JOURNAL
COMPUTER SCIENCES INSTITUTE

JCSI 31 (2024) 109–115
Received: 6 April 2024
Accepted: 21 May 2024

Comparative analysis of Microsoft's „low-code" programming technology

Analiza porównawcza technologii „low-code" oprogramowania firmy Microsoft

Grzegorz Łopata[a,*], Konrad Gromaszek[b]

[a] *Department of Computer Science, Lublin University of Technology, Nadbystrzycka 36B, 20-618 Lublin, Poland*
[b] *Department of Electronics and Information Technology, Nadbystrzycka 36B, 20-618 Lublin, Poland*

Abstract
According to various investigations, 'low code' platforms will form a large part of software development's market in the future. In that research paper was executed a comparison of Power Fx language functions: *Patch*, *UpdateIf*, *SubmitForm* and graphical interfaces, which allow making applications in Power Apps, were subjected to analysis. Research paper has shown that the average time of data recording into Dataverse using *Patch* function was slightly shorter than using *SubmitForm*. Moreover, it was found, that the average time required by the test group for creation Model – Driven app was 40% less than the average time for creation Canvas app and was demonstrated that a graphical interface, which enables making Power Apps Canvas is more efficient than Model – Driven graphical interface.

Keywords: comparative analysis; low-code; Microsoft; Dataverse

Streszczenie
Według różnych badań, platformy „*low-code*" w przyszłości będą stanowić dużą część rynku wytwarzania oprogramowania. Artykuł ma na celu porównać wydajności dostępnych funkcji języka Power Fx: *Patch*, *UpdateIf*, *SubmitForm* oraz ocenić interfejsy służące do wytwarzania oprogramowania w usłudze Microsoft Power Apps. Wykazano, że średni czas zapisu danych do Dataverse za pomocą funkcji *Patch* był nieznacznie krótszy niż za pomocą funkcji *SubmitForm*. Ponadto stwierdzono, że średni czas potrzebny grupie badawczej na stworzenie aplikacji „*model – driven*" był o 40% mniejszy, niż średni czas stworzenia aplikacji typu „*canvas*" oraz że interfejs *Power Apps Canvas* jest wydajniejszy niż interfejs *Power Apps Model – Driven*.

Słowa kluczowe: porównanie wydajności; low-code; Microsoft; Dataverse

*Corresponding author
Email address: **grzegorz.lopata@pollub.edu.pl** (G. Łopata)

1. Wstęp

Postęp technologiczny oraz związany z nim rozwój komputeryzacji w różnych dziedzinach życia powoduje, że tematyka związana z projektowaniem rozwiązań informatycznych jest bardzo ważna. Powszechnie następująca informatyzacja różnych sfer życia powoduje systematyczny rozwój nowych technologii wytwarzania oprogramowania.

Obecnie występująca złożoność kodowania aplikacji w językach programowania nie spowoduje zmniejszenia wytwarzanego oprogramowania, może jednak doprowadzić do wytworzenia języków programowania wyższego poziomu, które pozwolą na pisanie prostszego kodu [1]. Jedną z nowoczesnych technik w dziedzinie wytwarzania oprogramowania są technologie „*low-code*", będące usługami znajdującymi się w chmurze [2].

Termin „*low-code*", w kontekście rozwoju oprogramowania, został użyty przez firmę Forrester w 2014 roku. Platformy deweloperskie „*low-code*" sprawiają, że rozwijanie aplikacji jest dostępne dla większej liczby programistów ze względu na uproszczony proces wytworzenia oprogramowania, który wykorzystuje programowanie wizualne (wykorzystywanie zaimplementowanych przez twórców platformy elementów, np. formularzy do sprawnego tworzenia rozwiązań). Celem wytwarzania i rozwoju oprogramowania w technologiach „*low-code*" jest rozwiązanie problemu braku zasobów ludzkich w IT oraz przyśpieszenie procesu wytwarzania i dostarczania aplikacji poprzez znaczącą redukcję koniecznego programowania. Producenci platform „*low-code*" liczą na uczynienie transformacji cyfrowej oraz automatyzacji bardziej efektywną i dostępną. Obecnie w świecie biznesowym widoczny jest wpływ „*low-code*", wynikający z obiektywnych prognoz i realnych liczb oznaczających inwestycje w narzędzia niskiego kodu, bądź fakt, że dużym firmom zależy, żeby oferować usługi w tej dziedzinie. Firmy należące do *Big Tech* (Google, Amazon, Apple, Facebook oraz Microsoft) posiadają rozwiązania „*low-code*", które są odpowiedzialne za wsparcie klientów w wykorzystaniu usług podczas tworzenia własnych rozwiązań [3].

