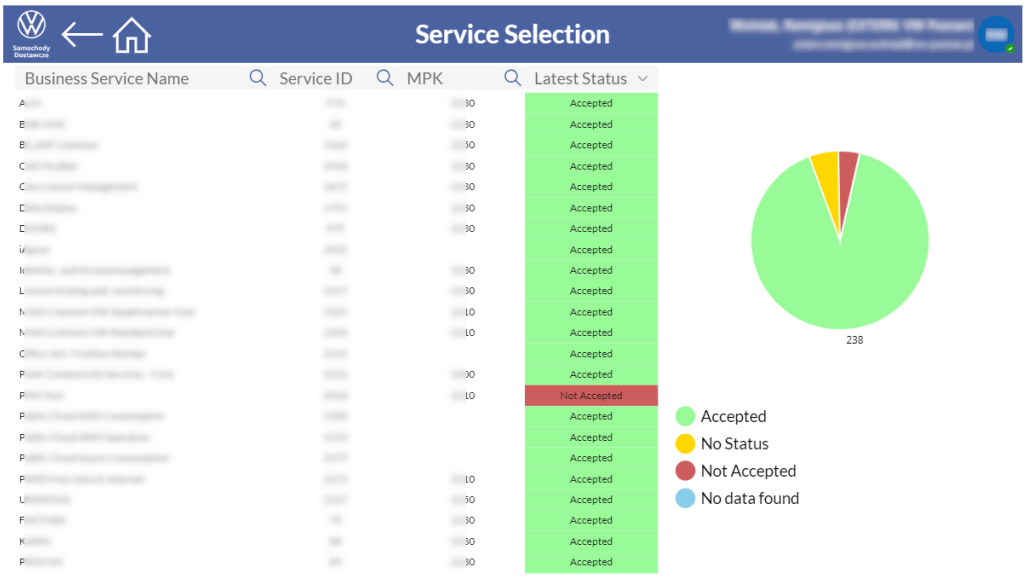
Po prawej stronie znajduje się kolejne pole informacyjne, które zawiera instrukcję dla użytkownika oraz liczbę serwisów, które nie są przypisane do żadnego obszaru.

Taki układ formularza pozwala na szybkie dodanie danych do systemu pozwalając na elastyczne dostosowanie ich do struktury bazy danych.

5.3 Ekran wypełniania formularza

Proces podejmowania decyzji odbywa się osobno dla każdej z usług. Dlatego ta część aplikacji składa się z dwóch ekranów: ekranu nawigacyjnego oraz ekranu szczegółowego.

5.3.1 Ekran nawigacyjny



Rysunek 5.8: Ekran wyboru usługi do edycji

Na tym ekranie umieszczono listę wszystkich serwisów, które znajdują się w bazie danych. Użytkownik ma możliwość filtrowania oraz wyszukiwania serwisów według różnych kryteriów poprzez pola znajdujące sę nad listą.

Po naciśnięciu wiersza odpowiadającego serwisowi, użytkownik zostaje przekierowany do ekranu szczegółowego.

Dodatkowo, po prawej stronie znajduje się wykres kołowy. Każdy segment odpowiada liczbie elementów z przypisanym statusem, co pozwala użytkownikowi szybko ocenić proporcje między kategoriami (*Accepted*, *Not Accepted*, *No Status*). Wykres jest dynamiczny — aktualizuje się w czasie rzeczywistym w oparciu o zastosowane filtry, co zapewnia aktualność prezentowanych informacji.

