RAPORT KOŃCOWY
Z WYKONANYCH BADAŃ

Warszawa, 30.11.2021 r.

Niniejszy raport prezentuje rezultaty prac badawczych w związku z realizacją projektu numer RPMA.01.02.00-14-b487/18, pt. „Prace badawcze i rozwojowe w zakresie innowacyjnej metody zarządzania przedsiębiorstwem z wykorzystaniem automatyzacji i robotyzacji procesów" współfinansowanego z Europejskiego Funduszu Rozwoju Regionalnego w ramach Osi Priorytetowej I „Wykorzystanie działalności badawczo-rozwojowej w gospodarce", Działanie 1.2 „Działalność badawczo-rozwojowa przedsiębiorstw", Regionalnego Programu Operacyjnego Województwa Mazowieckiego na lata 2014-2020.

Raport przygotowany przez DPC Polska (Lider) we współpracy z Polsko-Japońską Akademią Technik Komputerowych (Partner).

Wprowadzenie

Przedmiot projektu stanowiło opracowanie prototypu rozwiązania klasy RPA (ang. *Robotic Process Automation* – zrobotyzowanej automatyzacji procesów biznesowych) składającego się z robota procesowego, ang. *software robot*, w postaci automatu działającego jako program komputerowy oraz środowiska, w którym robot będzie działał, będzie uruchamiany, uczony, testowany i nadzorowany.

Wypracowane w toku projektu rozwiązanie środowiska robotycznego *Guardian* zostało oparte o wyniki badań przemysłowych prowadzonych przez Partnera akademickiego we współpracy z Liderem projektu. Zakres badań obejmował w szczególności metody i rozwiązania z obszarów konstruowania robotów w podejściu hybrydowym, łączącym podejście regułowe, wykorzystujące wiedzę ekspercką (ang. *rule-based*) z modułami umożliwiającymi automatyzację w oparciu o uczenie maszynowe (ML – ang. *Machine Learning*) i sztuczną inteligencję (AI – ang. *Artificial Intelligence*), w ujęciu Collaborative Interactive Machine Learning (CI ML).

Pomimo licznych wyzwań, które zostały napotkane w toku projektu, jego realizacja zakończyła się sukcesem. W szczególności w obliczu trudności związanych z sytuacją pandemiczną, skutkujących m.in. wydłużeniem procesu implementacji w związku z ograniczeniami pracy zdalnej wynikającej z reżimu sanitarnego po stronie obydwu partnerów udało się zgodnie z planem opracować, uruchomić i przetestować w środowisku operacyjnym prototyp rozwiązania. Osiągnięto tym samym zaplanowany w projekcie siódmy poziom gotowości technologicznej (TRL7) dający podstawę do dalszej drogi w kierunku pełnej komercjalizacji rezultatów projektu, którą jednakowoż należy rozpatrywać w kontekście zmieniających się wyzwań na rynku, w szczególności nowych regulacji w ramach otoczenia prawno-ekonomicznego (m.in. wymóg elektronicznego obiegu faktur).

Niniejszy raport składa się z dwóch zasadniczych części. W pierwszej przedstawiono zwięzłe podsumowanie przeprowadzonych prac badawczo-rozwojowych oraz uzyskanych

rezultatów. W dalszej części raportu przedstawiono szczegóły poszczególnych etapów i wypracowanych w nich rozwiązań.

Spis Treści

Podsumowanie raportu

Prace badawcze i rozwojowe

Prace badawcze i rozwojowe prowadzone były w trzech głównych fazach. W każdej z nich występowały po dwa komplementarne etapy: badań przemysłowych oraz eksperymentalnych prac rozwojowych, w następującym układzie:

- faza 1: etap badawczy 1 oraz etap prac rozwojowych 1 (EB.1+ER.1),
- faza 2: etap badawczy 2 oraz etap prac rozwojowych 2 (EB.2+ER.2),
- faza 3: etap badawczy 3 oraz etap prac rozwojowych 3 (EB.3+ER.3.).

Poniżej przedstawiono podsumowanie poszczególnych etapów, zaś szczegóły poszczególnych etapów i wypracowanych rezultatów przedstawiono w dalszej części opracowania.

Faza 1: komplementarne etapy badawcze i rozwojowe EB1+ER1

W ramach etapu pierwszego prac badawczych (EB1) opracowano techniki miar wskaźników i narzędzi do oceny potencjału automatyzacji procesów biznesowych klientów DPC. Opracowano również założenia technicznej specyfikacji języka programowania robotów, opis syntaktyczny i jego semantyki. Dokonano także oszacowania potencjału rezultatów prac badawczych w celu dalszego zastosowania w pracach rozwojowych. Na podstawie przeprowadzonych badań przemysłowych w etapie 1 (EB1) w komplementarnym do niego pierwszym etapie prac rozwojowych (ER1) zaimplementowano zaplanowany w tym etapie interpreter języka robotycznego oraz oprogramowanie potrafiące rozpoznawać i wykonywać polecenia ze specyfikacji języka dla robotów. W ramach prac rozwojowych przeprowadzono również prototypowe uruchomienie w środowisku klienta firmy DPC, na wybranym procesie biznesowym. Opracowano również prototyp oprogramowania

monitorującego i zarządzającego pracą robotów. Szczegóły dotyczące obydwu komplementarnych etapów (EB1 i ER1), w szczególności wypracowanych rezultatów opisano w dalszej części raportu.

Faza 2: komplementarne etapy badawcze i rozwojowe EB2+ER2

W ramach drugiego etapu badawczego (EB2) zostały wybrane rodzaje algorytmów w dopasowaniu do domeny dziedzinowej problemu badawczego. W oparciu o postulowane opisy cech klasyfikacyjnych, wag oraz modeli uczenia wykonano testy w warunkach laboratoryjnych. Opracowano założenia do modelu wprowadzania zmian oraz oceny wyników oraz technicznego PoC (proof of concept) dla typowego procesu biznesowego z zakresu procesów klientów DPC. W ramach realizacji prac rozwojowych drugiej fazy projektu (ER2) komplementarnych do drugiego etapu prac badawczych (EB2) wykonano prototyp rozwiązania na podstawie wybranych założeń i elementów PoC z EB2 realizujący dostosowywanie się do zmiennych formatów dokumentów występujących wśród klientów firmy DPC przetwarzających duże ilości dokumentów, w szczególności finansowych. Prototyp został zweryfikowany w kontekście środowiska biznesowego jednego z kluczowych klientów korporacyjnych firmy DPC. Prototyp oprogramowania oparto o zmienny algorytm działania robota pod wpływem decyzji związanych ze zmienionym środowiskiem wykonawczym.

Szczegóły dotyczące obydwu komplementarnych etapów (EB2 i ER2), w szczególności wypracowanych rezultatów opisano w dalszej części raportu.

Faza 3: komplementarne etapy badawcze i rozwojowe EB3+ER3

W ramach trzeciego etapu prac badawczych (EB3) opracowano metodę rejestrowania zachowania użytkowników podczas pracy z systemami a następnie techniczny PoC (ang. Proof of concept – dowód słuszności koncepcji) rozwiązania. Opracowane rozwiązanie przewiduje możliwość współdziałania z różnymi algorytmami sterującymi automatyzacją

z wykorzystaniem języka robotycznego. W ramach etapu eksperymentalnych prac rozwojowych (ER3), komplementarnego do trzeciego etapu prac badawczych EB3 opracowano prototyp silnika do analizy procesów, wytwarzający zapis procesu w postaci plików poleceń języka dla robotów. Prototyp opracowano na bazie rzeczywistych przykładów procesów biznesowych z obszaru objętego planowanymi wdrożeniami u klientów DPC. Prototyp został zweryfikowany w kontekście środowiska biznesowego jednego z kluczowych klientów korporacyjnych firmy DPC.

Szczegóły dotyczące obydwu komplementarnych etapów (EB1 i ER1), w szczególności wypracowanych rezultatów opisano w dalszej części raportu.

Podsumowanie rezultatów przeprowadzonych prac badawczo-rozwojowych

W wyniku przeprowadzonych prac badawczych, na podstawie pozyskanej wiedzy, firma DPC opracowała innowacyjną usługę automatyzacji opartą o system Guardian do przetwarzania artefaktów procesów biznesowych w postaci dokumentów, w szczególności dokumentów finansowych. W toku przeprowadzonego projektu przeprowadzono pilotażowe testy wdrożenia prototypu systemu w kontekście środowiska biznesowego wybranego klienta firmy DPC. W szczególności opracowany w toku projektu system Guardian jest zgodnie z założeniami projektowymi prototypem kompleksowego rozwiązania klasy RPA (Robotic Process Automation), składającego się z robota procesowego (software robot) oraz środowiska, w którym robot jest uruchamiany, uczony, testowany i nadzorowany. Rozwiązanie oparto o wnioski i rekomendacje z badań przemysłowych prowadzonych przez Partnera akademickiego we współpracy z Liderem projektu. Zakres badań obejmował w szczególności metody i rozwiązania z obszarów konstruowania robotów w podejściu hybrydowym, łączącym podejście regułowe wykorzystujące wiedzę ekspercką (rule-based) z modułami autonomicznymi tworzonymi w oparciu o uczenie maszynowe (ML –Machine Learning) i sztuczną inteligencję (AI –Artificial Intelligence).

Wymienione wyżej prace rozwojowe stanowiły w swej istocie implementację wyników badań przemysłowych do rzeczywistych warunków procesów biznesowych i weryfikację opracowanych na etapie badań przemysłowych koncepcji w celu zmniejszenia kosztów wdrożenia oraz jego skomplikowania, w tym obsługi rozwiązania dla końcowych użytkowników. Rezultaty prac rozwojowych nad prototypem kompleksowego systemu Guardian zakończyły się sukcesem. Pomimo szeregu wyzwań związanych z ograniczeniami pandemicznymi i reżimem sanitarnym, w toku projektu udało się zgodnie z założeniami projektu opracować, uruchomić i przetestować w środowisku operacyjnym prototyp rozwiązania, osiągając tym samym zaplanowany w projekcie siódmy poziom gotowości

technologicznej (TRL7) dający podstawę do dalszej drogi w kierunku pełnej komercjalizacji rezultatów projektu. Szczegóły dotyczące rezultatów wypracowanych w poszczególnych etapach badawczych i rozwojowych przedstawiono w dalszej części opracowania.

Cechy i funkcjonalności w kontekście innowacyjności

W toku projektu w ramach przeprowadzonych badań i zdobytej wiedzy wytworzono unikatowy na rynku produkt w postaci kompleksowego systemu robotycznego Guardian, stanowiącego podstawę zintegrowanej adaptacyjnej usługi automatyzacji procesów biznesowych dla klientów DPC. Poniżej przedstawiono najważniejsze cechy i funkcjonalności uzyskanych rezultatów projektu świadczące o innowacyjności uzyskanych rezultatów w trzech obszarach tematycznych przewagi konkurencyjnej.

Pierwszy obszar obejmuje rozpoznawanie przez system nowych rodzajów dokumentów stanowiących dane wejściowe (np. faktury, dokumenty magazynowe, wnioski itp.), w oparciu o elastyczny generyczny system szablonów umożliwiających wykorzystanie algorytmów uczenia maszynowego i sztucznej inteligencji. Szczegóły rozwiązania opisano w dalszej części raportu.

Drugi obszar obejmuje przygotowywanie przez system podstawy procesu do robotyzacji na podstawie zapisanych danych związanych z przebiegiem procesu w postaci zapisu i odwzorowania działań w przestrzeni użytkownika. Metoda ta zakłada rejestrowanie czynności wykonywanych przez użytkownika, w tym rejestrowanie zdarzeń systemowych, umożliwiając na tej podstawie stworzenie mechanizmu translacji zarejestrowanego materiału do języka programowania robotów RFL – w szczególności poprzez możliwość wykorzystania punktów decyzyjnych, źródeł danych, pętli, miejsc wprowadzania bądź wyszukiwania danych. Szczegóły rozwiązania opisano w dalszej części raportu.

