



---

# 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 Volkswa-  
gen 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, powie-  
lanie, 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 odpo-  
wiedzialność odszkodowawczą wobec spółki Volkswagen Poznań Sp. z o.o.

# Spis treści

<b>1</b>	<b>Wstęp</b>	<b>1</b>
<b>2</b>	<b>Podstawy teoretyczne</b>	<b>3</b>
2.1	Struktura procesu . . . . .	3
2.1.1	Gromadzenie danych dotyczących ofert usługodawców . . . . .	3
2.1.2	Przygotowanie danych . . . . .	3
2.1.3	Przebieg Iteracji . . . . .	4
2.2	Wykorzystane technologie . . . . .	5
2.2.1	Skrypty pakietu Office . . . . .	5
2.2.2	SharePoint . . . . .	5
2.2.3	Power Automate . . . . .	6
2.2.4	Power Apps . . . . .	6
<b>3</b>	<b>Architektura rozwiązania</b>	<b>8</b>
3.1	Założenia projektowe . . . . .	8
3.1.1	Systematyzacja danych . . . . .	8
3.1.2	Archiwizacja danych . . . . .	9
3.1.3	Interfejs przyjazny dla użytkownika . . . . .	9
3.1.4	Użycie pakietu Microsoft 365 . . . . .	9
3.1.5	Optymalizacja . . . . .	10
3.2	Koncepcja rozwiązania . . . . .	10
3.2.1	Baza danych . . . . .	10
3.2.2	Dodawanie informacji do bazy danych . . . . .	11
3.2.3	Interfejs procesu decyzyjnego . . . . .	12
3.2.4	Generowanie raportu . . . . .	13
<b>4</b>	<b>Implementacja</b>	<b>14</b>
4.1	Utworzenie bazy danych na platformie Sharepoint . . . . .	14
4.2	Ekran dodawania danych . . . . .	16
4.2.1	Zapis pliku w chmurze . . . . .	16
4.2.2	Skrypt pakietu Office . . . . .	18
4.2.3	Walidacja nazw kolumn . . . . .	20
4.2.4	Integracja z listami SharePoint . . . . .	20
4.2.5	Uzupełnianie numerów MPK . . . . .	25
4.3	Ekran startowy aplikacji i przygotowanie danych . . . . .	25
4.4	Edycja danych . . . . .	28
4.4.1	Ekran wyboru usługi do edycji . . . . .	28
4.4.2	Ekran edycji elementu . . . . .	29

4.4.3	Listing kodu . . . . .	31
4.5	Generowanie raportu . . . . .	32
4.5.1	Łączenie danych z list . . . . .	32
4.5.2	Przekazywanie danych do Power Automate . . . . .	33
4.5.3	Generowanie raportu w Power Automate . . . . .	34
4.6	Składniki . . . . .	36
4.6.1	Nagłówek ekranu . . . . .	36
4.6.2	Indykator ładowania . . . . .	36
4.7	Samouczek . . . . .	37
4.7.1	Nawigacja po samouczku . . . . .	37
4.7.2	Prezentacja zawartości w samouczku . . . . .	38
<b>5</b>	<b>Prezentacja działania</b>	<b>39</b>
5.1	Ekran startowy . . . . .	39
5.2	Ekran dodawania danych . . . . .	40
5.2.1	Formularz walidacji kolumn . . . . .	41
5.2.2	Formularz uzupełniania numerów MPK . . . . .	43
5.3	Ekran wypełniania formularza . . . . .	44
5.3.1	Ekran nawigacyjny . . . . .	44
5.3.2	Ekran szczegółowy . . . . .	45
5.3.3	Ekran generowania raportu . . . . .	47
5.3.4	Ekran samouczka . . . . .	48
<b>6</b>	<b>Napotkane problemy i rozwiązania</b>	<b>49</b>
<b>7</b>	<b>Zakończenie</b>	<b>52</b>
	<b>Literatura</b>	<b>53</b>

# 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<sup>1</sup> na najbliższy rok kalendarzowy. Praca została wykonana z wykorzystaniem *Power Platform* oraz *SharePoint*, które są integralną częścią 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.

Struktura pracy jest następująca:

W rozdziale 2 przedstawiono dotychczasowy przebieg procesu oraz opisano wykorzystane narzędzia. Rozdział 3 jest poświęcony architekturze rozwiązania. Omówiono w nim założenia projektowe, których należało się trzymać. Ponadto zawiera on pierwotną koncepcję rozwiązania, która opisuje

---

<sup>1</sup> Usługi IT należy rozumieć jako licencje oraz klucze dostępu do używanych systemów informatycznych.

jakie funkcjonalności powinny zostać zaimplementowane.

Rozdział 4 zawiera końcową implementację poszczególnych komponentów. Opisano w nim każdy element aplikacji oraz omówiono integrację między poszczególnymi komponentami.

Rozdział 5 stanowi przedstawienie działania systemu. Zawiera on opis przebiegu procesu z wykorzystaniem stworzonego rozwiązania.

W rozdziale 6 opisano problemy napotkane podczas implementacji oraz ich rozwiązania, zastosowane wprzty implementacji.

Rozdział 7 stanowi podsumowanie pracy. Przedstawiono w nim wnioski oraz możliwości dalszego rozwoju aplikacji.

Michał Gajdzis w ramach niniejszej pracy zaimplementował:

- strukturę bazy danych SharePoint,
- przepływ pozwalający na zapis i formatowanie pliku,
- mechanizm zmiany nazw kolumn w arkuszu,
- mechanizm zapisu danych wprowadzanych przez użytkownika w bazie danych SharePoint,
- ekran generowania raportu wraz z potrzebnymi przepływami.

Remigiusz Wolniak wykonał:

- przepływ służący do importu danych z pliku do bazy danych,
- skrypt formatujący plik,
- interfejs oraz logikę ekranów: głównego, dodawania danych od systemu, edycji danych oraz samouczka,
- mechanizmy zarządzania danymi wewnątrz aplikacji.

## Rozdział 2

# Podstawy teoretyczne

### 2.1 Struktura procesu

R. WOLNIAK

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*<sup>1</sup>, 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 strukturze kolumn podobnej do tabeli 2.1. Brakuje w nim jednak informacji kluczowych do rozpoczęcia cyklu. Dlatego pierwszym krokiem

---

<sup>1</sup>*indykacja* – wstępne głosowanie



jest przygotowanie danych przez osobę nadzorującą proces ze strony oddziału 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* to opinia wyrażona przez menedżera usługi, natomiast *Komentarz K-DES* stanowi odpowiedź międzynarodowego zarządu firmy, który odpowiada za kształtowanie strategii IT.

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

M. GAJDZIS

Aby usprawnić przebieg procesu, zabezpieczyć 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<sup>2</sup> (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 poszczególnych zasobów witryny co pozwala na skuteczne zabezpieczenie wrażliwych informacji.

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.

---

<sup>2</sup> *Konektor* (ang. *connector*) — moduł umożliwiający integrację aplikacji z usługami lub źródłami danych w celu wymiany informacji i synchronizacji systemów.

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.

Ekran 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*<sup>3</sup> dla przycisku.

---

<sup>3</sup>OnSelect – określa akcje, które zostaną wykonane po naciśnięciu elementu



RYSUNEK 2.1: Edytor Power Apps [1]

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 elementy 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

M. GAJDZIS

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 przeźreni 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<sup>1</sup> 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 wariantcie, 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.