Rozdział 6

Napotkane problemy i rozwiązania

Problemy i ich rozwiązania

- **Relacje między listami w SharePoint:**
  - **Problem:** Sharepoin pozwala na stworzenie relacji między listami poprzez użycie kolumn typu *lookup*. Mechanizm ten pozwala na tworzenie relacji tylko pomiędzy dwiema listami. To ograniczenie stwarza problemy w przypadku, gdy aplikacja wymaga utworzenia bardziej złożonych relacji, obejmujących więcej niż dwie listy.
  - **Rozwiązanie:** Zdecydowano się na implementację relacji między listami na poziomie aplikacji, zamiast polegać na wbudowanym mechanizmie SharePoint. W tym celu wykorzystywano funkcję *lookup*, umożliwiającą tworzenie powiązań między trzema listami. Dzięki temu, relacje między listami były definiowane w sposób bardziej elastyczny i dostosowany do potrzeb aplikacji. Szczegóły implementacji tej logiki zostały opisane w podrozdziale [4.3], gdzie omówiono sposób zarządzania powiązaniami i przechowywania informacji o relacjach między listami w kontekście aplikacji. No tutaj nie tylko lookup ale też filter było wykorzystywane. Bo lookup znajduje ci pierwszy pasujący wiersz a filter wypluwa ci listę pasujących elementów. I my robiliśmy te śmieszne zagnieżdżenia.
- **Wykorzystanie przepływów podrzędnych w Power Automate:**
  - **Problem:** W Power Automate istnieje problem z odświeżaniem statusu pliku, który został zapisany na SharePoint. Główny przepływ nie bierze pod uwagę zmiany statusu pliku po jego zapisaniu, przez co system nie jest w stanie kontynuować przetwarzania pliku. Po zapisaniu pliku, przepływ próbuje pracować na nieaktualnych danych, wskazując iż plik jeszcze nie istnieje lub nie jest gotowy do dalszego przetwarzania. Prowadzi to do błędów w procesach automatyzacji, uniemożliwiając poprawne wykonanie funkcji.
  - **Rozwiązanie:** W celu rozwiązania tego problemu, wydzielono etap przetwarzania pliku do osobnego przepływu podrzędnego. Przepływ główny zapisywał plik, a następnie uruchamiał przepływ podrzędny, który pobierał aktualne dane z SharePoint, co eliminowało problem z odświeżaniem statusu. Wykorzystanie przepływów podrzędnych poprawiło stabilność i płynność procesu automatyzacji, umożliwiając prawidłowe przetwarzanie plików. Imo to jest git, bo tam tylko mówimy, że używamy a wyjaśnienie masz tutaj. Conajwyżej można dodać refa do tego miejsca.
- **Ustawienia lokalne - średniki i przecinki:**

  - TO BYŁ TYLKO PROBLEM JAK WRZUCAŁEŚ FUNKCJE Z CHATA ALBO Z FORUM XDD ja bym to wywalił. w sensie mogłoby zostać ale trochę do pizdy jest to napisane moim zdaniem. Możemy zapytać pikeja co sadzi
  - **Problem:** Różnice w ustawieniach lokalnych, dotyczące separatorów argumentów w funkcjach (średniki i przecinki) w Power Apps, powodują błędy w przetwarzaniu danych. W Polsce jako separator argumentów funkcji używa się średnika (`;`), podczas gdy w innych częściach świata (np. w przypadku ustawień dla języka angielskiego) może zastosowany zostać przecinek (`,`). Tego rodzaju niezgodność prowadzi do problemów przy korzystaniu z niepolskojęzycznej dokumentacji czy przy współpracy między różnymi członkami zespołu, którzy mają różne ustawienia lokalne w swoich systemach. Zmiany w separatorach mogą powodować, że funkcje nie są wykonywane poprawnie, ponieważ jeden użytkownik używa przecinków, a inny średników. Różnice występują też np. w przypadku znaków końca linii.
  - **Rozwiązanie:** Aby zminimalizować ryzyko błędów związanych z różnicami w ustawieniach regionalnych, konieczne jest ujednolicenie tych ustawień wśród członków zespołu.

- **Ograniczenia w tworzeniu elementów na SharePoint:**
  - **Problem:** SharePoint posiada ograniczenie dotyczące liczby elementów, które można utworzyć w danym czasie - jeden element na dwie sekundy.

    Zarzucę zdanie o wynikach z flow 100k elementów

    Przy pracy z dużymi zbiorami danych, ten limit staje się problematyczny, ponieważ procesy tworzenia danych na SharePoint zaczynają trwać bardzo długo, dochodząc do kilkunastu godzin, w zależności od liczby dodawanych elementów. Ograniczenie to prowadzi do spadku wydajności rozwiązania, a także wydłuża czas potrzebny na zakończenie operacji.
  - **Rozwiązanie:** Aby poradzić sobie z tym ograniczeniem, zastosowano alternatywne rozwiązanie, takie jak wykorzystanie XDDD a ja się zastanawiałem co my bashujemy, przecież to nie linux. MY UŻYWAMY BATCHOWANIA!!!!!! *bashowania*[1] oraz *REST API*[2]. Dzięki tym technologiom udało się zwiększyć wydajność i szybkość tworzenia elementów na SharePoint. REST API pozwala na bardziej efektywne zarządzanie danymi w czasie rzeczywistym, a *bashowanie* umożliwa równoległe przetwarzanie dużych zbiorów danych, co znacznie skraca czas potrzebny na realizację operacji.

    btw. czy cos o tym nie powinno sie pojawic w implementacji?