Trzeci obszar obejmuje reagowanie na zmiany w środowisku wykonawczym, które nie wpływają na przebieg procesu. Opracowane rozwiązanie przewiduje możliwość dostosowania do zmian w zakresie dopasowania układu elementów na ekranie, od zmiany nazw i zawartości pól, po zmiany w procesie (zmiany źródeł danych, zmiany kolejności wykonywania operacji itp.), z uwzględnieniem nowoczesnego podejścia CI ML

(Collaborative Interactive Machine Learning). Szczegóły rozwiązania opisano w dalszej części raportu.

Opisane powyżej **cechy i funkcjonalności planowanego do wdrożenia wykazują wysoką innowacyjność uzyskanych rezultatów** w kontekście warunków biznesowych kluczowych klientów firmy DPC zainteresowanych automatyzacją procesów biznesowych. W szczególności opracowane w projekcie w wyniku przeprowadzonych prac badawczych i rozwojowych rozwiązania technologiczne w postaci systemu Guradian umożliwiają firmie DPC przygotowanie kompleksowej usługi automatyzacji procesów biznesowych pod kątem potencjalnej komercjalizacji, dzięki osiągnięciu poziomu 7 TRL (testy prototypu w warunkach operacyjnych).

Szczegóły dotyczące rezultatów wypracowanych w poszczególnych etapach badawczych i rozwojowych przedstawiono w dalszej części opracowania.

Szczegółowy opis przeprowadzonych prac badawczo-rozwojowych

EB1: Opracowanie metod budowy robotów i ich środowiska

W ramach etapu pierwszego prac badawczych (EB1) opracowano techniki miar wskaźników i narzędzi do oceny potencjału automatyzacji procesów biznesowych klientów DPC. Opracowano również założenia technicznej specyfikacji języka programowania robotów, opis syntaktyczny i jego semantyki. Dokonano także oszacowania potencjału rezultatów prac badawczych w celu dalszego zastosowania w pracach rozwojowych.

Jednym z podstawowych wyzwań tego etapu było opracowanie listy mierzalnych wskaźników procesu, pozwalających na oszacowanie stopnia możliwości zrobotyzowania danego procesu w powiązaniu z kontekstem operacyjno-finansowym. Wypracowana w ramach niniejszego etapu lista została przedstawiona w dalszej części opracowania. Kolejnym powiązanym wyzwaniem było opracowanie założeń do zwięzłego i precyzyjnego języka zapisu ciągu poleceń dla robotów softwareowych. Zgodnie z założeniami język oparto o konstrukcję algorytmiczną, zaś szczególną uwagę skupiono na zwięzłości i kompleksowości struktury poleceń, z zachowaniem elastyczności obsługi wyjątków oraz rozszerzalności jego struktury, w tym pod kątem elementów uczenia oraz mechanizmów ułatwiających kontrolowanie działania robota przez procesy nadzorujące i w kontekście podejmowania decyzji w warunkach zmieniającej się rzeczywistości procesowej obejmującej modyfikowanie pracy robotów w środowisku rozproszonym korporacyjnych sieci IP w sposób mieszany – oparte o wiedzę ekspertów i podlegający automatyzacji.

W oparciu o powyższe wyzwania technologiczne opracowano szereg elementów zgodnych z założeniami badawczymi. W szczególności wytypowano listy wskaźników oraz

obiektywnych ich miar wraz z dobranymi wagami służącymi do obliczania potencjału robotyzacji procesów. Przeprowadzono rozpoznanie znanych języków programowania, zwłaszcza w dziedzinie opracowanych już języków dla robotyki tradycyjnej pod kątem kompletności i przydatności do zastosowania w robotyce wirtualnej. Opracowano założenia dla języka, zakresu jego stosowania oraz podstawowych elementów. W oparciu o powyższe opracowano szczegółowy ustrukturyzowany wyjściowy katalog poleceń wraz ze specyfikacją zachowania i wyników w odniesieniu do pozostałych założeń projektowych, który w szczegółach zaprezentowano w dalszej części niniejszego opracowania.

Zaplanowane prace badawcze w ramach niniejszego etapu badawczego zostały zakończone sukcesem – szczegółowe omówienie przedstawiono poniżej.

Lista kluczowych kryteriów do wyznaczania wskaźników związanych z oceną potencjału robotyzacji procesów

Informacje dotyczące klienta

Czy istnieje już strategia na automatyzację po stronie klienta?

Ile osób jest dedykowanych do procesu w FTE?

Za ile osób płaci klient?

Czy kontrakt jest FTE, transaction based czy Fixed Price?

Informacje dotyczące procesów

Co rozpoczyna cały proces?

Jaka jest częstotliwość wykonywania kroku?

Jaka jest jakość danych wejściowych?

Czy mamy już pomiary i mapy procesu, które są aktualne?

Jaka jest ilość transakcji?

Czy dane trafiają elektronicznie? Jeżeli nie, to w jaki sposób?

Czy dane są w postaci obrazka/ PDF?

Czy jest dokumentacja opisująca proces?

Jaka jest liczba błędów w miesiącu?

Czy proces jest wykonywany dla innych jednostek?

Czy proces jest zharmonizowany?

Czy proces jest rozbity między zespoły?

Czy proces jest rozbity między lokalizacje?

Czy proces jest rozbity między różnych partnerów/klienta?

Jak często zachodzą zmiany w procesie?

Czy istnieje proces zarządzania zmianą?

Lista kluczowych kryteriów do wyznaczania wskaźników związanych z oceną kroków w ramach robotyzacji procesów

Matryca wskaźników w rozbiciu na kroki procesu (ID Procesu, Numer Kroku, Opis Kroku)

W jakim narzędziu / aplikacji jest wykonywany krok?

Czy krok jest manualny czy automatyczny?

Czy krok jest oparty na zasadach czy podlega ocenie?

Czy krok ma jasno opisane zasady?

Czy krok wymaga manualnego potwierdzenia?

Czy krok jest ustandaryzowany per kraj/jednostka?

Ile jest błędów w miesiącu?

Czy krok jest wartością dodaną (VA)?

Czy krok jest audytowany?

Jaki jest CT (min)?

Jaki jest oczekiwany CT?

Typ Automatyzacji (BPA, RPA)

Liczba transakcji (dziennie)

Czy krok nadaje się do automatyzacji?

Oszczędzony czas per transakcja (min)

Oszczędzony czas (min)

Czy znamy i potrafimy użyć technologii by krok zautomatyzować?

Oszczędzony czas per transakcja (min)

Oszczędzony czas (min)

Komentarze

OGÓLNA SPECYFIKACJA JĘZYKA RFL

W ramach języka określonych zostało kilka rodzajów elementów. Wśród nich należy wyróżnić:

IStep - kroki. Są to najważniejsze elementy naszego języka. Reprezentują pojedyncze czynności, które możemy wykonać w ramach definicji konkretnego procesu biznesowego (np. kliknij przycisk, wpisz tekst, itd.).

IVariable - zmienna. Reprezentuje zmienną, która może być wykorzystywana przez elementy **IStep** w ramach procesu. Zmienne przekazywane są do elementów **IStep** poprzez kontekst (**IProcessContext**). Każdy element **IStep** może zmodyfikować/odczytać wartość przekazanej zmiennej.

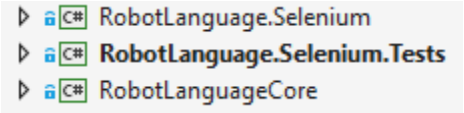
IProcessContext - kontekst uruchomienia. Jest to obiekt przekazywany pomiędzy poszczególnymi elementami **IStep**. Każdy element **IStep** może odczytać i zmodyfikować

zawartość kontekstu. Za pośrednictwem kontekstu elementy **IStep** mogą między sobą przekazywać informacje.

IElementSelector - selektor służący do określenia elementu/elementów na których działa dany element **IStep**. Zazwyczaj selektory stanowią parametry dla elementów **IStep.**

IAppSelector - selektor pozwalający na identyfikację aplikacji.

Poniżej zaprezentowany jest każdy z wymienionych wyżej elementów w dokładny sposób z pomocą szkicowych diagramów klas. Pamiętajmy, że gramatyka naszego języka jest w tym momencie oparta o interfejsy. Hierarchie przedstawionych interfejsów mogą być wielokrotnie zaimplementowane z wykorzystaniem różnych rodzajów sterowników (np. Selenium).

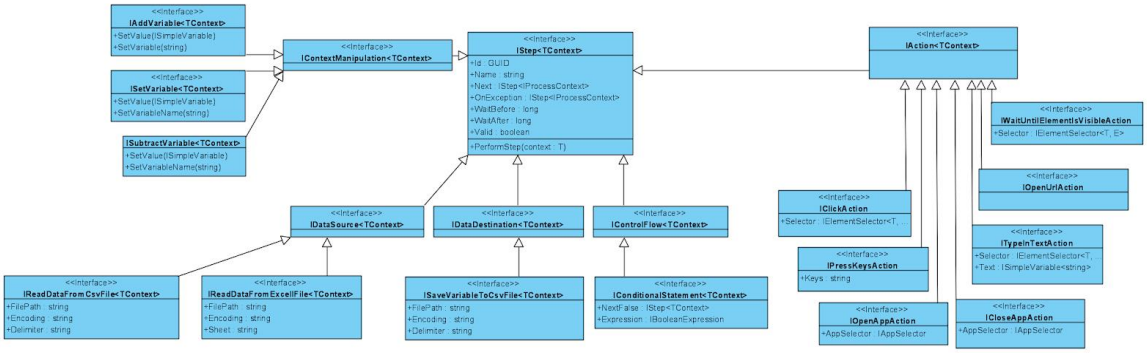


Powyżej przedstawiony jest drobny przykład hierarchii projektów typu “class library” w ramach pojedynczej solucji:

- RobotLanguageCore - schemat języka zdefiniowany w oparciu o abstrakcyjne elementy języka, czyli interfejsy;
- RobotLanguage.Selenium - projekt zawierający referencję do projektu “RobotLanguageCore” i stanowiący implementację języka dla aplikacji webowych w oparciu o bibliotekę Selenium;
- dodatkowe rozszerzalne implementacje zależne od bibliotek i środowiska uruchomienia (aplikacja webowa/desktopowa).

IStep - kroki

W tej części zdefiniowano najważniejsze elementy stanowiące bezpośrednie części języka robotycznego. Poniżej zaprezentowana jest dokładna hierarchia elementów **IStep** wraz z pewnym podstawowym podziałem.



Najważniejszym elementem powyższej hierarchii jest interfejs **IStep<TContext>.** Następnie następuje podział kroku na jedną z następujących kategorii:

IContextManipulation<TContext> - kroki związane z modyfikacją stanu kontekstu przekazywanego między elementami. Takie kroki mogą być związane np. ze zmianą wartości zmiennej przechowywanej w kontekście.

IDataSource<TContext> - kroki związane z pobieraniem danych z określonego źródła (np. odczyt z pliku CSV lub bazy danych).

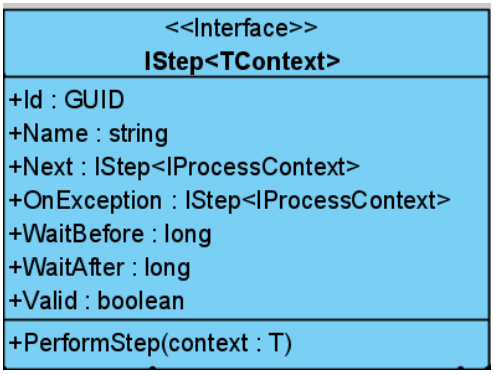
IDataDestination<TContext> - kroki związane z zapisaniem danych w konkretnej lokalizacji (np. zapis do pliku CSV).