---

<sup>1</sup>Platforma Power (ang. *Power Platform*) – składowa pakietu Microsoft 365, w skład której wchodzi 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

R. WOLNIAK

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, niezmiennie 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.

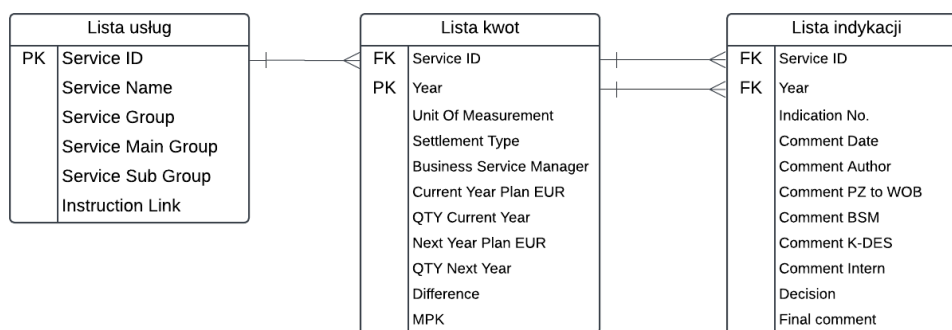
#### Atrybuty danych

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<sup>2</sup> 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. Utworzone zostaną mechanizmy wyszukiwania, które 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 prześle 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.

---

<sup>2</sup>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

R. WOLNIAK

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 skonfigurować.

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*  $\rightarrow$  1,
- *Not Accepted*  $\rightarrow$  -1,
- *No Status*  $\rightarrow$  0.

## 4.2 Ekran dodawania danych

R. WOLNIAK

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<sup>1</sup> *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*).

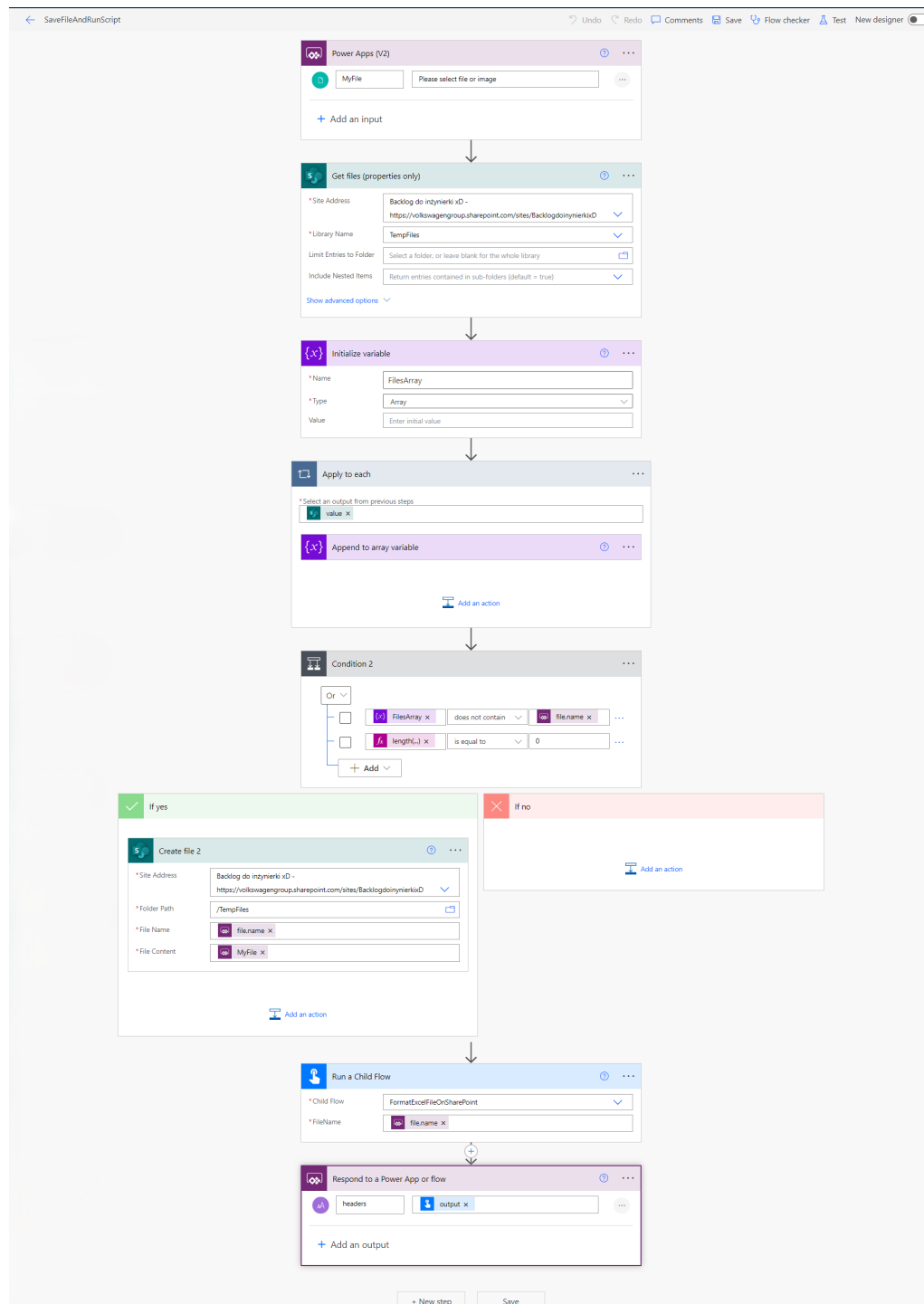
Pod kontrolką *Attachment Control* dodano przycisk *Save attachment*. Jego naciśnięcie skutkuje wywołaniem szeregu funkcji opisanych we właściwości *OnSelect*:

- sprawdzenie 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 nazywany *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.

<sup>1</sup>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*<sup>2</sup>. 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,

---

<sup>2</sup>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 kontroli *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:

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

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:

$$JW = J + l \cdot p \cdot (1 - J), \quad (4.2)$$

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*.

### 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*<sup>3</sup>.
- Prawa kolumna zawiera kontrolkę *ComboBox*<sup>4</sup>, 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, zawierającą 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*<sup>5</sup> 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 kontrolki *Dropdown*. Wybrane wartości są następnie wykorzystywane podczas importu danych do odpowiednich list, co odbywa się za pomocą przycisku *Upload data*.

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*

<sup>3</sup>*Label* – kontrolka tekstowa umożliwiająca wyświetlanie statycznych wartości.