Według firmy Gartner [4], udział nowych aplikacji na rynku wytwarzanych za pomocą platform „*low-code*" w 2020 roku wynosił mniej niż 25%, natomiast według prognoz w 2025 roku powinien osiągnąć 70% wszystkich nowych rozwiązań. Można założyć, że technologie „*low-code*" będą przyszłością rozwoju oprogramowania, jeżeli przewidywania Gartnera się sprawdzą.

Według badania dotyczącego popularności technologii niskokodowych w Chinach, opublikowanego przez firmę Forrester [5], w ciągu 18 miesięcy (pomiędzy

majem 2020 a listopadem 2021) 58% przedsiębiorstw zaczęło korzystać z usług i platform „*low-code*" do tworzenia oprogramowania informatycznego, natomiast 16% przedsiębiorstw zamierza wdrażać te rozwiązania w swoich firmach.

Platformy „*low-code*" posiadają własne języki umożliwiające wytwarzanie oprogramowania. W 2021 roku firma Microsoft [6] wydała język „*low-code*" o nazwie *Power Fx* [7] dla platformy Power Apps [8], która została wydana w 2016 roku.

2. Przegląd literatury

Istnieją liczne prace naukowe obejmujące tematyką sposoby porównywania i oceniania interfejsów graficznych aplikacji internetowych czy mobilnych. Autorzy wykorzystują różne metody badawcze, m.in.: metryka WUP (ang. *Web Usability Points*), listy kontrolne LUT, które umożliwiają jak najlepszą ocenę wybranego interfejsu w chmurze [9].

Biorąc pod uwagę charakter artykułu i rodzaj przeprowadzonych badań, publikacją naukową, która otwiera przegląd literatury, jest „Ergonomia systemów informatycznych" [10]. Autor przedstawia i opisuje koncepcje ergonomii systemów, czynniki, które wpływają na ergonomię interfejsu oraz sposoby projektowania i oceny jakości interfejsów oprogramowania.

W artykule [11] pięcioosobowa grupa badawcza postawiła zestaw zadań obejmujący: znalezienie i uzyskanie dostępu do strony z ocenami, odnalezienie planu zajęć studenta, odszukanie materiałów kursowych, wysłanie pustego pliku na stronę zadania, wysłanie wiadomości do osoby prowadzącej dany przedmiot. Przeprowadzone badanie składało się z czterech głównych części:

1. Przeprowadzenie przez grupę badawczą oceny aplikacji Microsoft Teams oraz Blackboard za pomocą heurystyki Nielsena i zestawu zadań, w celu identyfikacji błędów w wybranych systemach.
2. Przeprowadzenie przez ekspertów porównania parami czterech głównych kategorii kwestionariusza USE (ang. *Usefulness, Ease of Use, Ease of Learning, Satisfaction*) – metodą hierarchiczną AHP (ang. *Analytic Hierarchy Process*) [9]. Ta część pozwoliła badaczowi wykluczyć niechciane dane oraz pozostawić trzy podkategorie oparte na ważności, które docelowo znajdą się w nowym zestawie heurystyk.
3. Przeprowadzenie przez grupę badawczą porównania parami metodą AHP proponowanych heurystyk, aby autor artykułu mógł stworzyć nową propozycję heurystyki, która będzie podlegać ocenie.
4. Przeprowadzenie przez ekspertów oceny aplikacji Microsoft Teams oraz Blackboard za pomocą nowej proponowanej heurystyki i zestawu zadań, które zostały wykonane podczas oceny, w celu identyfikacji błędów w wybranych systemach.

Na zakończenie badania wyniki oceny grupy badawczej z części czwartej eksperymentu zostały porównane z wynikami uzyskanymi w pierwszej części. Celem pracy badawczej było ustalenie, czy nowa heurystyka pomogła znaleźć i zidentyfikować więcej błędów i problemów w badanych aplikacjach. Na podstawie uzyskanych wyników, autor artykułu stwierdził, że rozszerzenie heurystyki Nielsena o dodatkowe zagadnienia z kwestionariusza USE pozwoliło zwiększyć liczbę wyszukiwanych błędów o 177,4% dla aplikacji Microsoft Teams oraz o 144,2% dla aplikacji Blackboard w stosunku do zastosowania standardowej heurystyki Nielsena (część pierwsza opisanego eksperymentu).

W artykule [12] przeprowadzono eksperyment odnoszący się do określenia krótkotrwałej zapamiętywalności oraz wykorzystanie tych wyników do oceny porównawczej użyteczności stron internetowych administracji publicznej. Ocena dokonana przez ekspertów została wykonana z wykorzystaniem listy kontrolnej oraz metody określania oceny użyteczności strony internetowej. Eksperymenty zostały wykonane na trzech stronach internetowych urzędów administracji publicznej dla następujących miast: Warszawa, Lublin oraz Chełm. Każda strona obejmuje te same funkcje oraz została utworzona w oparciu o takie same zasady dostępu do informacji publicznej w Polsce rozpowszechniane jako Biuletyn Informacji Publicznej (BIP). Przeprowadzone badania zostały podzielone na trzy fazy:

1. Ocena WUP wszystkich badanych stron internetowych za pomocą wybranej metody – lista LUT.
2. Przeprowadzenie przez grupę badawczą eksperymentu, który umożliwił wyznaczenie wartości współczynnika krótkoterminowej zapamiętywalności ML (ang. Memorability Level) dla wszystkich analizowanych stron.
3. Analiza pozyskanych danych, wykonanie obliczeń, utworzenie wykresów oraz analiza korelacji między WUP a ML dla badanych stron użyteczności publicznej.