IControlFlow<TContext> - element związane z kontrolą przepływu sterowania w aplikacji (np. instrukcja warunkowa).

IAction<TContext> - element określający interakcję z elementem interfejsu. Np. kliknięcie, skopiowanie tekstu itp. Na diagramie przedstawionych zostało kilka konkretnych rodzajów akcji.,

Każda z kategorii stanowi **punkt rozszerzenia** aplikacji, który pozwala na dodanie nowych elementów języka.

IStep<TContext> - opis elementu



Element przedstawiony powyżej obrazuje podstawowe elementy dostępne w każdym elemencie **IStep**.

Id - identyfikator elementu w ramach procesu.

Name - czytelna nazwa elementu (np. kliknięcie przycisku “Wyślij”).

Next - kolejne element do przetworzenia po bieżącym elemencie.

OnException - kolejny element **IStep** uruchamiany w momencie zajścia błędu.

WaitBefore - liczba milisekund do odczekania przed uruchomieniem logiki zaimplementowanej w danym elemencie.

WaitAfter - licznik milisekund do odczekania po uruchomieniu logiki zaimplementowanej w danym elemencie.

Valid - właściwość pozwalająca na sprawdzenie czy dany element jest poprawnie skonfigurowany.

PerformStep(context) - metoda pozwalająca na zaimplementowanie logiki biznesowej związaną z danym elementem.

IStep<TContext> - przykład interfejsu

Poniżej zaprezentowany jest prosty przykład implementacji interfejsu.

```
public interface IStep<T>
 {
      public Guid Id { get; set; }
      public string Name { get; set; }
      public IStep<T> Next { get; set; }
      public IStep<T> OnException { get; set; }
      public long WaitBefore { get; set; }
      public long WaitAfter { get; set; }
      public bool Valid { get; }
      public void PerformStep(T context);
 }
```

IStep<TContext>, IAction<TContext>, IClickAction<TContext> - przykład implementacji

W tym miejscu zaprezentowany jest przykład kodu reprezentujący konkretny rodzaj akcji - **IClickAction<TContext>** i jego naiwnej implementacji z wykorzystaniem biblioteki Selenium.

```
public interface IAction<TContext> : IStep<TContext>
{
}

public interface IClickAction<TContext, S> : IAction<TContext>
{
    public IElementSelector<TContext, S> Selector { get; set; }
}
```

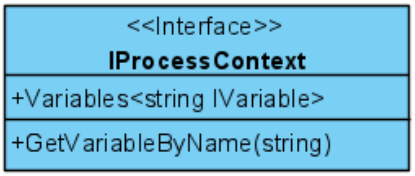
Poniżej przedstawiony jest przykład implementacji. Klasa implementuje interfejs **IClickAction**. Dwa parametry to klasa reprezentująca kontekst i obiekt zwracany przez selektor. Poniższy kod prezentuje w prosty sposób użycie parametrów **WaitBefore**, **OnException** itp.

```
public class ClickAction : IClickAction<ProcessContext, IEnumerable<IWebElement>>
{
   public Guid Id { get; set; }
   public string Name { get; set; }
   public IStep<ProcessContext> Next { get; set; }
```

```
    public IElementSelector<ProcessContext, IEnumerable<IWebElement>> Selector { get; set; }
    public IStep<ProcessContext> OnException { get; set; }
    public long WaitBefore { get; set; }
    public long WaitAfter { get; set; }

    public bool Valid => Selector != null && Id != null;

    public void PerformStep(ProcessContext context)
    {
        if (!Valid) throw new ElementNotConfiguredException($"Element with id={Id} is not
configured");
        try
        {
            if (WaitBefore > 0) Thread.Sleep((int)WaitBefore);

            var el = Selector.GetElement(context);

            if (el.Any())
            {
               el.First().Click();
            }

            if (WaitAfter > 0) Thread.Sleep((int)WaitAfter);
            if (Next != null) Next.PerformStep(context);
            }
            catch (Exception)
            {
                if (OnException != null)
                {
                    OnException.PerformStep(context);
                }
                else
                {
                    throw;
                }
            }
        }
    }
```

IProcessContext - kontekst

Kontekst reprezentuje stan procesu przekazywany między elementami IStep. Każdy element IStep może dodać/zmodyfikować/usunąć element z kontekstu.



Kontekst przechowuje słownik zmiennych (obiektów typu **IVariable**). Kluczem wykorzystywanym w słowniku jest wartość tekstowa nadana podczas konfiguracji procesu.

IVariable<G> - zmienna

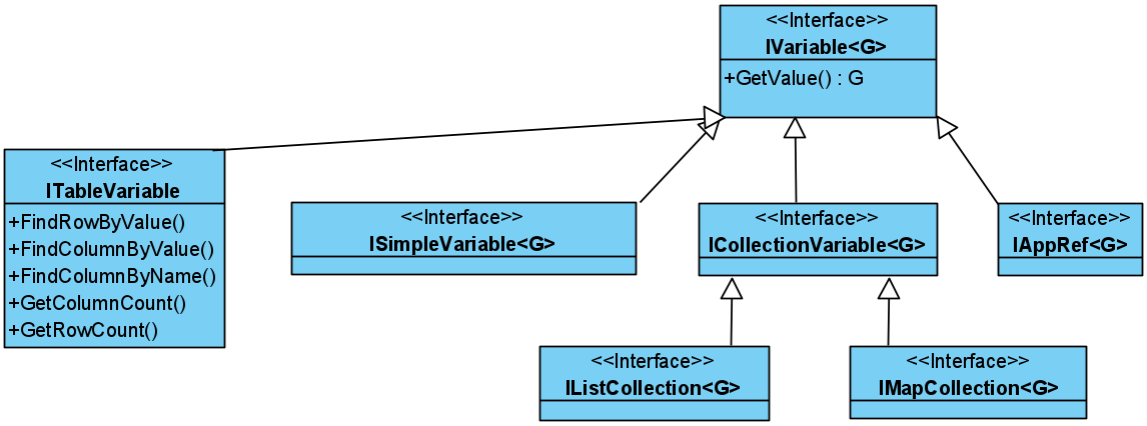
Element służy do reprezentacji zmiennej. Zmienne przekazywane są poprzez kontekst do elementów **IStep<T>.** Poniżej zaprezentowana jest hierarchia interfejsów reprezentująca zmienne. Przedstawiona hierarchia jest kolejnym punktem rozszerzenia i pozwala na dodanie kolejnych rodzajów zmiennych.

ISimpleVariable<G> - prosta zmienna. Reprezentuje takie dane jak: string, int, double, boolean

ICollectionVariable<G> - zmienna reprezentująca kolekcje.

IAppRef<G> - zmienna reprezentująca referencję.

ITableVariable - zmienna reprezentująca tablicę dwuwymiarową

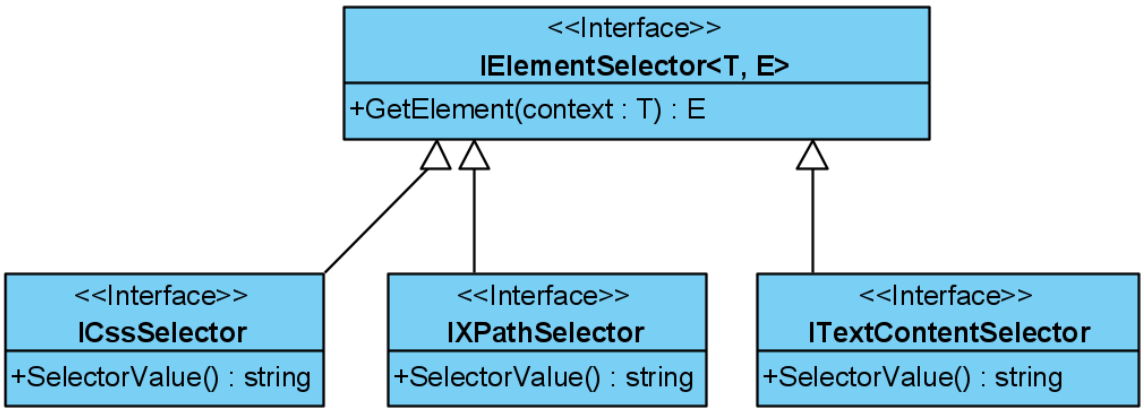


IElementSelector<TContext, E> - selektory

Poniżej zaprezentowany jest schemat interfejsów związany z selektorami. Selektory wykorzystywane są przez elementy **IAction<T>**. Np. element typu IClickAction<T> pozwalana na wykonanie "kliknięcia" na elemencie określonym przez element typu IElementSelector<T>.

Na ten moment selektory typu ICssSelector i IXPathSelector pokrywają dużą część potrzeb.

Poniżej zaprezentowane zostały trzy podstawowe selektory bazujące na: css, xpath i text content (zawartość tekstowa).



IElementSelector<TContext, E> - przykład implementacji

Poniżej zaprezentowany jest przykład prostej implementacji w oparciu o bibliotekę Selenium.

```
public interface IElementSelector<T, E>
{
```

```
    public E GetElement(T context);
}

public interface IXPathSelector<T, E> : IElementSelector<T, E>
{
    public string SelectorValue { get; set; }
}
```

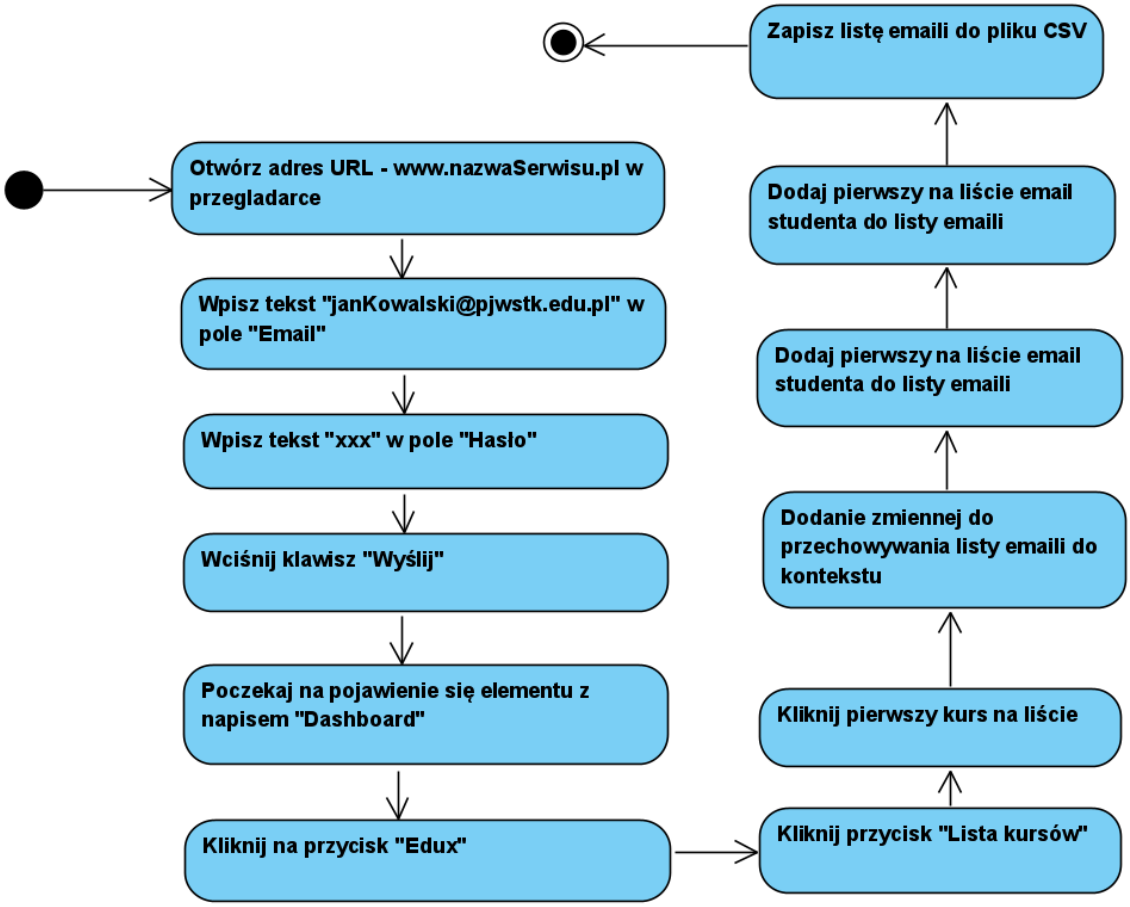
Poniżej zaprezentowana jest przykładowa implementacji przedstawionych powyżej interfejsów.

```
public class XPathSelector : IXPathSelector<ProcessContext,
IEnumerable<IWebElement>>
{
    public string SelectorValue { get; set; }

    public IEnumerable<IWebElement> GetElement(ProcessContext context)
    {
        return context.WebDriver.FindElementsByXPath(SelectorValue);
    }
}
```

Opis procesu biznesowego

W tej sekcji przedstawiono podstawowe elementy służące do opisu poszczególnych kroków procesu biznesowego. Zbiór niniejszych kroków stanowi podstawą, która może być rozszerzana w miarę potrzeb z zachowaniem podstawowych zasad określonych poniżej.