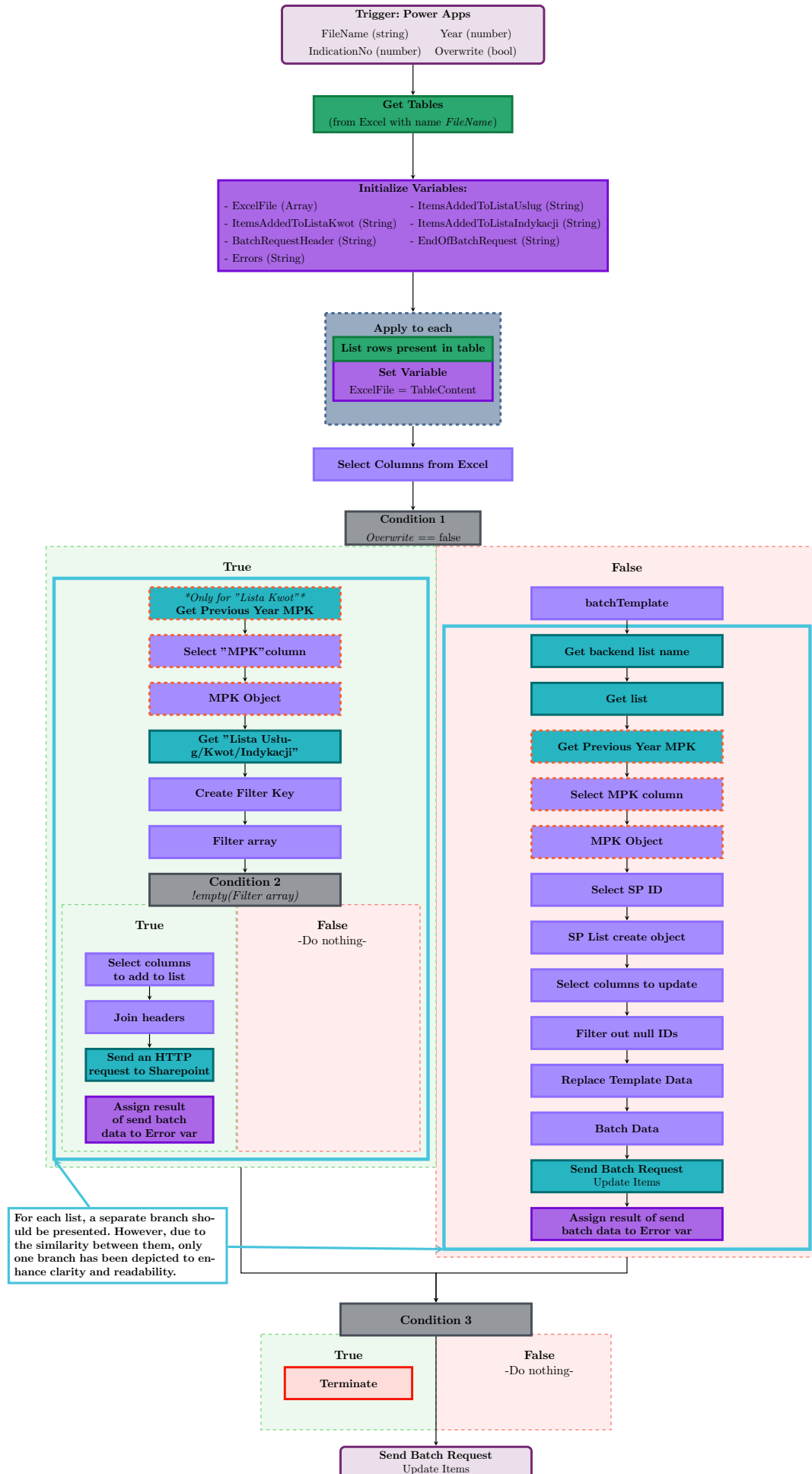
<sup>4</sup>*ComboBox* – rozwijana lista z możliwością wprowadzania tekstu

<sup>5</sup>*Dropdown* – kontrolka umożliwiająca wybór jednej z dostępnych wartości z rozwijanej listy, bez możliwości edycji.

*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 operację. 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 bloki 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 gałęzi 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 wcisnie przycisk w aplikacji. Jako parametry wejściowe przyjmuje nazwę pliku, rok i numer indykacji jakie mają być przypisane 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,
  - *ItemsAddedToListaUslug/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. Przepływ 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<sup>6</sup> (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

R. WOLNIAK

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.

```

1      // Ustawienie zmiennej varDownloadingData na true , aby wskazać, że
      trwa proces pobierania danych
2  Set(varDownloadingData, true );
3
4  // Tworzenie kolekcji colYears, która zawiera pięć lat: dwa lata wstecz,
      bieżący rok oraz dwa lata do przodu
5  ClearCollect(colYears,
6      {Value: Text(Now(), "yyyy") - 2},
7      {Value: Text(Now(), "yyyy") - 1},
8      {Value: Text(Now(), "yyyy") + 0},
9      {Value: Text(Now(), "yyyy") + 1},

```

<sup>6</sup>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.

```

10     {Value: Text(Now(), "yyyy") + 2}
11 );
12
13 // Ustawienie zmiennej VWBlue na kolor o wartości heksadecymalnej #002e5f
14 Set(VWBlue, ColorValue("#002e5f"));
15
16 // Tworzenie kolekcji colNumbers, która zawiera liczby od 1 do 5
17 ClearCollect(colNumbers,
18     {Value: 1}, {Value: 2}, {Value: 3}, {Value: 4}, {Value: 5}
19 );
20
21 // Pobranie profilu użytkownika z usługi Office 365 i przypisanie go do
    zmiennej UserVar
22 Set(UserVar, UżytkownicyusługiOffice365.MyProfile());
23
24 // 1. Pobieranie danych z Lista_Uslug w partiach
25 Clear(LocalServiceData); // Wyczyszczenie kolekcji LocalServiceData przed
    rozpoczęciem pobierania
26 ForAll(Sequence(Round((First(Sort(Lista_Uslug, 'Identyfikator (ID)',
    SortOrder.Descending)).'Identyfikator (ID)' - First(Sort(Lista_Uslug, '
    Identyfikator (ID)', SortOrder.Ascending)).'Identyfikator (ID)') / 2000
    + 1, 0), 1, 1),
27     With({
28         _firstID: First(Sort(Lista_Uslug, 'Identyfikator (ID)', SortOrder.
            Ascending)).'Identyfikator (ID)' + (ThisRecord.Value - 1) * 200
            0,
29         _lastID: First(Sort(Lista_Uslug, 'Identyfikator (ID)', SortOrder.
            Ascending)).'Identyfikator (ID)' + ThisRecord.Value * 2000
30     },
31     Collect(LocalServiceData, Filter(Lista_Uslug, 'Identyfikator (ID)' >=
        _firstID && 'Identyfikator (ID)' < _lastID)))
32 );
33
34 // 2. Pobieranie danych z Lista_Kwot w partiach
35 Clear(LocalCostData); // Wyczyszczenie kolekcji LocalCostData przed rozpocz
    ęciem pobierania
36 ForAll(Sequence(Round((First(Sort(Lista_Kwot, ID, SortOrder.Descending)).ID
    - First(Sort(Lista_Kwot, ID, SortOrder.Ascending)).ID) / 2000 + 1, 0),
    1, 1),
37     With({
38         _firstID: First(Sort(Lista_Kwot, ID, SortOrder.Ascending)).ID + (
            ThisRecord.Value - 1) * 2000,
39         _lastID: First(Sort(Lista_Kwot, ID, SortOrder.Ascending)).ID +
            ThisRecord.Value * 2000
40     },
41     Collect(LocalCostData, Filter(Lista_Kwot, ID >= _firstID && ID <
        _lastID)))
42 );
43
44 // 3. Pobieranie danych z Lista_Indykacji w partiach
45 Clear(LocalIndicationsData); // Wyczyszczenie kolekcji LocalIndicationsData