Badanie korelacji między miarami użyteczności stron internetowych – miara WUP oraz współczynnik ML – wskazało silną ujemną korelację. Do określenia miary stopnia korelacji zastosowano współczynnik korelacji Pearsona. Współczynnik ten dla przeprowadzonych eksperymentów wyniósł -0,97. Wynik badania jest niespodziewany – wraz ze wzrostem użyteczności interfejsu graficznego, maleje stopień zapamiętywalności.

Autorzy artykułu [13] przedstawili eksperymenty umożliwiające ocenę wpływu jakości interfejsu na poziom zapamiętywalności wybranych stron internetowych sklepów online. W celu oceny jakości określonych interfejsów wykorzystano heurystyki Nielsena oraz listy LUT. W dalszej części artykułu wykonano badanie, które pozwoliło wyrazić poziom zapamiętywalności wybranych aplikacji webowych. Wynik przeprowadzonego eksperymentu jest zgodny z oczekiwaniami – prosty, dobrze zbudowany, klarowny interfejs przyczynia się do zwiększenia poziomu zapamiętywalności aplikacji internetowych. Dodatkowo, autorzy publikacji zauważyli, że im mniejsza wartość metryki WUP, tym gorsza jakość interfejsu. co skutkuje wydłużeniem czasu potrzebnego na znalezienie konkretnej informacji danej aplikacji.

Ze względu na fakt, że usługi chmurowe firmy Microsoft, służące do tworzenia oprogramowania są stosunkowo młodą technologią wytwarzania aplikacji, nie ma możliwości znalezienia prac naukowych dotyczących

porównania różnych metod zapisu danych w usłudze Power Apps. Autorzy w swoich publikacjach naukowych skupiali się na zagadnieniach związanych z m.in.: opisem i porównaniem różnych technologii „*low-code*", oceną popularności tych technologii oraz rozwojem różnego oprogramowania za pomocą chmurowych technologii niskokodowych.

W artykule [3] przedstawione zostały kluczowe koncepcje, czynniki i zmienne obejmujące LCDP (ang. *Low-code Development Platforms*), w celu zrozumienia obecnego stanu wiedzy. Autorzy przeanalizowali dane uzyskane z dwóch najpopularniejszych forów internetowych związanych z branżą IT: Stack Overflow oraz Reddit (301 wybranych postów spośród 1574 uzyskanych). Na podstawie tych danych, autorzy sporządzili listy zalet i wad dotyczących platform programistycznych *low-code*. Do zalet LCDP zaliczono: szybszy rozwój aplikacji, niższe koszty IT, łatwość nauki i użytkowania, w pełni gotowe do użycia moduły oraz możliwość dalszej rozbudowy. Do ograniczeń i wyzwań związanych z LCDP należą: wysokie ceny licencji, wysoki poziom trudności nauki, wolne ładowanie i publikowanie. Według autorów, wykorzystanie platform *low-code* do tworzenia oprogramowania ciągle rośnie, technologie *low-code* są ciągle rozwijane.

Możliwość projektowania procesów biznesowych zgodnie z określonymi wymaganiami w celu rozwoju systemu ERP (ang. *Enterprose Resources Planning*) w technologii „*low-code*" omówiono w artykule [14]. Autor zaprezentował zakres wdrożenia rozwojowego systemu w sytuacji, kiedy każda zmiana wymagań biznesowych została podzielona na elementy składowe: węzły procesu oraz wykorzystane usługi, bądź komponenty chmury. W dalszej części artykułu przedstawiono badania dotyczące popularności technologii niskokodowych w Chinach. Według tych badań w ciągu 18 miesięcy – między majem 2020 roku a listopadem 2021 – 58% przedsiębiorstw zaczęło korzystać z usług i platform „*low-code*" do tworzenia systemów informatycznych, natomiast 16% przedsiębiorstw zamierza wdrażać w swoich firmach platformy „*low-code*". W celu poprawy elastyczności i jakości systemów ERP, autor zasugerował stosowanie hierarchicznej struktury, która wspiera rozwój systemu aplikacji danego przedsiębiorstwa. Zdefiniowane zostały trzy obiekty semantyczne, które mogą być wykorzystywane wielokrotnie, należą do nich: kluczowe aplikacje, procesy oraz usługi / komponenty chmury.