Procesy biznesowe możemy przedstawić jako **zestaw kroków**. Każdy krok reprezentuje pewną czynność - zazwyczaj interakcję między użytkownikiem i interfejsem. Poniżej zaprezentowany jest opis prostego **generycznego** procesu biznesowego z pomocą **diagramu aktywności**.



Przykład implementacji procesu w kodzie

Poniższy przykład pokazuje wykorzystanie zaimplementowanych elementów języka.

```
ProcessContext context = new ProcessContext();

//1. Wejście na stronę nazwaSerwisu.pl
OpenUrlAction action1 = new OpenUrlAction();
action1.Id = Guid.NewGuid();
action1.Name = "Wejdź na stronę nazwaSerwisu.pl";
action1.Url = "http://www.gakko.pjwstk.edu.pl";


//2. Wpisz email w pole 1
TypeInTextAction action2 = new TypeInTextAction();
action1.Next = action2;
action2.Name = "Wpisz adres email";
```

```
XPathSelector sel1 = new XPathSelector()
{
     SelectorValue = "//input[@id='userNameInput']"
};
action2.TextValue = new SimpleVariable<string>() { Value =
"xxx@pjwstk.edu.pl" };
action2.Selector = sel1;

//3. Wpisanie hasła
TypeInTextAction action3 = new TypeInTextAction();
action2.Next = action3;
action3.Name = "Wpisz hasło";


XPathSelector sel2 = new XPathSelector()
{
     SelectorValue = "//input[@id='passwordInput']"
};
action3.TextValue = new SimpleVariable<string>() { Value = "xxxx" };
action3.Selector = sel2;

//4. Wcieśnięcie klawisza Wyslij
ClickAction action4 = new ClickAction();
action3.Next = action4;
action4.Name = "Kliknij przycisk Submit";







XPathSelector sel3 = new XPathSelector()
{
     SelectorValue = "//span[@id='submitButton']"
};
action4.Selector = sel3;

//5. Poczekaj na pojawienie się napisu Dashboard
WaitUntilElementIsVisibleAction action5 = new
WaitUntilElementIsVisibleAction();
action4.Next = action5;
action5.Name = "Poczekaj na pojawienie się napisu Dashboard";

XPathSelector sel4 = new XPathSelector()
{
     SelectorValue = "//h3[contains(@class,'m-subheader__title')]"
};
action5.Selector = sel4;
```

```
//6. Kliknij na Edux
ClickAction action6 = new ClickAction();
action5.Next = action6;
action6.Name = "Kliknij na przycisk Edux";
action6.WaitAfter = 1000;


XPathSelector sel5 = new XPathSelector()
{
    SelectorValue = "//span[contains(text(),'Edux')]"
};
action6.Selector = sel5;

//7. Kliknij na Lista kursów
WaitUntilElementIsVisibleAction action7 = new
WaitUntilElementIsVisibleAction();
action6.Next = action7;
action7.Name = "Kliknij na przycisk Lista kursów";


XPathSelector sel6 = new XPathSelector()
{
     SelectorValue =
"/html[1]/body[1]/div[2]/div[1]/div[1]/div[1]/ul[1]/li[3]/div[1]/ul[1]
/li[1]/a[1]/span[1]"
};
action7.Selector = sel6;

//8. Click
ClickAction action8 = new ClickAction();
action7.Next = action8;
action8.Name = "Kliknij na przycisk Lista kursów";


XPathSelector sel7 = new XPathSelector()
{
     SelectorValue =
"/html[1]/body[1]/div[2]/div[1]/div[1]/div[1]/ul[1]/li[3]/div[1]/ul[1]
/li[1]/a[1]/span[1]"
};
action8.Selector = sel7;


//9. Click
ClickAction action9 = new ClickAction();
action8.Next = action9;
action9.Name = "Kliknij na przycisk Lista kursów";
```

```
XPathSelector sel8 = new XPathSelector()
{
       SelectorValue =
"/html[1]/body[1]/div[2]/div[1]/div[2]/div[2]/div[1]/div[1]/div[1]/div
[1]/div[2]/div[1]/div[1]/div[1]/div[2]/div[1]/table[1]/tbody[1]/tr[1]/
td[1]/a[1]"
};
action9.Selector = sel8;

//10/11. Dodaj liste emaili
AddVariable<ISet<string>> action11 = new AddVariable<ISet<string>>();
action9.Next = action11;
action11.Name = "Dodanie listy emaili";
action11.VariableName = "Emails";
action11.VariableValue = new HashSet<string>();

//12.
ClickAction action12 = new ClickAction();
action11.Next = action12;
action12.Name = "Kliknij studenci";


XPathSelector sel9 = new XPathSelector()
{
     SelectorValue =
"/html[1]/body[1]/div[2]/div[1]/div[1]/div[1]/ul[1]/li[4]/a[1]/span[1]
"
};
action12.Selector = sel9;

//13.
AddElementTextIntoVariable action13 = new
AddElementTextIntoVariable();
action12.Next = action13;
action13.Name = "Kliknij studenci";
action13.VariableName = "Emails";


XPathSelector sel10 = new XPathSelector()
{
       SelectorValue =
"/html[1]/body[1]/div[2]/div[1]/div[2]/div[2]/div[1]/div[1]/div[2]/div
[3]/table[1]/tbody[1]/tr[1]/td[4]/span[1]"
};
action13.Selector = sel10;

//14.
AddElementTextIntoVariable action14 = new
AddElementTextIntoVariable();
action13.Next = action14;
action14.Name = "Kliknij studenci";
```

```
action14.VariableName = "Emails";


XPathSelector sel11 = new XPathSelector()
{
      SelectorValue =
"/html[1]/body[1]/div[2]/div[1]/div[2]/div[2]/div[1]/div[1]/div[2]/div
[3]/table[1]/tbody[1]/tr[2]/td[4]/span[1]"
};
action14.Selector = sel11;

//15.
SaveVariableToCsvFile action15 = new SaveVariableToCsvFile();
action14.Next = action15;
action15.Name = "Zapisz do pliku CSV";
action15.VariableName = "Emails";

//Run the process
action1.PerformStep(context);
```

Powyższa lista kroków może być przedstawiona w łatwy sposób w formie dokumentu XML. Poniżej znajduje się kilka przykładów elementów języka wraz z reprezentacją w formie pliku XML.

Przykład:

```
<?xml version="1.0" encoding="utf-8" ?>
<process name="Praca z notatnikiem">
      <step id="4c0afed3-4c22-4fda-a86b-e8f8cee2864d" type="variable"
start="true" nextStep="66cee1b5-da0f-42b9-a6f5-5ed20ad21bfd">
            <!-- other elements related to this step -->
      </step>
      <step id="66cee1b5-da0f-42b9-a6f5-5ed20ad21bfd" type="variable">
            <!-- other elements related to this step -->
      </step>
      <!-- other steps -->
</process>
```

Powyższy przykład prezentuje **dwa kroki**. Oczywiście nie jest to kompletny proces. Element `step` stanowi pewną abstrakcją, która może oznaczać dowolny rodzaj akcji.

Szczegółowy opis wybranych akcji

ActionStep

Niniejsza kategoria elementów pozwala nam na modelowanie różnego rodzaju interakcji w ramach procesu biznesowego.

OpenAppAction

Element pozwalający na uruchomienie danej aplikacji. Zazwyczaj jest to jeden z pierwszych kroków. Po nim następują elementy związane z interakcją w ramach danej aplikacji.

Nazwa:	OpenAppAction
Przeznaczenie:	Element pozwala na uruchomienie aplikacji.
Elementy podrzędne	
<appSelector type="FilePath">C:\Users\app.exe</appSelector>	Element określający w jaki sposób możemy zlokalizować aplikację, którę chcemy uruchomić. Język można rozszerzyć o kolejne typy lokalizatorów.
<appSelector type="FilePath">{32e6c746-8715-48b5-b42d-6962f7a626d0}</appSelector>	Przykład po lewej obrazuje możliwość przekazania ścieżki poprzez zdefiniowaną wcześniej zmienną tekstową. **W nawiasach klamrowych przekazana jest wartość id elementu. W ten sposób możemy przekazać jako ścieżkę wartość ze zmiennej tekstowej.**
<targetVariable>{eab71a3d-3bbf-4e20-8b9f-b89000633deb}</targetVariable>	Element pozwala na zapisanie referencji do otwartej aplikacji w zmiennej typu “appReference” wskazanej przez wartość jej id.
Atrybuty	
id	**Atrybut wymagany.** Reprezentuje identyfikator

	elementu. Wykorzystywany do nawigacji pomiędzy poszczególnymi elementami. Zapisywany zazwyczaj jako wartość zgodna z formatem GUID.
type	**Atrybut wymagany. W tym wypadku ustawiony na wartość "action".**
subType	**Atrybut wymagany. W tym wypadku jego wartość ustalona jest na "openApp".**
nextStep	**Atrybut wymagany z wyjątkiem ostatniego kroku w przepływie i instrukcji warunkowej.**
name	**Atrybut opcjonalny.** Pozwala na nadanie czytelnej nazwy dla danego kroku w procesie biznesowym.
start	**Atrybut opcjonalny.** Wykorzystywany w celu oznaczenia pierwszego elementu w naszym procesie biznesowym.
onException	**Atrybut opcjonalny.** Pozwala na oprogramowanie wyjątku w sytuacji kiedy wystąpił w trakcie przetwarzania niniejszego kroku.
waitBefore	**Atrybut opcjonalny.** Liczba milisekund do odczekania przed uruchomieniem logiki związanej z danym krokiem.
waitAfter	**Atrybut opcjonalny.** Liczba milisekund do odczekania po uruchomieniu logiki związanej z danym krokiem.
valid	**Atrybut wymagany.** Wartość logiczna pozwalająca sprawdzić czy dany element jest poprawnie skonfigurowany (wszystkie wymagane parametry zostały określone).

Przykłady:

```
<step id="b5bfb6b4-c91a-4048-9dce-5d83d9825dd0" type="action"
subType="openApp" next="027d05ec-6cc2-4691-b6d5-38882f6e7a49">
      <appSelector type="FilePath">C:\Users\app.exe</appSelector>
<targetVariable>{eab71a3d-3bbf-4e20-8b9f-
b89000633deb}</targetVariable>
</step>

<step id="b5bfb6b4-c91a-4048-9dce-5d83d9825dd0" type="action"
subType="openApp" next="027d05ec-6cc2-4691-b6d5-38882f6e7a49">
      <appSelector type="FilePath">{32e6c746-8715-48b5-b42d-
6962f7a626d0}</appSelector>
<targetVariable>{eab71a3d-3bbf-4e20-8b9f-
b89000633deb}</targetVariable>
</step>
```

CloseAppAction

Element pozwalający na zamknięcie danej aplikacji/procesu.