```

```

    przed rozpoczęciem pobierania
46  ForAll(Sequence(Round((First(Sort(Lista_Indykacji, ID, SortOrder.Descending
    ))).ID - First(Sort(Lista_Indykacji, ID, SortOrder.Ascending)).ID) / 200
    0 + 1, 0), 1, 1),
47      With({
48          _firstID: First(Sort(Lista_Indykacji, ID, SortOrder.Ascending)).ID
              + (ThisRecord.Value - 1) * 2000,
49          _lastID: First(Sort(Lista_Indykacji, ID, SortOrder.Ascending)).ID +
              ThisRecord.Value * 2000
50      },
51      Collect(LocalIndicationsData, Filter(Lista_Indykacji, ID >= _firstID &&
              ID < _lastID)))
52 );
53
54 // 4. Łączenie danych z Lista_Uslug, Lista_Kwot i Lista_Indykacji
55 ClearCollect(MergedData,
56     AddColumns(LocalServiceData As ServiceRecord,
57         Kwoty,
58         With({
59             MaxYearCostRecord: First(Sort(Filter(LocalCostData, Service_ID
              = ServiceRecord.Service_ID), Year, SortOrder.Descending))
60         },
61         If(IsBlank(MaxYearCostRecord), Blank(), LookUp(LocalCostData,
              Service_ID = ServiceRecord.Service_ID && Year =
              MaxYearCostRecord.Year))),
62         Indykacje,
63         With({
64             MaxYearRecord: First(Sort(Filter(LocalIndicationsData,
              Service_ID = ServiceRecord.Service_ID), Year, SortOrder.
              Descending))
65         },
66         If(IsBlank(MaxYearRecord), Blank(), LookUp(LocalIndicationsData,
              Service_ID = ServiceRecord.Service_ID && Year = MaxYearRecord.
              Year && IndicationNo = MaxYearRecord.IndicationNo)))
67     )
68 );
69
70 // Ustawienie zmiennej varDownloadingData na false, aby wskazać, że proces
    pobierania danych został zakończony
71 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 lokalne kopie trzech list danych: *Lista\_Uslug*, *Lista\_Kwot* oraz *Lista\_Indykacji*, wykorzystując funkcję *ClearCollect()*. Dane są pobierane w partiach po 2000 rekordów, aby zoptymalizować wydajność i uniknąć przekroczenia limitów pamięciowych. Proces ten polega na iteracyjnym przetwarzaniu danych, gdzie dla każdej listy określany jest zakres identyfikatorów (ID) pochodzących z listy Sharepoint dla kolejnych partii, zaczynając od najmniejszego ID i zwiększając go o 2000. Każda partia jest filtrowana według tego zakresu i dodawana do odpowiedniej kolekcji lokalnej. Takie podejście pozwala na szybsze przetwarzanie danych i uniknięcie problemów przy pobieraniu dużych zestawów danych.

Kolejnym krokiem jest utworzenie kolekcji *MergedData*, która łączy dane z trzech lokalnych kopii list. W tym celu zastosowano funkcję *AddColumns()*, aby dodać dwie nowe kolumny: *Kwoty* oraz *Indykacje*. Funkcja *With()* pozwala na uproszczenie złożonych obliczeń poprzez przypisanie wyników pośrednich do zmiennych, co zwiększa czytelność kodu. Dane w kolumnach są wyodrębniane przy użyciu funkcji *Lookup()* oraz *Filter()*, które umożliwiają precyzyjne filtrowanie i wyszukiwanie rekordów na podstawie trzech kluczowych kryteriów: *Service\_ID*, *Year* oraz *IndicationNo*.

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

M. GAJDZIS

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 identyfikatora (*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 jaką 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*).

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 wyświetlenie kalendarza,
- **Autor** — pole *ComboBox*, umożliwia wybór autora z listy pracowników. Lista ta pobierana jest poprzez connector *Office365Users*<sup>7</sup>, który zawiera informacje na temat wszystkich użytkowników infrastruktury firmy.

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.

---

<sup>7</sup> *Office365Users* – konektor pozwalający na dostęp do listy zawierającej informacje na temat użytkowników takie jak imię, nazwisko, dane kontaktowe lub dział.

- 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.
29                         DisplayName;
30                     Comment_PZ_to_WOB: CommentToWOBInput.Value;
31                     Comment_Intern: InternalCommentInput.Value;
32                     Decision: Switch(
33                         DropdownCanvas1.Selected.Value;
34                         "Accepted"; 1;
35                         "No Status"; 0;
36                         "Not Accepted"; -1;
37                         0
38                     )
39                 }

```

```

39         )
40     );;
41     Notify("A record has been successfully updated!";
42           NotificationType.Success);
43
44     // Jeśli rekord nie istnieje; utwórz nowy
45     Set(
46         TempOutput;
47         Patch(
48             Lista_Indykacji;
49             Defaults(Lista_Indykacji);
50             {
51                 Service_ID: ChosenServiceID;
52                 Year: Year_Dropdown.Selected.Value;
53                 IndicationNo: 1;
54                 Comment_date: Text(CommentDateInput.SelectedDate);
55                 Comment_author: ComboboxCanvas1.Selected.
56                     DisplayName;
57                 Comment_PZ_to_WOB: CommentToWOBInput.Value;
58                 Comment_Intern: InternalCommentInput.Value;
59                 Decision: Switch(
60                     DropdownCanvas1.Selected.Value;
61                     "Accepted"; 1;
62                     "No Status"; 0;
63                     "Not Accepted"; -1;
64                     0
65                 )
66             }
67         )
68     );;
69     Notify("The record has been successfully created!";
70           NotificationType.Success)
71 )
72 );;

```

LISTING 4.2: Kod zapisywania danych w formularzu edycji

## 4.5 Generowanie raportu

M. GAJDZIS

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 *Drop-down* po lewej stronie ekranu. Ich uzupełnienie skutkuje wygenerowaniem podglądu danych, który

pozwała na walidację ich poprawności. Listing 4.3 przedstawia kod odpowiedzialny za utworzenie kolekcji *CombinedData*, która zawiera odpowiednio odfiltrowane informacje.

```

1      ClearCollect(
2      CombinedData;
3      AddColumns(
4          LocalServiceData;
5          MPK;
6          Lookup(
7              LocalCostData;
8              Service_ID = LocalServiceData[@Service_ID] && Year =
                Year_Dropdown_1.Selected.Value;
9              MPK
10         );
11         Comment_PZ_to_WOB;
12         Lookup(
13             LocalIndicationsData;
14             Service_ID = LocalServiceData[@Service_ID] && Year =
                Year_Dropdown_1.Selected.Value && IndicationNo =
                IndicationNo_Dropdown_1.Selected.Value;
15             Comment_PZ_to_WOB);
16     [... ]
17 );;
```

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<sup>8</sup>, wykorzystując połączone informacje z trzech źródeł zawarte w kolekcji *CombinedData*. Kod wywołujący funkcję *GenerateReport* przedstawiono w listingu 4.4.

```

1      GenerateReport.Run(
2          Substitute(
3              "[" &
4              Concat(
5                  CombinedData;
6                  "{"Service_ID":"" & Service_ID & """," &
7                  ""Service_Name":"" & Service_Name & """," &
8                  ""MPK":"" & MPK & """," &
9                  ""Comment_PZ_to_WOB":"" & Comment_PZ_to_WOB & """," &
```