W przypadku korzystania z tej hierarchii, gdy wyniknie potrzeba zmiany wymagań biznesowych, można w szybki sposób zidentyfikować oraz zlokalizować komponenty i usługi, które będą na nią wpływać, zgodnie z hierarchiczną strukturą.

Autorzy publikacji [1] opisują problem związany z brakami kadrowymi wykwalifikowanych pracowników w dziedzinie sztucznej inteligencji oraz przedstawiają możliwość rozwiązania tego kłopotu za pomocą narzędzi i platform „*low-code*". Według przytoczonych badań 83% przedsiębiorstw wymieniło AI jako priorytet rozwoju, a miało problem z rekrutacją odpowiednich specjalistów.

Autorzy zauważyli, że liczba narzędzi „*low-code*" stale rośnie, a do najbardziej znanych i mających największą liczbę użytkowników zaliczyć można m.in.: Mendix, OutSystems, Appian i GeneXus. Według dostępnych danych oprogramowanie OutSystems, Appian i Mendix wykazały przekroczenie 100 milionów dolarów rocznych przychodów. Należy również zauważyć, że firmy należące do *Big Tech* (Google, Amazon, Apple, Facebook oraz Microsoft) mają własne rozwiązania „*low-code*", które są zorientowane na pomaganie klientom w wykorzystaniu usług podczas tworzenia własnych rozwiązań.

Według autorów niskokodowa architektura oprogramowania *Smart* powinna obejmować:

- Edytor modeli – dostarczenie opisu systemu oprogramowania w określonej notacji;
- Generator kodu – dane przekazane w modelu prowadzą do wytworzenia kodu implementującego zróżnicowane procesy w narzędziu „*low-code*";
- Szkolenie – generator kodu tworzy materiały do nauki modelu ML;
- Wdrożenie – na wybranej platformie AI;
- Standardowe komponenty oprogramowania –nie zawierają funkcji *smart*. Moduły te współpracują z komponentami AI przy użyciu dedykowanego interfejsu programowania aplikacji.
- Monitorowanie i *feedback* – wszystkie komponenty AI powinny być monitorowane w czasie rzeczywistym oraz testowane po procesie wdrożenia, w celu oceny poprawności ich działania przez zespół programistyczny.

Wszystkie wymienione komponenty w zaprezentowanej architekturze są stworzone i istnieją jako pojedyncze elementy w innych platformach „*low-code*", sztucznej inteligencji oraz monitoringu. Połączenie ich w jedną strukturę, może znacząco wpłynąć na rozwój inteligentnego oprogramowania.

Badania Gartnera, według których udział nowych aplikacji tworzonych przez organizacje przy użyciu technologii „*low-code*" powinny osiągnąć 70% wszystkich nowo tworzonych rozwiązań w 2025 roku zostały przedstawione w artykule [2]. Wskazano w nim, że w 2020 roku, udział nowych rozwiązań wytworzonych w technologiach „*low-code*" wynosił mniej niż 25%. Niemniej jednak, jeżeli prognozy Gartnera okażą się słuszne, to technologie niskiego kodu staną się przyszłością rozwoju oprogramowania. Ponadto, artykuł dostarcza informacje, dotyczące otwartego języka „*low-code*" wydanego przez Microsoft w 2021 roku dla platformy Power Apps, o nazwie *Power Fx*.

Według autorów artykułu, do wymagań nowej generacji języków „*low-code*" należą następujące elementy:

- Obiekty danych (pole tekstowe, numer, data, załącznik, użytkownik);
- Model procesu (dodanie cyklu życia do obiektów danych, zadań);
- Role i użytkownicy (dany użytkownik ma nadaną konkretną rolę);

- Akcje (reakcja na zdarzenia na obiektach danych i zadań).

3. Cel i hipotezy badawcze

Celem artykułu jest analiza porównawcza dwóch technologii „*low-code*" firmy Microsoft Power Platform – Canvas Application oraz Model-driven Application. Powyższe technologie porównano pod względem wyglądu aplikacji oraz interfejsu użytkownika. Do oceny interfejsu graficznego, który służy do tworzenia aplikacji w usłudze Power Apps zostaną wykorzystane metryki oparte na pomiarach czasu – testy użytkownika, listy LUT oraz metryki WUP.

Porównaniu podlegać będą również dostępne metody zapisu danych do bazy danych – Dataverse. Technologie te, będą zestawione na podstawie dwóch aplikacji posiadających te same funkcjonalności, w oparciu o to samo źródło danych. Do oceny wydajności metod zapisu (kontrolka formularza oraz funkcja PowerFx *Patch*) wykorzystano metryki oparte na pomiarach czasu wykonywania zapisu danych do źródła danych Dataverse, które uzyskane zostaną z wbudowanego w Power Platform narzędzia monitora.