Nazwa:	CloseAppAction
Przeznaczenie:	Element pozwala na zamknięcie wybranej aplikacji.
Elementy podrzędne	
<appSelector type="AppReference">{32e6c746-8715-48b5-b42d-6962f7a626d0}</appSelector>	**Element wymagany.** Pozwala na określenie aplikacji którą chcemy zamknąć. Wartość powinna wskazywać na zmienną typu “AppReference”.
Atrybuty	
id	**Atrybut wymagany.** Reprezentuje identyfikator elementu. Wykorzystywany do nawigacji pomiędzy poszczególnymi elementami. Zapisywany zazwyczaj jako wartość zgodna z formatem GUID.

type	**Atrybut wymagany. W tym wypadku ustawiony na wartość "action".**
subType	**Atrybut wymagany. W tym wypadku jego wartość ustalona jest na "closeApp".**
nextStep	**Atrybut wymagany z wyjątkiem ostatniego kroku w przepływie i instrukcji warunkowej.**
name	**Atrybut opcjonalny.** Pozwala na nadanie czytelnej nazwy dla danego kroku w procesie biznesowym.
start	**Atrybut opcjonalny.** Wykorzystywany w celu oznaczenia pierwszego elementu w naszym procesie biznesowym.
onException	**Atrybut opcjonalny.** Pozwala na oprogramowanie wyjątku w sytuacji kiedy wystąpił w trakcie przetwarzania niniejszego kroku.
waitBefore	**Atrybut opcjonalny.** Liczba milisekund do odczekania przed uruchomieniem logiki związanej z danym krokiem.
waitAfter	**Atrybut opcjonalny.** Liczba milisekund do odczekania po uruchomieniu logiki związanej z danym krokiem.
valid	**Atrybut wymagany.** Wartość logiczna pozwalająca sprawdzić czy dany element jest poprawnie skonfigurowany (wszystkie wymagane parametry zostały określone).

CalculateAction

Element pozwala na obliczenie wyrażenia matematycznego i zapisanie rezultatu do innej zmiennej.

Nazwa:	CalculateAction
Przeznaczenie:	Element pozwala na zamknięcie wybranej aplikacji.

Elementy podrzędne	
<expression> <add> <value>{023...}</value> <value>{027...}</value> <multiply> <value>{02....}</value> <value>5</value> </multiply> </add> </expression>	**Element wymagany.** Pozwala na określenie obliczanego wyrażenie. Wyrażamy je za pomocą elementów: <add>, <subtract>, <multiply>, <divide>. Następnie elementy <value> wskazują na zmienną lub konkretną wartość, którą np. Dodajemy. Poniżej znajduje się kilka przykładów wyrażeń.
<targetVariable>{GUID zmiennej 4}</targetVariable>	**Element wymagany.** Określa zmienną do której chcemy zapisać rezultat kalkulacji.
Atrybuty	
id	**Atrybut wymagany.** Reprezentuje identyfikator elementu. Wykorzystywany do nawigacji pomiędzy poszczególnymi elementami. Zapisywany zazwyczaj jako wartość zgodna z formatem GUID.
type	**Atrybut wymagany. W tym wypadku ustawiony na wartość "action".**
subType	**Atrybut wymagany. W tym wypadku jego wartość ustalona jest na "calculationAction".**
nextStep	**Atrybut wymagany z wyjątkiem ostatniego kroku w przepływie i instrukcji warunkowej.**
name	**Atrybut opcjonalny.** Pozwala na nadanie czytelnej nazwy dla danego kroku w procesie biznesowym.
start	**Atrybut opcjonalny.** Wykorzystywany w celu oznaczenia pierwszego elementu w naszym procesie biznesowym.
onException	**Atrybut opcjonalny.** Pozwala na oprogramowanie wyjątku w sytuacji kiedy wystąpił w trakcie przetwarzania niniejszego kroku.
waitBefore	**Atrybut opcjonalny.** Liczba milisekund do odczekania przed uruchomieniem logiki związanej z danym krokiem.
waitAfter	**Atrybut opcjonalny.** Liczba milisekund do odczekania po uruchomieniu logiki związanej z danym krokiem.

valid	**Atrybut wymagany.** Wartość logiczna pozwalająca sprawdzić czy dany element jest poprawnie skonfigurowany (wszystkie wymagane parametry zostały określone).

Przykład:

```
<step id="b5bfb6b4-c91a-4048-9dce-5d83d9825dd0" type="action"
subType="calculation" next="027d05ec-6cc2-4691-b6d5-38882f6e7a49">
      <expression>
            <add>
                  <value>{027d05ec-6cc2-4691-b6d5-38882f6e7a49}</value>
                  <value>{027d05ec-6cc2-4691-b6d5-38882f6e7a49}</value>
                  <multiply>
                        <value>{027d05ec-6cc2-4691-b6d5-
38882f6e7a49}</value>
                        <value>5</value>
                  </multiply>
            </add>
      </expression>
      <targetVariable>{eab71a3d-3bbf-4e20-8b9f-
b89000633deb}</targetVariable>
</step>
```

Przykład kalkulacji: (zmienna1+10)/zmienna3 = zmienna4

```
<step id="b5bfb6b4-c91a-4048-9dce-5d83d9825dd0" type="action"
subType="calculation" next="027d05ec-6cc2-4691-b6d5-38882f6e7a49">
      <expression>
            <divide>
                  <add>
                        <value>{GUID zmiennej 1}</value>
                        <value>10</value>
                  </add>
                  <value>{GUID zmiennej 3}</value>
            </divide>
      </expression>
      <targetVariable>{GUID zmiennej 4}</targetVariable>
</step>
```

Zwróćmy uwagę na **zagnieżdżenia** między elementami. Ponadto **kolejność elementów** <value> zależnie od rodzaju operacji **może być istotna**.

Elementy <add>, <subtract>, <multiply>, <divide>, <value> mogę być zagnieżdżane między sobą co pozwala na tworzenie bardziej złożonych wyrażeń.

ClickAction

Element pozwala na wykonanie kliknięcia na wybranym elemencie.

Nazwa:	ClickAction
Przeznaczenie:	Element pozwala na zamknięcie wybranej aplikacji.
Elementy podrzędne	
<appSelector type="AppReference">{32e6c746-8715-48b5-b42d-6962f7a626d0}</appSelector>	**Element wymagany.** Wskazuje na element typu “AppReference”. Określa aplikację w ramach której wykonujemy kliknięcie.
<elementSelector type="css">div.title</elementSelector>	**Element wymagany.** Pozwala na określenie elementu, który chcemy kliknąć. Można wyróżnić kilka typów selektora: **css, xpath, name, id.** Oczywiście podczas implementacji można rozwinąć wspomniany element implementując kolejne wymagane typy selektorów.
Atrybuty	
id	**Atrybut wymagany.** Reprezentuje identyfikator elementu. Wykorzystywany do nawigacji pomiędzy poszczególnymi elementami. Zapisywany zazwyczaj jako wartość zgodna z formatem GUID.
type	**Atrybut wymagany. W tym wypadku ustawiony na wartość “action”.**
subType	**Atrybut wymagany. W tym wypadku jego wartość ustalona jest na “click”.**

nextStep	**Atrybut wymagany z wyjątkiem ostatniego kroku w przepływie i instrukcji warunkowej.**
name	**Atrybut opcjonalny.** Pozwala na nadanie czytelnej nazwy dla danego kroku w procesie biznesowym.
start	**Atrybut opcjonalny.** Wykorzystywany w celu oznaczenia pierwszego elementu w naszym procesie biznesowym.
onException	**Atrybut opcjonalny.** Pozwala na oprogramowanie wyjątku w sytuacji kiedy wystąpił w trakcie przetwarzania niniejszego kroku.
waitBefore	**Atrybut opcjonalny.** Liczba milisekund do odczekania przed uruchomieniem logiki związanej z danym krokiem.
waitAfter	**Atrybut opcjonalny.** Liczba milisekund do odczekania po uruchomieniu logiki związanej z danym krokiem.
valid	**Atrybut wymagany.** Wartość logiczna pozwalająca sprawdzić czy dany element jest poprawnie skonfigurowany (wszystkie wymagane parametry zostały określone).

Przykłady:

```
<step id="b5bfb6b4-c91a-4048-9dce-5d83d9825dd0" type="action"
subType="click" next="027d05ec-6cc2-4691-b6d5-38882f6e7a49">
      <appSelector type="AppReference">{32e6c746-8715-48b5-b42d-
6962f7a626d0}</appSelector>
      <elementSelector type="css">div.title</elementSelector>
</step>
```

ConditionalFlowControl

Element pozwala nam na kontrolowanie przepływu sterowania w naszym procesie. Z jego pomocą możemy symulować pętle, instrukcje warunkowe itd.

Nazwa:	ConditionalFlowControl
Przeznaczenie:	Element pozwala na warunkowe przejście do jednego z dwóch elementów na podstawie warunku logicznego.
Elementy podrzędne	
<conditions> <not> <and> <left>{GUID zmiennej}</left> <right>25</right> </and> </not> </conditions>	**Element wymagany.** Pozwala na zbudowanie złożonego warunku logicznego. Pozwala na użycia elementów <and>, <or>, <not>, <gt>, <lt>, <gte>, <lte>, <eq>. Wszystkie elementy z wyjątkiem <not> wymagają dwóch argumentów - <left> i <right>. Element <not> wymaga wyłącznie jednego.
<onTrue>{GUID innego elementu}</onTrue>	**Element wymagany.** Element pozwala na określenie jaki element ma zostać uruchomiony w przypadku ewaluacji warunków logiczny do wartości “true”.
<onFalse>{GUID innego elementu}</onFalse>	**Element wymagany.** Element pozwala na określenie jaki element ma zostać uruchomiony w przypadku ewaluacji warunków logiczny do wartości “false”.
Atrybuty	
id	**Atrybut wymagany.** Reprezentuje identyfikator elementu. Wykorzystywany do nawigacji pomiędzy poszczególnymi elementami. Zapisywany zazwyczaj jako wartość zgodna z formatem GUID.
type	**Atrybut wymagany. W tym wypadku ustawiony na wartość “flowControl”.**
subType	**Atrybut wymagany. W tym wypadku jego wartość ustalona jest na “conditionalStatement”.**
nextStep	**Atrybut wymagany z wyjątkiem ostatniego kroku w przepływie i instrukcji warunkowej.**
name	**Atrybut opcjonalny.** Pozwala na nadanie czytelnej nazwy dla danego kroku w procesie biznesowym.
start	**Atrybut opcjonalny.** Wykorzystywany w celu oznaczenia pierwszego elementu w naszym procesie biznesowym.

onException	**Atrybut opcjonalny.** Pozwala na oprogramowanie wyjątku w sytuacji kiedy wystąpił w trakcie przetwarzania niniejszego kroku.
waitBefore	**Atrybut opcjonalny.** Liczba milisekund do odczekania przed uruchomieniem logiki związanej z danym krokiem.
waitAfter	**Atrybut opcjonalny.** Liczba milisekund do odczekania po uruchomieniu logiki związanej z danym krokiem.
valid	**Atrybut wymagany.** Wartość logiczna pozwalająca sprawdzić czy dany element jest poprawnie skonfigurowany (wszystkie wymagane parametry zostały określone).