<sup>8</sup>JSON - format wymiany danych, który jest oparty na strukturze tekstowej

```

10         """Decision":"""" & Decision & """, "
11     ) &
12     "]" ,
13     "},]" ;
14     "}] "
15 ) ,
16 IndicationNoCollect.Value ,
17 YearNoCollect.Value
18 )

```

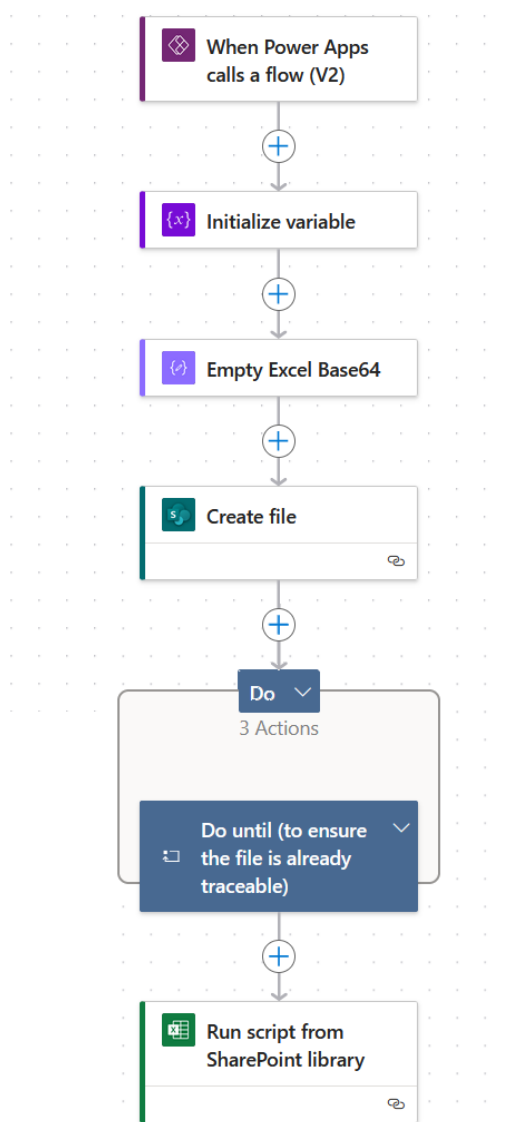
LISTING 4.4: Kod wywołujący funkcję GenerateReport

### 4.5.3 Generowanie raportu w Power Automate

*Flow* o nazwie *GenerateReport* 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 danych, 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

R. WOLNIAK

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. Poniżej 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()*<sup>9</sup>.
- 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 zmniejszenie 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 informująca 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*.

<sup>9</sup> Atrybut *MyProfile* pozwala na dostęp do informacji o bieżącym użytkowniku.



RYSUNEK 4.5: Kontrolka ładowania

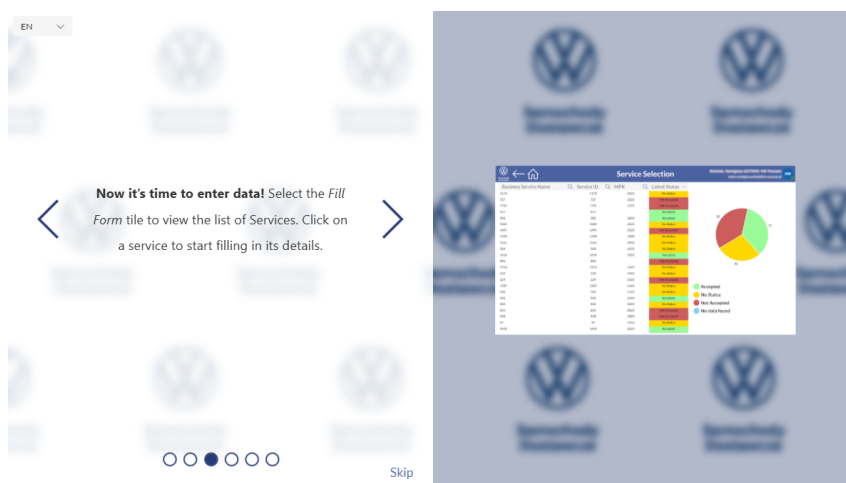
## 4.7 Samouczek

R. WOLNIAK

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 multimediiów z prawej. Na widok całości składają się dodatkowo ikony strzałek, umożliwiające nawigację po kolekcji tekstów oraz nawigatora, prezentującego aktualny postęp prezentacji samouczka.

Obraz 5.12 przedstawia przykładowy widok działania samouczka.



RYSUNEK 4.6: Prezentacja widoku samouczka

### 4.7.1 Nawigacja po samouczku

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

LISTING 4.5: Kod wywołany 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 postępującą nawigacją. Każdy z etapów samouczka zawiera odpowiedni obrazek, który wyświetlany jest razem z przypisanym do niego tekstem.

## Rozdział 5

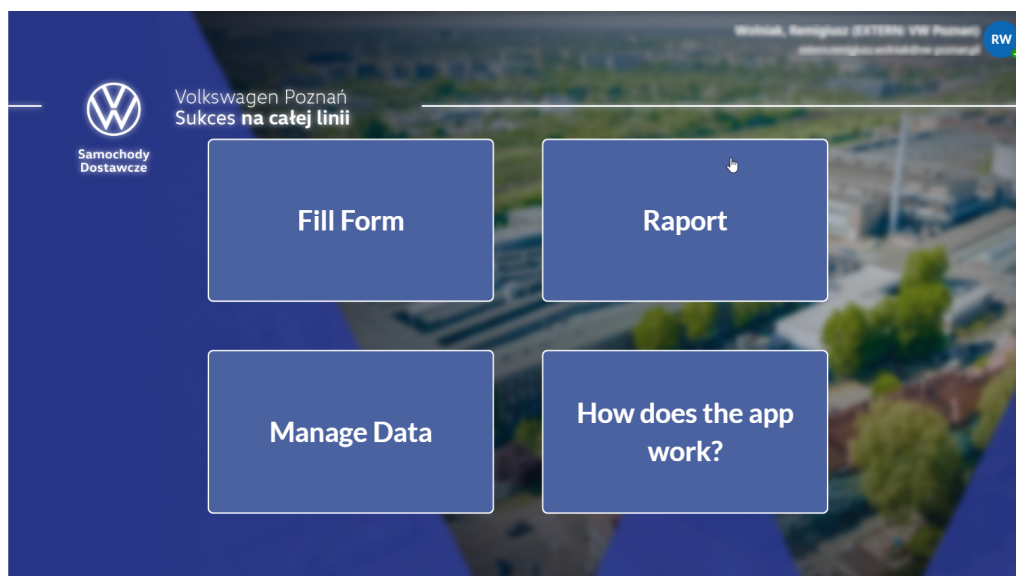
# Prezentacja działania

R. WOLNIAK

W niniejszym 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 czterech 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 the 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.

The screenshot shows the 'Ekran dodawania danych' (Data Addition Screen) in the VW system. The interface is divided into three main sections:

**Top Section: File Upload and Preview**

- Add attachments to process:** A box with 'Nothing is attached.' and an 'Attach file' button.
- Choose file to preview:** A box showing a preview of 'Fake\_data\_2022' (23.01.2025 11:22) with an 'X' icon to close it.
- Save attachments:** A button at the bottom of the first box.
- Click to open:** A button labeled 'Click to open: Fake\_data\_2022.xlsx in new tab.'