Końcowym rezultatem wynikającym z przeprowadzonych prac, będzie określenie, która technologia wytwarzania aplikacji „*low-code*" jest bardziej wydajna – Canvas Application czy Model-Driven Application oraz który sposób zapisu danych jest bardziej efektywny.

Bazując na przeprowadzonym przeglądzie literatury oraz na doświadczeniu autora z usługami chmurowymi firmy Microsoft przedstawiono następujące hipotezy badawcze:

- H1: Zapis danych z formularza do źródła danych Dataverse za pomocą funkcji PowerFx – *Patch* jest bardziej wydajny niż zapis danych za pomocą domyślnej kontrolki „*Form*".
- H2: Średni czas wykonania aplikacji typu „*model-driven*" według scenariusza testowego, jest mniejszy niż średni czas wykonania aplikacji typu „*canvas*".
- H3: Interfejs do tworzenia aplikacji typu „*canvas*" jest bardziej wydajny niż interfejs do tworzenia aplikacji typu „*model-driven*".

4. Metodyka badawcza

4.1. Stanowisko badawcze

W celu porównania wydajności różnych metod zapisu danych do źródła danych w usłudze Microsoft Power Apps użyto fizyczną maszynę Dell Latitude 5511, która pełni rolę hosta. Na stacji badawczej zainstalowany jest 64-bitowy system operacyjny Windows 10 Pro w wersji 22H2 (OS Build 19045.3155).

W tabeli 1 przedstawiona została szczegółowa specyfikacja techniczna maszyny hosta.

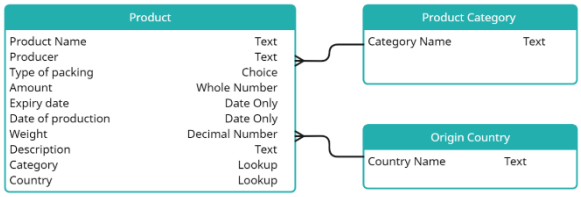
Tabela 1: Dane techniczne maszyny hosta, na której przeprowadzono opisywane badania

Procesor	Intel(R) Core(TM) i5-10400H CPU @ 2.60GHz, 2600 MHz, liczba rdzeni: 4, liczba wątków: 8
Dysk	Model: Micron 2200S NVMe, Typ: SSD NVMe (ang. Solid State Drive, Non – Volatile Memory Express), Pojemność dysku: 256 GB
Pamięć RAM	16 GB (DDR4 2933 MHz)

Do badań wykorzystano bezpłatną wersję Microsoft Power Platform – Microsoft 365 Developer Program. Programistyczna platforma firmy Microsoft zawiera następujące licencje: „Microsoft Power Apps Plan 2 Trial", „Microsoft Power Automate Free" oraz „Microsoft 365 E5 Developer (without Windows and Audio Conferencing)" [15, 16].

4.2. Źródło danych Dataverse

Na Rysunku 1 przedstawiono strukturę bazy danych sklepu online (z tabelami: produkty, kategorie produktów oraz kraj pochodzenia); utworzoną za pomocą środowiska bazodanowego wbudowanego w Power Platform – Dataverse.



Rysunek 1: Baza danych Dataverse dla testowych aplikacji Power Apps Canvas.

W skład utworzonej bazy danych wchodzą trzy tabele Dataverse:

- *Product* z podstawowymi informacjami, dotyczącymi produktu: nazwę, producenta, rodzaj opakowania, ilość, datę produkcji, datę ważności, wagę, opis, rodzaj kategorii oraz nazwę kraju pochodzenia;
- *Product Category,* zawierająca nazwy kategorii, które charakteryzują dany produkt;
- *Origin Country*, która zawiera nazwy państw z których pochodzą produkty.

4.3. Aplikacje testowe

Testowe aplikacje zostały zrealizowane w technologii „*low-code*" Microsoft Power Apps Canvas, są zbudowane w oparciu o bazę danych Dataverse oraz posiadają identyczne funkcjonalności. Stworzone oprogramowanie jest w stanie realizować zadania typu CRUD – tworzenie (ang. *create*), odczytanie (ang. *read*), aktualizacja (ang. *update*) oraz usuwanie (ang. delete). Zrealizowane aplikacje testowe są zbudowane w oparciu o analizowane metody zapisu danych do źródła danych Dataverse:

- Domyślny formularz interfejsu Power Apps Canvas wraz z funkcją *SubmitForm()* oraz funkcja *UpdateIf()* języka PowerFx;
- Funkcja PowerFX – *Patch()*.

Wymagania funkcjonalne utworzonej aplikacji testowej:

- Wyświetlanie danych przechowywanych w tabelach Dataverse: *Product*, *Product Category* oraz *Origin Country*;
- Dodanie nowego rekordu do tabel *Product*, *Product Category*, *Origin Country*;
- Usuwanie wybranego rekordu z tabel *Product*, *Product Category*, *Origin Country*;
- Edytowanie wybranego rekordu z tabel *Product*, *Product Category*, *Origin Country*.