- **Brak możliwości pisania standardowego kodu w Power Automate:**
  - **Problem:** Power Automate, choć jest narzędziem dedykowanym do automatyzacji procesów, ma pewne ograniczenia związane z brakiem wsparcia dla niestandardowego kodu. Oznacza to, że nie jest możliwe napisanie własnych funkcji czy rozszerzeń, które pozwalałyby na realizację bardziej złożonych operacji, które nie były dostępne w standardowych akcjach i konektorach Power Automate.
  - **Rozwiązanie:** W związku z tym brakiem elastyczności, rozwiązaniem mogłaby być integracja z Azure Functions lub Logic Apps, które oferują większą elastyczność i wsparcie dla niestandardowego kodu. Niestety, w tym projekcie to rozwiązanie nie mogło zostać wykorzystane, w związku z czym dostosowano się do dostępnych narzędzi i metod.

[1] Bashowanie - używanie powłoki Bash do automatyzacji zadań systemowych.
[2] REST API - interfejs umożliwiający komunikację między systemami za pomocą zapytań HTTP.

- **Ograniczenia dotyczące rozmiaru list SharePoint:**
  - **Problem:** SharePoint posiada limit dotyczący liczby elementów w jednej liście, co stwarza problemy z wydajnością przy pracy z bardzo dużymi zbiorami danych. Listy o zbyt dużych rozmiarach powodują znaczny spadek wydajności, a także prowadzą do problemów z synchronizacją danych między różnymi użytkownikami. Dodatkowo, duże listy skutkują wolniejszym ładowaniem danych i wykonywaniem zapytań.
  - tutaj ograniczenie rozmiaru listy nie ma się w cale do pobierania lokalnej kopii. Pierwsza sprawa że pobieramy lokalną kopie bo connector ssie i średnio działa kiedy używamy lookup/filter bezpośrednio na liście (tak jak było z listą wyboru serwisu). A druga sprawa że lokalne kopie są pobierane w kilku miejscach apki (dodałem to ostatnio), żeby zapewnić aktualne dane.
  - **Rozwiązanie:** Aby rozwiązać ten problem, zdecydowano się na optymalizację działania list SharePoint poprzez zaciąganie danych do Power Apps tylko raz (szczegóły implementacji wyjaśniono w rozdziale [4.3]), podczas uruchamiania aplikacji (przy użyciu funkcji *OnStart*). Następnie dane są aktualizowane tylko wtedy, gdy jest to wymagane, w przeciwieństwie do ciągłego łączenia się z SharePoint. Takie podejście pozwala na zmniejszenie liczby operacji i poprawę wydajności pracy z danymi, szczególnie przy dużych listach.

Rozdział 7

Zakończenie

Zakończenie pracy zwane również Uwagami końcowymi lub Podsumowaniem powinno zawierać ustosunkowanie się autora do zadań wskazanych we wstępie do pracy, a w szczególności do celu i zakresu pracy oraz porównanie ich z faktycznymi wynikami pracy. Podejście takie umożliwia jasne określenie stopnia realizacji założonych celów oraz zwrócenie uwagi na wyniki osiągnięte przez autora w ramach jego samodzielnej pracy.

Integralną częścią pracy są również dodatki, aneksy i załączniki zawierające stworzone w ramach pracy programy, aplikacje i projekty.

Literatura

Instytut Robotyki i Inteligencji Maszynowej

Politechnika Poznańska, Wydział Automatyki, Robotyki i Elektrotechniki

Skład dokumentu został wykonany przy użyciu systemu LaTeX, z wykorzystaniem narzędzi takich jak *LaTeX Workshop*, *MiKTeX*, *latexmk* oraz *Visual Studio Code*.