**Middle Section: Column Matching**

Current incorrect column names:	Choose correct name for each column:
Service group	Service group
Service main group	Service main group
Service sub group	Service sub group
Business service	Business service
Service ID	Service ID
Business Service Manager	Business Service Manager
Unit of Measurement	Unit of Measurement
EUR w KVA	Select correct column name
QTY	Select correct column name
EUR w KVA1	Select correct column name
QTY1	Select correct column name
MPK	MPK
Status	Select correct column name
Kommentar BSM	Select correct column name
Kommentar K-DES	Comment K-DES
Comment intern	Comment Intern

**Right Side: Column Matching Instructions**

**Column Matching Instructions**  
Select the pre-uploaded Excel file from the top list.  
If you don't make changes just press Save button.

**Update column names** (button)

Choose year: 2025 | Choose indication No.: 1

**Upload data** (button)

**Bottom Section: Assign/Edit MPK number**

Assign/Edit MPK number: Filter by Name / ID / MPK | Show filled fields (checkbox)

Service Name:	Service ID:	MPK:
Service 35	32	
Service 37	804	
Service 38	551	
Service 42	836	
Service 50	696	
Service 62	564	
Service 73	1513	
Service 83	1602	
Service 89	247	
Service 90	1909	
Service 93	336	
Service 111	1221	
Service 113	780	
Service 115	1172	
Service 143	1969	
Service 150	150	
Service 154	795	

**Bottom Right: Instructions**

This step is required when some data does not have a 'Cost center number' assigned to it.  
Remaining MPK to fill: 94.  
Data is saved automatically.

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

**Add attachments to process:**

Nothing is attached.

Attach file

**Choose file to preview:**

Fake\_data\_2022  
23.01.2025 11:22

Save attachments

Click to open: Fake\_data\_2022.xlsx in new tab.

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 biblioteki SharePoint, należy nacisnąć przycisk *Save attachments* znajdujący się poniżej. 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 przycisk oznaczony krzyżykiem przy każdym pliku pozwala na usunięcie go z systemu – w tym z folderu tymczasowego.

Ostatnim elementem tej sekcji jest przycisk *Click 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

Current incorrect column names:	Choose correct name for each column:
Service group	Service group
Service main group	Service main group
Service sub group	Service sub group
Business service	Business service
Service ID	Service ID
Business Service Manager	Business Service Manager
Unit of Measurement	Unit of Measurement
EUR w KVA	Select correct column name
QTY	Select correct column name
EUR w KVA1	Select correct column name
QTY1	Select correct column name
MPK	MPK
Status	Select correct column name

**Column Matching Instructions**

Select the pre-uploaded Excel file from the top list.

If you don't make changes just press Save button.

Update column names

Choose year: 2025 Choose indication No.: 1

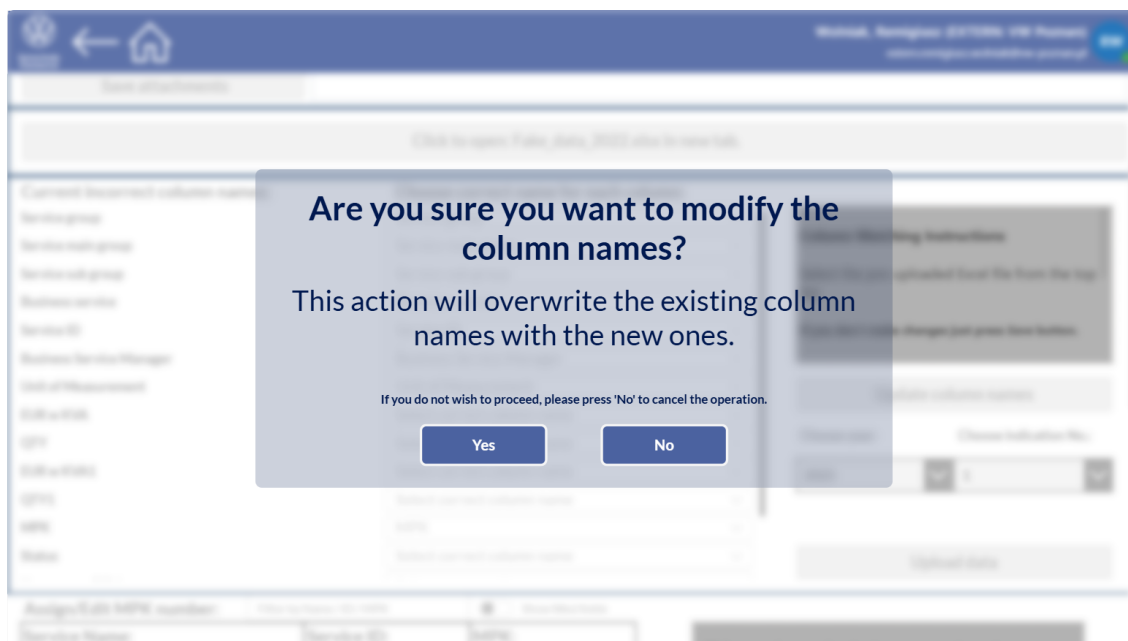
Upload data

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 walidację. 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 się 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. Dostosowanie poprawnych nazw kolumn jest bardzo ważne dla dalszego działania aplikacji, dlatego zaleca się dokładne sprawdzenie poprawności wybranych nazw. Zatwierdzenie zmiany, skutkuje nadpisaniem nagłówków, o



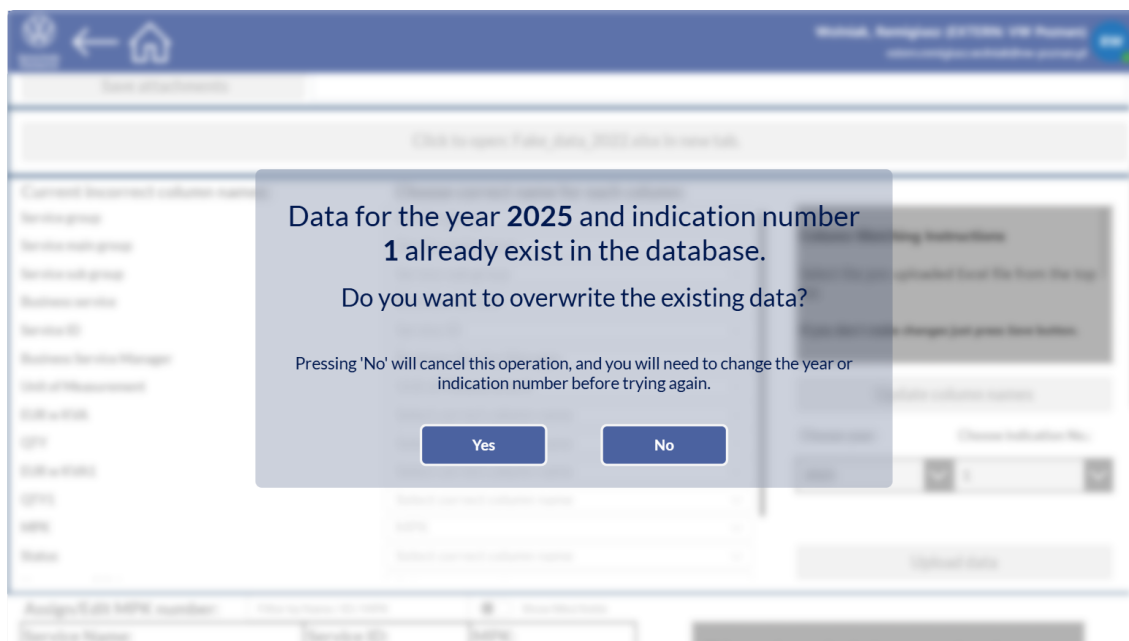
RYСУNEK 5.5: Zapytanie o poprawność nazw kolumn

czym informuje odpowiedni indyktor. 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 zostały 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 po 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 zbiorowych, 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