Przykład:

Warunek: !(((guid1==10 or guid2>25) and guid3<=guid4))

```
<step id="4c0afed3-4c22-4fda-a86b-e8f8cee2864d" type="flowControl"
subType="conditionalStatement">
      <conditions>
         <not>
            <and>
               <left>
                  <or>
                     <left>
                        <eq>
                           <left>{GUID zmiennej}</left>
                           <right>10</right>
                        </eq>
                     </left>
                     <right>
                        <gt>
                           <left>{GUID zmiennej}</left>
                              <right>25</right>
                        </gt>
                     </right>
                  </or>
               </left>
               <right>
```

```
                <lte>
                   <left>{GUID zmiennej}</left>
                   <right>{GUID zmiennej}</right>
                </lte>
             </right>
          </and>
       </not>
    </conditions>
    <onTrue>{GUID innego elementu}</onTrue>
    <onFalse>{GUID innego elementu}</onFalse>
</step>
```

Zaprezentowane powyżej szczegóły dotyczące osiągniętych kamieni milowych w postaci opracowanych miar, wskaźników i narzędzi do oceny potencjału automatyzcji wraz z opracowaniem specyfikacji technicznej języka programowania robotów pozwoliło osiągnąć cele badawcze do dalszego zastosowania w pracach badawczych i rozwojowych, w szczególności w komplementarnym etapie prac rozwojowych (ER1).

EB2: Opracowanie metod automatyzacji dostosowywania robotów do zmiennych danych wejściowych i środowiska w oparciu o metody AI

W ramach drugiego etapu badawczego (EB2) zostały wybrane rodzaje algorytmów dopasowane do domeny dziedzinowej problemu badawczego, w oparciu o postulowane opisy cech klasyfikacyjnych, wag oraz modeli uczenia wykonano testy w warunkach laboratoryjnych. Opracowano założenia do modelu wprowadzania zmian oraz oceny wyników oraz technicznego PoC (proof of concept) dla typowego procesu biznesowego z zakresu procesów klientów DPC.

Jednym z podstawowych wyzwań tego etapu było opracowanie mechanizmu skutecznego reagowania na zmieniające się dane wejściowe poprzez zapewnienie tolerancji w ustalonych zakresach. Przy pomocy metod eksperckich wspartych rozwiązaniami z zakresu uczena maszynownego w ujęciu CI ML wypracowano sposób na akceptowanie różnorodnych rodzajów danych, obejmujących m.in. różne formaty dokumentów skanowanych do systemu powiązane z ich klasyfikowaniem i podejmowaniem decyzji odnośnie dalszego przetwarzania. Powiązanym wyzwaniem było odpowiednie dobranie wag, wskaźników i progów reagowania wspierających automatyzację, w szczególności w ramach środowiska wykonawczego w kontekście rozpoznawania i klasyfikowanie zmian, tak aby umożliwić skonstruowanie technicznego PoC (Proof of Concept) – szczegółowy opis funkcjonalności zamieszczono w dalszej części opracowania wraz z podsumowaniem testów rozwiązania przeprowadzonych w ramach niniejszego etapu prac badawczych. Na podstawie tak sklasyfikowanych i opisanych zmian w środowisku wykonawczym robota opracowano skuteczne metody na zmianę w zachowaniu robota tak, by w nieprzerwany sposób mógł kontynuować pracę w procesie obejmujących m.in. utworzenie dodatkowych punktów decyzyjnych, miejsc pobierania danych i klasyfikacji wyników działania robota.

W oparciu o powyższe wyzwania technologiczne opracowano szereg elementów zgodnych z założeniami badawczymi, w szczególności w zakresie oceny potencjału różnych metod automatyzacji w odniesieniu do postawionego problemu oraz opracowania właściwego zestawu cech pod kątem automatyacji dla tak postawionych problemów.

Zaplanowane prace badawcze w ramach niniejszego etapu badawczego zostały zakończone sukcesem – szczegółowe omówienie przedstawiono poniżej.

Opis głównych funkcjonalności dziedziowych

W ramach niniejszego etapu przygotowano poniższy szczegółowy opis funkcjonalności dziedziowej podsystemu robotycznego w odniesieniu do systemu **rozpoznawania dokumentów finansowych** na przykładzie dokumentów i danych fakturowych.

Aplikacja na wejściu jest w stanie rozpoznać typy dokumentów (bazowych plików):

PDF, SKAN.

Poza scopem dziedziowym: pliki (skany, pdf) z pismem ręcznym.

Formaty dokumentów:

Dowolny format, przy czym najbardziej oczekiwanym formatem jest A4.

Rodzaj pisma treści dokumentów:

Pismo komputerowe – wszystkie czcionki/fonty biznesowe (Arial, Times etc.),

Pismo maszynowe – wszystkie czcionki/fonty biznesowe (Arial, Times etc.).

Poza scopem dziedzinowym: Pismo ręczne

Liczba stron:

Dowolna liczba stron (uwaga: w środowisku testowym w fazie PoC, uwagi na specyfikę procesów badawczych nie więcej niż 10 stron w 1 pliku; w fazie prototypu stopniowe zwiększenie limitu).

Po przesłaniu zeskanowenego pliku do aplikacji robotycznej w wymaganym formacie następuje pierwszy najważniejszy etap procesu:

Rozpoznanie wartości pól przez algorytm, po którym rodzaj dokumentu wyszukuje definiowany szablon i schemat do dalszego procesowania i ekstrakcji danych z dokumentu.

Proces ekstrakcji danych na przykładzie typu dokumentu fakturowego obejmuje następujące kluczowe elementy:

Pola obiekty tekstowe,

Pola obiekty liczbowe,

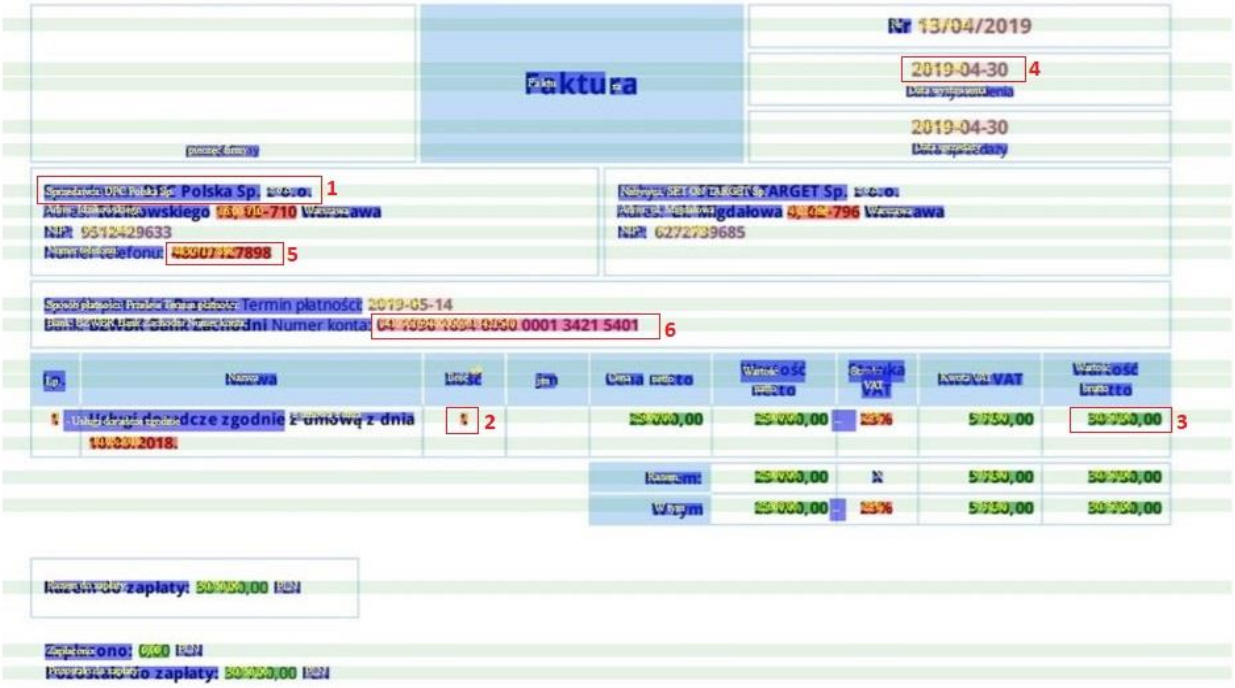
Pola obiekty kwotowe,

Pola obiekty dat - w wymaganych formatach,

Pola obiekty formatu NIP,

Pola obiekty numeru rachunku bankowego.

Poniżej przedstawiono **przykład rozpoznanych przez algorytm danych** z pliku zawierającego fakturę w ramach przygotowanego w toku prac badawczych **PoC (Proof of Concept)**. Zaprezentowany przykład ilustruje rozmieszczenie pól obiektowych na dokumencie (pliku) rozpoznanym przez algorytm.



W ramach PoC plik z fakturą otrzymuje status w aplikacji o treści: *Przetworzony i przygotowany do rozpoznania* – stanowi zbiór ciągów co do którego nie podjęto decyzji wykorzystania szablonu i rozpoznania treści dokumentu.

W kolejnym kroku po rozpoznaniu wszystkich pól jakie dokonał algorytm w ramach PoC dla przesłanego dokumentu (pliku) następuje budowanie szablonu według rodzaju dokumentu lub rodzaj dokumentu zostaje rozpoznany już według istniejących szablonów obejmujących elementy i dane statyczne oraz dynamiczne.

Warunki spełniające rozpoznanie i ekstrakcje danych przez algorytm dla pól obiektowych:

Dane statyczne:

- *Pola obiekty tekstowe* - dowolna treść zawierająca ciąg wyrazów, znaków diakrytycznych, znaków specjalnych.

- Numer dokumentu - dowolna treść zawierająca ciąg cyfr, wyrazów, znaków diakrytycznych, znaków specjalnych.

- Dane nabywcy i sprzedawcy - dowolna treść zawierająca ciąg cyfr, wyrazów, znaków diakrytycznych, znaków specjalnych, polskich znaków (odchylenie nierozpoznania polskich znaków ok. 1-5%).

- *Pola obiekty liczbowe* - dowolna treść zawierająca ciąg cyfr.

- *Pola obiekty dat* – algorytm rozpoznaje poniższe rodzaje formatów:

"yyyy-MM-dd",
"yyyy.MM.dd",
"yyyy/MM/dd",
"dd-MM-yyyy",
"dd.MM.yyyy",
"dd/MM/yyyy",
"dd.MM.yy",
"dd/MM/yy".

- *Pola obiekty formatu NIP* - ciąg 10 cyfr, aplikacja jest w stanie rozpoznać czy NIP jest prawidłowy, wychwycić jego niepoprawność jeśli został źle napisany, gdy nr NIP nie istnieje. Rozpoznaje rodzaj NIP zapisany w formacie przed szeregiem 10 cyfr z wyrazem i bez "PL".

- *Pola obiekty numeru rachunku bankowego* - ciąg 26 cyfr rozpoznaje wszystkie rodzaje kont bankowych w Polsce. Wskazuje niepoprawność jeśli nie istnieje takie konto bankowe.

Dane dynamiczne:

- *Pola obiekty kwotowe (księgowe)* - treść w polach zawierająca cyfry i formę wyliczeń (dziesiętne, setne, tysięczne, milionowe etc.) w polach tabel oraz w polach podsumowania księgowego w opisie faktury m.in. suma netto, suma brutto, suma VAT, Razem inne. Algorytm wskaże nieprawidłowości w wyliczeniach jeśli wystąpią.

Dane tabelaryczne:

- Pola obiekty w nagłówku oraz we wszystkich kolumnach tabeli oraz treść pod tabelą (patrz pkt. 7).

Automatyzacja procesu budowania szablonów

W ramach automatyzacji zadań rozpoznawania dokumentów PoC uwzględnia, że w zakładce *Pliki bez szablonów* znajdują się wszystkie pliki przetworzone i odebrane prawidłowo przez system aplikacji, które nie są jeszcze przypisane do konkretnego szablonu, ale mogą służyć do stworzenia nowego szablonu lub dopasowania jeśli istnieje już utworzony szablon a także umożliwiają ręczną akceptacji dokumentu przez użytkownika.