4.4. Scenariusze badawcze

Badanie wydajności wybranych funkcji języka Power Fx, umożliwiających dokonanie operacji CRUD zrealizowano według poniższego scenariusza:

1. Wykonanie operacji CRUD w aplikacjach testowych (dodanie nowego rekordu do bazy danych, modyfikacja istniejącego rekordu, usunięcie istniejącego rekordu).
2. Uzyskanie czasu trwania operacji przez aplikację testową za pomocą wbudowanego w Power Apps narzędzia monitora zdarzeń.
3. Powtórzenie czynności, aż do uzyskania 10 wyników.
4. Analiza uzyskanych danych wraz z opracowaniem wyników badań.

W celu oceny poprawności drugiej hipotezy badawczej H2 pięcioosobowa grupa testowa wykonała poniższe scenariusze badawcze. Określone zostały warunki początkowe: Użytkownik korzysta z przeglądarki Google Chrome. Użytkownik jest zalogowany na stronie Microsoft Power Platform – make.powerapps.com. Użytkownik korzysta ze środowiska „*20xqks (default)*" i znajduje się w solucji o nazwie „*MA Thesis Survey*".

Scenariusz 1 – Tworzenie Canvas Application

1. Stwórz nową aplikację – „*Canvas app*". Nadaj nazwę aplikacji oraz wybierz orientację aplikacji.
2. Dodaj źródło danych – Dataverse, tabela o nazwie „*MA Thesis Table*".
3. Wyświetl dane z Dataverse w aplikacji. W tym celu wykorzystaj kontrolkę galerii (ang. *Gallery*).
4. Zmodyfikuj galerię (każdy rekord galerii ma zawierać pola: „*Title*", „*Date*" oraz „*Priority*").
5. Dodaj formularz (kontrolka „*Edit form*") wraz z przyciskiem, który będzie dodawał nowy rekord do bazy danych Dataverse. Należy skorzystać z właściwości „*OnSelect*" przycisku oraz z funkcji „*SubmitForm()*". Formularz ma zawierać pola: „*Title*", „*Date*", „*Priority*" oraz „*Description*".
6. Za pomocą utworzonego formularza, dodaj dane do bazy danych Dataverse i zweryfikuj poprawność działania scenariusza.

Scenariusz 2 – Tworzenie Model – Driven Application

1. Stwórz nową aplikację – „*Model – driven app*". Nadaj nazwę aplikacji.
2. W „*site map*" dodaj do komponentu „*Subarea*" źródło danych – tabelę Dataverse: „*MA Thesis Table*". Opublikuj i zapisz zmiany dokonane w „*site map*".
3. Zmodyfikuj widok domyślny publiczny danych wyświetlanych dla tabeli „*MA Thesis Table*". Widok powinien zawierać następujące pola: „*Title*", „*Date*", „*Priority*" oraz „*Description*".
4. Zmodyfikuj formularz główny, tak aby zawierał następujące pola: „*Title*", „*Date*", „*Priority*" oraz „*Description*".
5. Za pomocą formularza dodaj dane do bazy danych Dataverse i zweryfikuj poprawność działania scenariusza.

W celu weryfikacji trzeciej hipotezy badawczej H3 przeprowadzono ocenę interfejsów służących do wytwarzania oprogramowania w usłudze Microsoft Power Apps za pomocą metryki WUP.

5. Wyniki

Wyniki dotyczące badania pierwszego, w którym mierzony był czas wykonywania operacji CRUD w aplikacjach Power Apps Canvas za pomocą badanych funkcji języka Power Fx (*Patch*, *UpdateIf* oraz *SubmitForm*) zostały przedstawione w tabeli 2 [17].

Tabela 2: Zestawienie średnich czasów wykonania operacji CRUD, dla Aplikacji I oraz II

Aplikacja	Średni czas dodania rekordu [ms]	Średni czas edycji istniejącego rekordu [ms]	Średni czas usunięcia istniejącego rekordu [ms]
Aplikacja I SubmitForm oraz UpdateIf	139,8	181,7	157,2
Aplikacja II Patch	138,97	244,8	153,4

Wyniki przedstawione w tabeli 2, zostały uzyskane za pomocą narzędzia monitora zdarzeń, który jest wbudowany w usługę Microsoft Power Apps [18].

Średni czas dodania nowego rekordu za pomocą funkcji *SubmitForm* jest o 0,6 % dłuższy niż czas dodania rekordu za pomocą funkcji *Patch*. Średni czas edycji istniejącego rekordu za pomocą funkcji *UpdateIf* jest o 25,78 % krótszy niż czas zmodyfikowania rekordu za pomocą funkcji *Patch*. Średni czas usunięcia istniejącego rekordu w Aplikacji I przy użyciu funkcji *Remove* jest o 2,48 % dłuższy niż czas usunięcia rekordu za pomocą tej samej funkcji w Aplikacji II.