Assign/Edit MPK number:
Filter by Name / ID / MPK
Show filled fields

Service Name:	Service ID:	MPK:
Service 35	32	
Service 37	804	
Service 38	551	
Service 42	836	
Service 50	696	
Service 62	564	
Service 73	1513	
Service 83	1602	
Service 89	247	
Service 90	1909	
Service 93	336	
Service 111	1221	
Service 113	780	
Service 115	1172	
Service 143	1969	
Service 150	150	
Service 154	795	

This step is required when some data does not have a 'Cost center number' assigned to it.  
Remaining MPK to fill: **94**.  
Data is saved automatically.

RYSUNEK 5.7: Formularz wypełniania/edycji numerów MPK

Ostatnią sekcją tego ekranu jest 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 automatycznie zapisywane.

Nad galerią umieszczono pole tekstowe, które pozwala na filtrowanie wyników względem każdej z kolumn.

Obok znajduje się przełącznik, który pozwala na przełączenie się między 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.

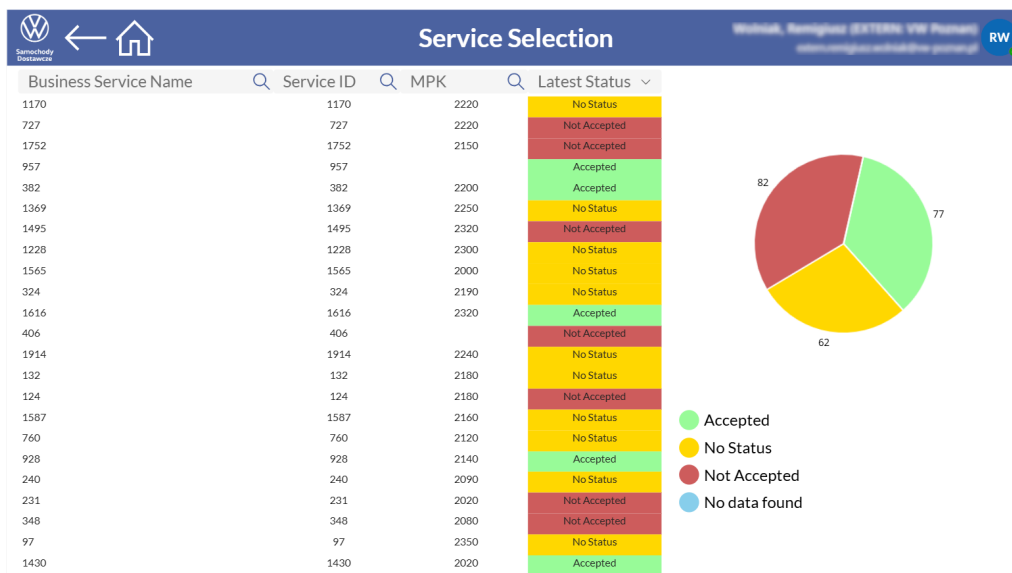
Po prawej stronie znajduje się kolejne pole informacyjne, które zawiera instrukcję dla użytkownika oraz liczbę serwisów, które nie są przypisane do żadnego obszaru.

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 się 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 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.

### 5.3.2 Ekran szczegółowy

[illegible]

RYSUNEK 5.9: Ekran edycji elementu

Rysunek 5.9 przedstawia interfejs ekranu szczegółowego dla wybranej usługi. Pierwszym nie-standardowym elementem jest nagłówek, zawierający najważniejsze informacje takie jak: nazwa, identyfikator, numer MPK czy dane kontaktowe do osoby odpowiedzialnej za serwis.

## Zestawienie danych rocznych

Poniżej znajduje się galeria zawierająca zestawienie informacji rocznych na przestrzeni trzech poprzednich lat. Zawiera on:

- **Rok (*Year*)** — użytkownik może wybrać rok, którego dotyczy wpis.
- **Jednostka miary (*Unit of Measurement*)** — określa jednostkę, w której będą wprowadzane dane (np. sztuki, kilogramy).
- **Sposób rozliczenia (*Settlement Type*)** — pole wyboru umożliwiające określenie sposobu rozliczenia, np. planowane ilości.

- **Cena w roku obecnym (*Current Year Plan*)** — przewidywana cena jednostkowa dla bieżącego roku.
- **Liczba w roku obecnym (*Current Year QTY*)** — planowana liczba jednostek w bieżącym roku.
- **Cena w roku następnym (*Next Year Plan*)** — przewidywana cena jednostkowa dla następnego roku.
- **Liczba w roku następnym (*Next Year QTY*)** — planowana liczba jednostek w następnym roku.
- **Różnica cen (*Difference*)** — różnica pomiędzy ceną w roku obecnym a ceną w roku następnym.
- **Decyzja końcowa (*Final Decision*)** — ostateczna decyzja dotycząca danego wpisu, wybierana z listy rozwijanej.

Galeria ta daje możliwość analizy podjętych decyzji względem cen w poszczególnych latach. Jest ona interaktywna – kliknięcie w wiersz powoduje wyświetlenie danych szczegółowych w tabeli poniżej.

#### Zestawienie indykacji

Druga galeria zawiera:

- **Rok (*Year*)** — użytkownik może wybrać rok, którego dotyczy wpis. Domyślnie pole to jest ustawiane na bieżący rok.
- **Numer indykacji (*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.
- **Komentarze** — pola do wprowadzenia uwag wewnętrznych, komentarzy między działami oraz końcowych komentarzy.
- **Status decyzji (*Decision*)** — lista rozwijana, umożliwiająca wybór odpowiedniego statusu (*Accepted*, *Not Accepted*, *No Status*). Domyślnie status jest ustawiany na podstawie decyzji z poprzedniego roku.
- **Data i autor** — data oraz osoba odpowiedzialna za wprowadzenie wpisu. Pole autora jest automatycznie uzupełniane danymi aktualnie zalogowanego użytkownika.

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

Z uwagi na utrzymanie przejrzystości interfejsu, w komórkach tabeli wyświetlane są skrócone komentarze. Użytkownik ma możliwość podglądu pełnej treści poprzez naciśnięcie komórki – wtedy wyświetlone zostaje okno dialogowe z pełnym tekstem, autorem oraz datą dodania (rysunek 5.10).

**Information from individual indications for 2025:**

Indication Number	Internal Comment	Comment PZ to WOB	Comment BSM	Comment K-DES	Final Comment	Decision
1	The text examples are varied. Test with longer...		Another simple text format	Keep things consistent here		Accepted

Autor: Wolniak, Remigiusz (EXTERN: VW Poznan)

Data: 23.01.2025

**Comment:**

The text examples are varied. Test with longer text to show comment overview.