Zgodnie z poniższym widokiem struktura zawiera odpowiednio:

w pierwszej kolumnie: Nazwy Plików, w drugiej Data dodania do systemu, kolejno Nadawca, czwarta kolumna to Status pliku, który dzieli się na dwa o treści:

W ramach PoC pierwszy status wyświetla się po przetworzeniu pliku przez aplikacje o treści: „Przetworzony i przygotowany do rozpoznania - rozebrany na słowa i jeszcze nie podjęto szablonu i rozpoznania treści dokumentu”; drugi status w momencie kiedy tworzony jest szablon i nastąpi próba dopasowania

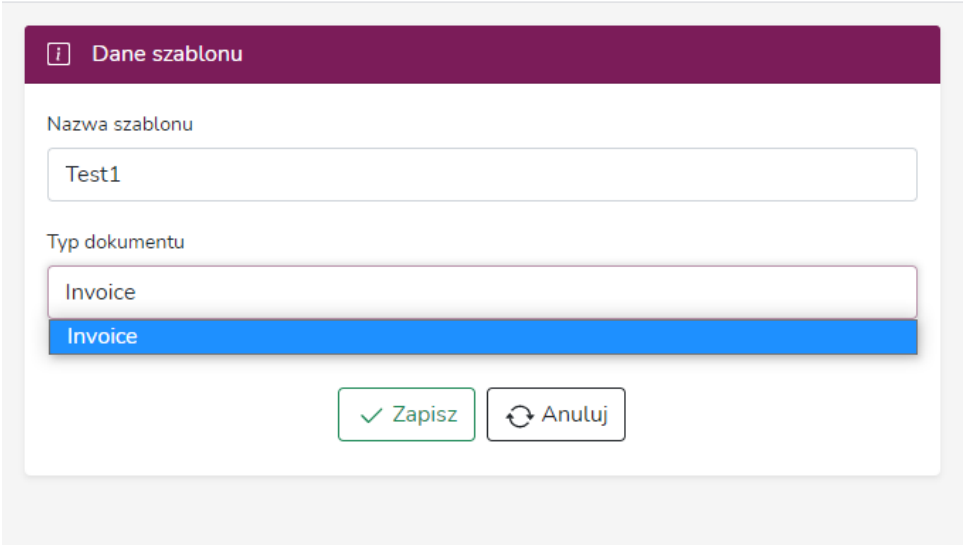
W celu ułatwienia nadzorowania wyników pracy aplikacji pliki ułożone są chronologicznie według daty dodania do systemu od najpóźniejszej do najwcześniejszej wyświetlane od góry.

Na zaprezentowanym powyżej widoku PoC, po prawej górnej stronie paska menu znajduje się przycisk „Filtruj” po jego naciśnięciu wyświetla się nowe okno z możliwością wyfiltrowania danych jakie potrzebujemy w tej zakładce.

Automatyzacja tworzenia szablonu

W ramach PoC można wybrać plik dla którego można przyporządkować lub stworzyć szablon klikając na przycisk w aplikacji *Stwórz/dopasuj szablon*.

W polu Dane szablonu, wpisujemy opis skrócony Nazwa szablonu i wybieramy Typ szablonu. Po prawej stronie wyświetla się ekran „Dane szablonu” w pustym polu o treści: „Nazwa szablonu” wpisujemy ręcznie nazwę dla nowego szablonu jaki chcemy utworzyć. Po wykonaniu tej akcji klikamy przycisk Zapisz i szablon zostaje zapisany, w prawym górnym rogu otrzymujemy komunikat wyświetlany na zielono, że „Szablon został poprawnie zapisany”. Jeśli klikniemy przycisk Anuluj wszystkie zmiany zostaną zaniechane i niezapisane.



Widok zakładki: Dane szablonu.

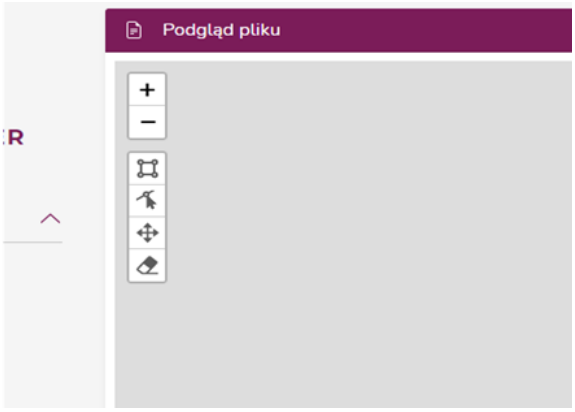
Responsywność widoku pliku.

Z lewej strony wyświetla się widok „Podgląd pliku”, pod paskiem znajduje się znak + i – służą one do przybliżania i oddalania pliku wyświetlanego się w polu z widokiem, czynność tą użytkownik może również wykonywać za pomocą myszki interakcji zewnętrznej, najeżdżając kursorem na pole dokumentu i sterując myszką góra/dół, dokument odpowiednio się przybliża i oddala jest responsywny względem wymagań użytkownika oraz jego posiadanego sprzętu na którym odczytuje widok aplikacji.

W ramach PoC Aplikacja pozwala na maksymalne przybliżenie oraz oddalenie podglądu pliku. Jeżeli dokument jest wielostronicowy można po prawej stronie paska wybrać stronę dla której chcemy przenieść dane. Jeżeli plik będzie posiadał na przykład 32 strony po rozwinięciu paska wyświetli się poprawna ilość stron dokumentu.

Narzędzie do tworzenia szablonu „Template Creator”.

Z lewej strony na ekranie PoC „Podgląd pliku” pod znacznikami +/- znajduje się narzędzie do tworzenia szablonu. Aby stworzyć szablon musimy ręcznie narysować za pomocą kreatora odpowiednie pola ramki i przypisać im odpowiednie pola dla wybranych wartości jakie chcemy otrzymać z dokumentu (pliku).



Widok narzędzia do tworzenia szablonu „Template Creator”.

Korekta i tworzenie ramek odpowiednich pól dla szablonu.

Pierwszy znacznik służy do oznaczenia pól ramek i określonych pól w ramkach dla przypisania odpowiednich wartości. Klikając oznaczamy pole ramki, które chcemy wykorzystać do stworzenia szablonu pole podświetla się na niebiesko, aby zakończyć to pole klikamy w miejscu, które chcemy zakończyć dany obiekt. Pola ramek dzielimy na pole statyczne, tabelaryczne, inne wartości na dokumencie pod tabelą (pływające).

Do utworzenia szablonów na podstawie rozpoznanych danych w ramach PoC możemy zastosować typy dokumentów (plików):

- Faktury,
- Faktury korygujące,
- Duplikaty faktur,
- Noty księgowe,
- Noty odsetkowe,
- Rachunki,
- Paragony lub dowody kasowe,
- Dokumenty celne,
- Dowody opłat bankowych, pocztowych,
- Dowody przesunięć,
- Czeki, weksle,
- Zamówienia,
- Umowy,
- Wszystkie inne typy dokumentów zawierające powyższe wymagania dla aplikacji.

Szczegóły PoC w zakresie rozpoznania wszystkich wartości pól z tabel:

- Posiadające różne typy wartości w kolumnach z nieograniczoną liczbą kolumn w tabeli dla 1 i więcej ilości stron tabeli:

Lp.	Symbol	GTU	Nazwa towaru/usługi	Ilość	jm.	Cena netto	Cena brutto	VAT (%)	Wartość netto	Wartość brutto
1.	210.807.700		PLYN D/SPRYS SL -22 METANOL CZERW.zima	16	SZT	5.70	7.01	23	91.20	112.18
2.	K177		AKRA 770 ATOM	12	SZT	5.90	7.26	23	70.80	87.08
3.	02.180.200	03	OLEJ PLATINUM 10/40 1L.	12	SZT	9.58	11.78	23	114.96	141.40
4.	02.180.220	03	OLEJ PLATINUM 10/40 4,5L.	3	SZT	40.67	50.02	23	122.01	150.07

- Opis faktury – specyfikacja rodzaju dokumentu w tabeli. Każdy opis zawiera się w jednej linijce danego wiersza, gdzie występują wartości jedna pod drugą:

Lp.	Symbol	GTU	Nazwa towaru/usługi	Ilość	jm.	Cena netto	Cena brutto	VAT (%)	Wartość netto	Wartość brutto
1.	210.806.785	03	OLEJ HIPOL GL-4 80W 5L	8	SZT	28.00	34.44	23	224.00	275.52
2.	210.806.900/2	03	OLEJ HYDRAUL.HL-46 5L.	8	SZT	26.74	32.89	23	213.92	263.12
3.	210.805.511	03	OLEJ MASZYNOWY PILAK EKO 5L	8	SZT	22.00	27.06	23	176.00	216.48
4.	210.801.555.	03	VALVOLINE SYNPOW MST 5/40 C31L	9	SZT	24.95	30.69	23	224.55	276.20

- Dane w tabeli muszą znajdować się na tej samej wysokości co typ wartości dla danej kolumny:

PKWiU	Ilość	j.m.	Rabat [%]	Cena netto	VAT [%]	Wartość netto	VAT	Wartość brutto
	5,000	szt.	0,00	7,85	23	39,25	9,03	48,28
	1,000	szt.	0,00	110,00	23	110,00	25,30	135,30
	1,000	kg.	0,00	58,00	23	58,00	13,34	71,34

- Dane w tabeli są rozmieszczone w kilku linijkach:

Lp	Nazwa towaru lub usługi	Symbol PKWiU	J.m.	Ilość towaru	Cena jednostkowa netto	Cena jednostkowa brutto	Wartość towaru usługi bez podatku	Stawka VAT (%)	Kwota podatku VAT	Wartość towaru usługi z podatkiem
1	Usługa SSL - abonament (01.03.2021-31.03.2021)		szt.	2	10		20,00	23%	4,60	24,60
2	Dzierżawa - terminal przenośny (01.03.2021-31.03.2021)		szt.	1	29		29,00	23%	6,67	35,67
3	Dzierżawa - terminal stacjonarny (01.03.2021-31.03.2021)		szt.	2	40		80,00	23%	18,40	98,40
4	Serwis GPRS (01.03.2021-31.03.2021)		szt.	1	5		5,00	23%	1,15	6,15
						Razem:	134,00		30,82	164,82
						w tym:	134,00	23%	30,82	164,82

Szczegółowe informacje nt. ograniczeń PoC.