Szybszą metodą zapisu w badanych rozwiązaniach okazała się funkcja *Patch* języka Power Fx. Różnica w czasie zapisu danych do źródła Dataverse, pomiędzy funkcjami *SubmitForm* oraz *Patch* jest bardzo niewielka i wynosi 0,83 ms. Pod względem modyfikacji danych, zdecydowanie bardziej wydajnym rozwiązaniem jest wykorzystanie funkcji *UpdateIf* (Aplikacja I). Średni czas edycji danych za pomocą funkcji *UpdateIf* jest o 63,1 ms krótszy niż przy użyciu funkcji *Patch*.

Wyniki dotyczące badania drugiego, w którym mierzone były czasy wykonania aplikacji w usłudze Power Apps zostały przedstawione w tabeli 3 i tabeli 4.

Tabela 3: Zestawienie średnich czasów wykonania poszczególnych kroków Scenariusza I przez grupę badawczą

Punkt scenariusza	Średni czas [s]
1	42,4
2	92,6
3	80

4	307,2
5	371,6
6	134,6

Najszybciej wykonywanym krokiem scenariusza był krok pierwszy, który polegał na utworzeniu pustej aplikacji. Średni czas wykonywania tego polecenia wyniósł 42,4 sekundy.

Najwolniej wykonywanym krokiem scenariusza był krok piąty, który polegał na dodaniu do aplikacji formularza wraz z przyciskiem obsługującym proces zapisania danych z formularza do bazy danych. Średni czas wykonywania tego polecenia wyniósł 371,6 sekundy. Największymi trudnościami, które znacznie wydłużyły czas realizacji kroku piątego była zmiana własności „Default mode" na „New" kontrolki „Form" oraz konfiguracja przycisku, tak aby po naciśnięciu go, dane z formularza zostały zapisane w źródle danych.

Równie duży problem stanowił krok czwarty instrukcji – modyfikacja galerii do wyglądu oczekiwanego, przedstawionego w scenariuszu. Średni czas wykonywania tego polecenia wyniósł 307,2 sekundy. Średni czas podłączenia aplikacji do źródła danych (tabela Dataverse) wyniósł 92,6 sekundy, natomiast wyświetlenie danych z bazy danych w aplikacji za pomocą galerii 80 sekund. Weryfikacja poprawności wykonania scenariusza (krok szósty) średnio zajęła 134,6 sekundy.

Tabela 4: Zestawienie średnich czasów wykonania poszczególnych kroków Scenariusza II przez grupę badawczą

Punkt scenariusza	Średni czas [s]
1	33,8
2	150
3	226,2
4	100,6
5	106,2

Najszybciej wykonywanym krokiem scenariusza II był krok pierwszy, który polegał na utworzeniu pustej aplikacji. Średni czas wykonywania tego polecenia wyniósł 33,8 sekundy.

Najwolniej wykonywanym krokiem scenariusza był krok trzeci, który polegał na zmodyfikowaniu widoków domyślnych (publicznych) w tabeli „MA Thesis Table" w Dataverse. Średni czas wykonywania tego polecenia wyniósł 226,2 sekundy. Największą trudnością, która znacznie wydłużyła czas realizacji kroku trzeciego, był brak znajomości środowiska bazodanowego „Dataverse" – grupie testowej trudność sprawiła nawigacja i odnalezienie odpowiedniego widoku do zmodyfikowania.

Punktem scenariusza, który był średnio drugim najwolniejszym, był drugi krok instrukcji. Polegał on na dodaniu do komponentu „subarea" źródła danych – tabeli „MA Thesis Table". Średni czas wykonywania tego polecenia wyniósł 150 sekund.

Krok czwarty, który polegał na modyfikacji głównego formularza w tabeli „MA Thesis Table", zajął średnio 100,6 sekundy. Średni czas potrzebny grupie badawczej na modyfikację głównego formularza był o 56% mniejszy niż średni czas modyfikacji widoku domyślnego. Weryfikacja poprawności wykonania scenariusza (krok piąty) zajęła średnio 106,2 sekundy.

Utworzone aplikacje testowe („Canvas App" oraz „Model – Driven App") miały takie same funkcjonalności – pozwalały użytkownikowi dodać nowy rekord do tabeli w „Dataverse".

Średni czas potrzebny grupie testowej na stworzenie aplikacji „Model – Driven" był o 40% mniejszy niż średni czas stworzenia aplikacji typu „canvas".

W tabeli 5 zostały przedstawione wyniki badania oceny interfejsów, służących do tworzenia aplikacji dostępnych w usłudze Microsoft Power Apps, które zostały uzyskane za pomocą metryki WUP [10].