RYSUNEK 5.10: Okno dialogowe z pełnym tekstem komentarza

### Formularz dodawania danych

Ostatnią sekcją tego ekranu jest formularz umożliwiający dodanie nowego rekordu do listy indykacji. Zawiera on następujące pola:

- **Listy wyboru roku i numeru indykacji,**
- **Lista wyboru autora komentarzy/decyzji** – na liście znajdują się wszyscy dostępni użytkownicy,
- **Lista wyboru statusu decyzji** – dostępne są trzy statusy: *Accepted*, *Not Accepted*, *No Status*,
- **Pola wprowadzania komentarzy** – każde z dostępnych pól pozwala na wprowadzenie wiadomości, która trafi do odpowiedniej jednostki.

Na końcu znajduje się przycisk pozwalający na zapisanie zmian w bazie danych oraz odświeżający dane pobrane przez aplikację.

#### 5.3.3 Ekran generowania raportu

Service_Name	Service_ID	MPK	Comment_PZ_to...	Decision_2
Service 1	1170	2220		
Service 2	727	2220		
Service 3	1752	2150		
Service 4	957			
Service 5	382	2200		
Service 6	1369	2250		
Service 7	1495	2320		
Service 8	1228	2300		
Service 9	1565	2000		
Service 10	324	2190		
Service 11	1616	2320		
Service 12	406			
Service 13	1914	2240		

Wiersze: 221

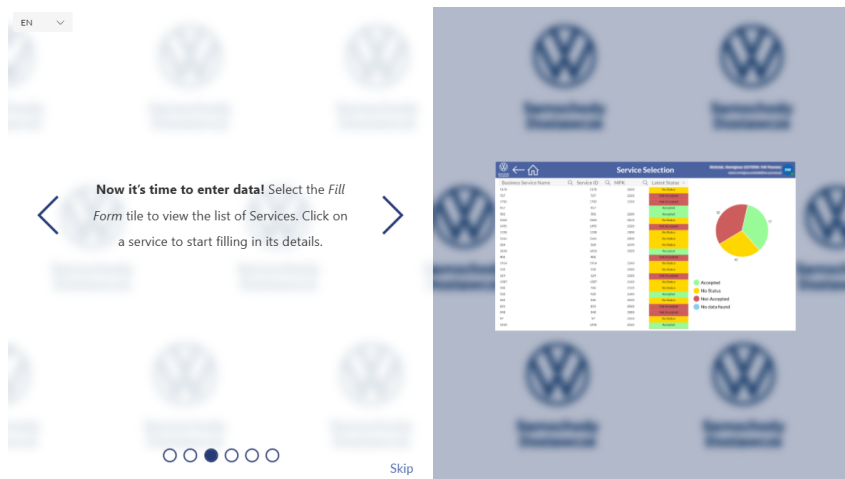
RYSUNEK 5.11: Ekran generowania raportu

Ostatnim ekranem, kluczowym do działania procesu jest ekran generowania raportu. Na rysunku 5.11 przedstawiono interfejs tego ekranu. Jest on bardzo prosty, ponieważ większość operacji została zautomatyzowana. Zawiera on dwa rozwijane pola, które umożliwiają wybór roku oraz numeru indykacji, dla których ma zostać wygenerowany raport. Po ich zmianie, system przetwarza

dane według podanych kryteriów i generuje podgląd w prawej części ekranu w celu walidacji poprawności.

Kiedy użytkownik potwierdzi poprawność, może nacisnąć przycisk *Generate file*, który uruchamia proces generowania raportu. Powstały plik zostaje zapisany w folderze SharePoint, a użytkownik zostaje poinformowany o pomyślnym zakończeniu operacji.

#### 5.3.4 Ekran samouczka



RYSUNEK 5.12: Prezentacja widoku samouczka *GenerateReport*

Jest to dodatkowy ekran, zaimplementowany aby pomóc nowym użytkownikom w opanowaniu działania systemu. Jest on podzielony na dwie sekcje. Pierwsza z nich zawiera krótki opis każdego z ekranów aplikacji oraz przyciski nawigacji. Druga część przedstawia przykładowe grafiki w celu lepszego zrozumienia działania. Samouczek podzielony został na kilka kroków a nawigacja po nich odbywa się poprzez strzałki znajdujące się po obu stronach opisu ekranu. Na dole ekranu znajdują się kropki, przedstawiające ilość kroków samouczka oraz postęp. Ekran ten jest dostępny w dwóch językach – Polskim oraz Angielskim.

## Rozdział 6

# Napotkane problemy i rozwiązania

M. GAJDZIS

### Problemy i ich rozwiązania

- **Relacje między listami w SharePoint:**

- **Problem:** Sharepoint 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 funkcje *lookup* oraz *filter*, umożliwiające tworzenie powiązań między trzema listami. Funkcja *lookup* znajduje pierwszy pasujący wiersz, natomiast *filter* zwraca listę pasujących elementów. Dodatkowo, stosowano zagnieżdżenia tych funkcji. 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.

- **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. 4.2.1.

- **Ustawienia lokalne - średniki i przecinki:**

- **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 określonym czasie. W ramach testu stworzono przepływ składający się z pętli, która wykonała się 100 tysięcy razy, a w każdej iteracji tworzonego elementu na SharePoint o losowej nazwie. Proces ten zajął niecałe 18 godzin, co daje średnio około 1,5 elementu na sekundę. Jest to nieefektywne przy dużych zestawach danych, ponieważ procesy tworzenia danych na SharePoint mogą trwać bardzo długo, dochodząc do kilkunastu godzin w zależności od liczby dodawanych elementów. To ograniczenie wpływa na wydajność rozwiązania i wydłuża czas potrzebny na zakończenie operacji.
- **Rozwiązanie:** Aby poradzić sobie z tym ograniczeniem, zastosowano alternatywne rozwiązanie, takie jak wykorzystanie *batchowania*<sup>1</sup> oraz *REST API*<sup>2</sup>. Dzięki tym technologiom udało się zwiększyć wydajność i szybkość tworzenia elementów na SharePoincie. REST API pozwala na bardziej efektywne zarządzanie danymi w czasie rzeczywistym, a *batchowanie* umożliwia równoległe przetwarzanie dużych zbiorów danych, co znacznie skraca czas potrzebny na realizację operacji.

- **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 pozwoliłyby 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.

- **Ograniczenia dotyczące rozmiaru list SharePoint:**

- **Problem:** Praca na oryginalnej liście SharePoint z dużą liczbą elementów powoduje spadek wydajności aplikacji. Bezpośrednie operacje na liście, takie jak użycie funkcji

---

<sup>1</sup>Batchowanie - technika przetwarzania danych w partiach, co pozwala na efektywniejsze zarządzanie dużymi zbiorami danych.

<sup>2</sup>REST API - interfejs umożliwiający komunikację między systemami za pomocą zapytań HTTP.

*lookup* i *filter*, prowadzą do długiego czasu ładowania danych i wolniejszego wykonywania zapytań. Problemy te są szczególnie widoczne przy dużych zbiorach danych, gdzie każda operacja na liście wymaga znacznych zasobów i czasu.

- **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

- [1] lancedMicrosoft. Understand Power Apps Studio - Power Apps, October 2024.



© 2025 Remigiusz Wolniak, Michał Gajdzis

Instytut Robotyki i Inteligencji Maszynowej

Politechnika Poznańska, Wydział Automatyki, Robotyki i Elektrotechniki

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