Tabela 5: Zestawienie średnich czasów wykonania poszczególnych kroków Scenariusza II przez grupę badawczą

Typ interfejsu	Metryka WUP
Canvas Application	4,192
Model – Driven Application	3,782

Wartość metryki WUP dla interfejsu do tworzenia aplikacji typu „*model – driven*" jest o 11% niższa niż wartość metryki WUP interfejsu do tworzenia aplikacji typu „*canvas*". Na podstawie list LUT oraz metryk WUP, należy zauważyć, że interfejs do tworzenia aplikacji typu „*canvas*" jest lepszym interfejsem (wyżej ocenionym) niż interfejs do tworzenia aplikacji typu „*model – driven*".

6. Wnioski

Na podstawie zrealizowanych badań oraz uzyskanych wyników, które zostały przedstawione na Rysunku 2 hipoteza H1 została potwierdzona. Średni czas zapisu danych do Dataverse dla wszystkich trzech tabel, które tworzyły badane rozwiązanie („*Product*", „*Product Category*" oraz „*Origin Country*"), przy użyciu funkcji języka Power Fx *Patch* był o 0,6% krótszy niż przy użyciu funkcji *SubmitForm*.

Na podstawie zrealizowanych badań oraz uzyskanych wyników, które zostały przedstawione w tabeli 3 oraz w tabeli 4 hipoteza H2 została potwierdzona. Utworzone przez badaną grupę aplikacje testowe (Canvas App oraz Model – Driven App) miały takie same funkcjonalności – pozwalały użytkownikowi dodać nowy rekord do tabeli w Dataverse. Średni czas potrzebny grupie testowej na stworzenie aplikacji „*model – driven*" był o 40% mniejszy niż średni czas stworzenia aplikacji typu „*canvas*".

Literatura

[1] J. Cabot, R. Clarisó, Low Code for Smart Software Development, IEEE Software Volume: 40, Issue: 1, (2022), https://doi.org/10.1109/MS.2022.3211352.

[2] G. Juhás, L. Molnár, A. Juhásová, M. Ondrišová, M. Mladoniczky, T. Kovácik, Low-code platforms and languages: the future of software development, In 20th International Conference on Emerging eLearning Technologies and Applications (ICETA) (2022) 286–293.

[3] P. M. Gomes, M. A. Brito, Low-Code Development Platforms: A Descriptive Study, In 17th Iberian Conference on Information Systems and Technologies (CISTI) (2022).

[4] Gartner, Gartner Says Cloud will be the Centerpiece of New Digital Experiences, 2021 https://www.gartner.com/en/newsroom/press-releases/2021-11-10-gartner-says-cloud-will-be-the-centerpiece-of-new-digital-experiences [18.09.2023].

[5] G. Lu, C. Dai, F. Giron, J. Bratincevic, T. Feng, B. Nagel, The State of Low-code Platforms in China, released by Forrester, Listopad 2021 [18.09.2023].

[6] Low-Code Development Platform | Microsoft Power Apps, https://powerapps.microsoft.com/en-us/low-code-platform/, [31.01.2023].

[7] Microsoft Power Fx overview, https://learn.microsoft.com/en-us/power-platform/power-fx/overview, [31.01.2023].

[8] Microsoft Power Apps, https://learn.microsoft.com/en-us/training/powerplatform/power-apps, [31.01.2023].

[9] M. Laskowski, Propozycje metodyk badania użyteczności interfejsów aplikacji, Informatyka Automatyka Pomiary w Gospodarce i Ochronie Środowiska (IAPGOŚ), Politechnika Lubelska, Styczeń 2012.

[10] M. Miłosz, Ergonomia systemów informatycznych, Politechnika Lubelska, 2014.

[11] M. Benaida, Developing and extending usability heuristics evaluation for user interface design via AHP, Soft Computing 27 (2023), 9693–9707, https://doi.org/10.1007/s00500-022-07803-4.

[12] M. Miłosz, M. Borys, M. Laskowski, Memorability Experiment Vs. Expert Method in Websites Usability Evaluation, In 15th International Conference on Enterprise Information Systems (ICEIS) (2013) 151–157.

[13] K. Pytka, M. Miłosz, Influence the quality of the interface of online stores on their memorability, Journal Computer Sciences Institute JCSI 15 (2020) 114–120, http://dx.doi.org/10.35784/jcsi.2040.

[14] L. Tang, ERP Low-code Cloud Development, In IEEE 13th International Conference on Software Engineering and Service Science (ICSESS) (2022).

[15] Microsoft 365 Developer Program, https://developer.microsoft.com/en-us/microsoft-365/dev-program, [01.07.2023].

[16] Microsoft 365 Developer Program FAQ, Microsoft 365 Developer Program FAQ | Microsoft Learn, [02.07.2023].

[17] Patch function in Power Apps, https://learn.microsoft.com/en-us/power-platform/power-fx/reference/function-patch, [15.04.2023].

[18] Omówienie narzędzia Monitor, https://learn.microsoft.com/pl-pl/powerapps/maker/monitor-overview, [14.06.2023].