- Dane w tabeli są rozmieszczone w kilku linijkach, ale nie na tej samej wysokości i szerokości powierzchni jak kolumny w nagłówkach ex.:

L.p	Nazwa towaru	Kod CN GTU	Ilość szt./ kompl	JM	Cena jednost. bez podatku w złotych	Wartość towaru bez podatku w złotych	VAT stwk %
1 AC 577A	Zawór przekaźni	84812090	1	szt	186,88	186,88	23
2 WINNO BYC ->	Zawór przekaźniko	84812090	0	szt	186,88	0,00	23

- Dane w tabeli są rozmieszczone w różnych miejscach pod względem szerokości, nie jedna pod drugą, ex.:

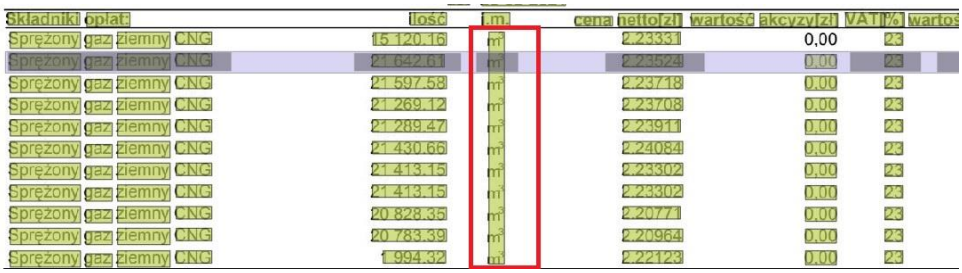
- Wartości danych z tabeli, które znajdują się w różnych polach szerokości co typ wartości (ex. Stawka VAT) w tabeli, ex.:

Poz. Kod produktu / Nazwa produktu lub usługi	Ilość	JM	Kod CN	Wartość netto	Stawka VAT[%] / Kwota VAT	Wartość brutto
40 S-02230-3 ON VERVA - OSOBOWE	282,00	L		1.242,93	23% 285,88	1.528,81
100 S-05230-3 BENZYNA SUPER PLUS 98 - OSOBOWE	252,40	L		1.158,34	23% 266,42	1.424,76
180 S-12230-0 PŁYNY EKSPLOATACYJNE	18	SZT		433,86	23% 99,79	533,65
220 S-20230-0 USŁUGI SAMOCHODOWE	28	SZT		230,76	23% 53,07	283,83
280 S-29230-0 KOSMETYKI SAMOCHODOWE	4	SZT		52,07	23% 11,98	64,05
330 S-36230-0 OPŁATA AUTOSTRADOWA	5	SZT		93,50	23% 21,50	115,00

- Jeśli nastąpi rzadki błąd (randomowy) brak odczytania danych pola nie są zidentyfikowane przez aplikacje (patrz brak podświetlenia wartości na żółto w polu obiektowym), powtarzalność błędu 1%, ex.:

				rabatem	wa netto					
1	Naprawa autobusu Solaris U18 o nr rej. WSC015AL (9915), PZU PL2021021100486	142,800	rbg	115,00	115,00	23	16 422,00	3 777,06	20 199,06	0,00
2	Lakierowanie	21,800	rbg	115,00	115,00	23	2 507,00	576,61	3 083,61	0,00
3	Materiały lakiernicze	1,000	szt.	1 002,80	1 002,80	23	1 002,80	230,64	1 233,44	0,00
4	Normalia	1,000	szt.	584,30	584,30	23	584,30	134,39	718,69	0,00
5	0004-490-214 zderzak p/p	1,000	szt.	1 732,50	1 732,50	23	1 732,50	398,48	2 130,98	0,00
6	0004-487-274 nakładka zderzaka p/p	1,000	szt.	356,13	356,13	23	356,13	81,91	438,04	0,00
7	0004-488-458 maska przednia	1,000	szt.	2 298,45	2 298,45	23	2 298,45	528,64	2 827,09	0,00
8	0004-486-591 rama maski przedniej	1,000	szt.	1 481,76	1 481,76	23	1 481,76	340,80	1 822,56	0,00
9	0004-533-567 nakładka słupka p/p	1,000	szt.	849,60	849,60	23	849,60	195,41	1 045,01	0,00
10	zestaw montażowy	1,000	szt.	150,00	150,00	23	150,00	34,50	184,50	0,00
11	0004-097-421 panel podszybia	1,000	szt.	1 449,48	1 449,48	23	1 449,48	333,38	1 782,86	0,00

- Jeśli wartość w dokumencie zostanie oznaczona na żółto tzn. więcej niż jedna wartość z ilości kolumn w orientacji pionowej czy poziomej wtedy aplikacja nie zwróci żadnych danych powtarzalność błędu 1-2%, ex.:



Składniki opłat	ilość	[m]	cena netto[zł]	wartość	akcyza[zł]	VAT[%]	wartos
Sprężony gaz ziemny CNG	15 120,16	m³	2,23331		0,00	23	
Sprężony gaz ziemny CNG	21 642,61	m³	2,23524		0,00	23	
Sprężony gaz ziemny CNG	21 597,58	m³	2,23718		0,00	23	
Sprężony gaz ziemny CNG	21 269,12	m³	2,23708		0,00	23	
Sprężony gaz ziemny CNG	21 289,47	m³	2,23911		0,00	23	
Sprężony gaz ziemny CNG	21 430,66	m³	2,24084		0,00	23	
Sprężony gaz ziemny CNG	21 413,15	m³	2,23302		0,00	23	
Sprężony gaz ziemny CNG	21 413,15	m³	2,23302		0,00	23	
Sprężony gaz ziemny CNG	20 828,35	m³	2,20771		0,00	23	
Sprężony gaz ziemny CNG	20 783,39	m³	2,20964		0,00	23	
Sprężony gaz ziemny CNG	1 994,32	m³	2,22123		0,00	23	

Walidacja treści otrzymanych danych z ekstrakcji po wykonanym procesie dla określonego szablonu.

- Wszystkie otrzymane wartości z wybranych przez użytkownika pól zostaną zwalidowane według założonych warunków wymagań.
- Występują 3 typy walidatorów w aplikacji:
  - Non Critical - niekrytyczny,
  - Critical - krytyczny,
  - Cleaner - czyszczenie zbędnych słów/wyrazów.

Pola obiektowe tekstowe - kod pocztowy – proces walidacji wskaże jego poprawność w formacie xx-xxx.

Proces przebiegu walidacji przy niepoprawnym formacie:

- Jeśli będzie składał się z dwóch oddzielnych pól lub więcej, połączy w jedno pole.
- Jeśli będzie niepoprawny format wskaże błąd, że taki kod nie istnieje o treści komunikatu: Wartość pola nie spełnia formatu kodu pocztowego(xx-xx).
- Jeśli będą występować dodatkowe niechciane słowa/wyrazy występujące przy wymaganych treściach walidator usunie je poprzez typ walidatora Cleaner.

Pola obiektowe dat - data wystawienia, data sprzedaży, data usługi, data płatności, etc. Proces przebiegu walidacji wskaże jego poprawność według formatu dat przy założonych parametrach na wejściu.

Proces przebiegu walidacji przy niepoprawnym formacie:

- Jeśli będzie składał się z dwóch oddzielnych pól lub więcej, połączy w jedno pole.
- Jeśli będzie niepoprawny format wskaże błąd, że taki format daty nie istnieje o treści komunikatu: Wartość pola nie jest datą.
- Jeśli będą występować dodatkowe niechciane słowa/wyrazy występujące przy wymaganych treściach walidator usunie je poprzez typ walidatora Cleaner.

Pola obiektowe kwotowe (księgowe) - suma netto, suma brutto, suma vat etc. Proces przebiegu walidacji wskaże jego poprawność według formatu księgowego przy założonych parametrach na wejściu.

Proces przebiegu walidacji przy niepoprawnym formacie:

- Jeśli będzie składał się z dwóch oddzielnych pól lub więcej, połączy w jedno pole.
- Jeśli będzie niepoprawny format wyliczeń wskaże błąd i wskaże w którym miejscu nastąpił błąd w określonym wyliczeniu przy podsumowaniu. Algorytm przelicza następujące sumy i wskazuje błąd, gdy:
    - Wartość kwoty brutto nie jest równa sumie netto + vat,
    - Wartość kwoty netto nie jest równa różnicy brutto i vat,
    - Wartość kwoty vat nie równa się różnicy brutto i netto.
- Jeśli będą występować dodatkowe niechciane słowa/wyrazy występujące przy wymaganych treściach walidator usunie je poprzez typ walidatora Cleaner.

Pola obiektowe NIP (Numer identyfikacji podatkowej) - ciąg 10 cyfr – proces walidacji wskaże jego poprawność w wymaganym formacie.

Proces przebiegu walidacji przy niepoprawnym formacie:

- Jeśli będzie składał się z dwóch oddzielnych pól lub więcej, połączy w jedno pole.

- Jeśli będzie niepoprawny format wskaże błąd, że taki numer identyfikacji podatkowej nie istnieje o treści komunikatu: Wartość pola nie jest poprawnym numerem NIP. Popraw ręcznie w aplikacji.
- Jeśli będą występować dodatkowe niechciane słowa/wyrazy występujące przy wymaganych treściach walidator usunie je poprzez typ walidatora Cleaner.

Pola obiektowe numeru rachunku bankowego - ciąg 26 cyfr – proces walidacji wskaże jego poprawność w wymaganym formacie.

Proces przebiegu walidacji przy niepoprawnym formacie:

- Jeśli będzie składał się z dwóch oddzielnych pól lub więcej, połączy w jedno pole.
- Jeśli będzie niepoprawny format wskaże błąd, że taki numer rachunku bankowego nie istnieje o treści komunikatu: Wartość pola nie jest poprawnym numerem konta bankowego. Popraw ręcznie w aplikacji.
- Jeśli będą występować dodatkowe niechciane słowa/wyrazy występujące przy wymaganych treściach walidator usunie je poprzez typ walidatora Cleaner.

Pola nadrzędne niechciane w uzyskiwaniu danych - walidator usuwa niezbędne pola słów/wyrazów za pomocą typu walidatora Cleaner, które również mogą występować przy otrzymywaniu danych, takie jak:

Nabywca, Sprzedawca, Sprzedawca:, Nabywca:, Konto, Adres, Nr, Nr., Nr:.

Zakończenie procesu ekstrakcji danych i walidacji

Plik po zakończeniu procesu ekstrakcji danych i procesu walidacji zmienia status dokumentu oraz miejsce zakładki w aplikacji:

- Jeśli nie wystąpiły żadne błędy i każde z wymaganych pól zostały uzupełnione w określonym zdefiniowanym szablonie trafia do zakładki o nazwie - Pliki poprawnie przetworzone.
- Jeśli wystąpił przynajmniej jeden błąd lub więcej na etapie tworzenia szablonu lub walidacji ex. jedno z pól nie zostało uzupełnione w określonym zdefiniowanym szablonie lub na etapie walidacji ex. sumy kwotowe są niepoprawne (patrz pkt. 8.) plik trafia do zakładki o nazwie - Pliki z błędami.

Po zakończeniu procesu ekstrakcji danych i walidacji prawidłowo otrzymanych danych przy zdefiniowaniu określonych pól obiektowych, wszystkie otrzymane wartości z poszczególnych plików z zakładki Pliki poprawnie przetworzone zostają zapisane w formatach plików: json, xml, csv., gdzie następuje eksport danych do innych zewnętrznych narzędzi odbiorcy danych.

Podsumuwując PoC pozwala na:

- zalogowanie/wylogowanie się do/z menu aplikacji,
- przesłanie dokumentu w odpowiednim wymaganym formacie,
- przesłanie kilku dokumentów w odpowiednim formacie,
- utworzenie/dopasowanie szablonu,
- korzystania z tego samego szablonu dla kolejnych dokumentów tego samego kontrahenta,
- - korygowanie i przypisanie odpowiednich pól i ramek za pomocą „Template creator”:
    - Plik jednostronicowy bez czytania tabel,
    - Plik jednostronicowy z czytaniem tabel pozycji,
    - Plik wielostronicowy bez czytania tabel,
    - Plik wielostronicowy z czytaniem tabel (tabela na 1 stronie),
    - Plik wielostronicowy z czytaniem tabel - tabela przecina strony,
    - Plik wielostronicowy z czytaniem tabel (tabela nie na pierwszej stronie),
    - Plik wielostronicowy z polami pływającymi.

Szczegółowe zestawienie informacji z testów badawczych PoC

Testy zostały przeprowadzone przy użyciu gotowych plików (79 wersji, z różnymi ustawieniami PoC). Poza obszarem testów znalazła się weryfikacja poprawności względem błędów znajdujących się w aplikacji PoC (zadaniem testera było sprawdzenie już wygenerowanych danych z plików – główny moduł funkcjonalności aplikacji).

Zakres: testy akceptacyjne, funkcjonalności, eksploracyjne, testy w oparciu o user stories, regresywne na podstawie defektów zgłoszonych w poprzednich wersjach PoC oraz w oparciu o dodawaniu nowych funkcjonalności na podstawie dostarczonej specyfikacji.

Testy dymne PoC – po każdej zmianie dokonano walidacji poprawności głównych funkcjonalności aplikacji (moduł Zarządzanie dokumentami).

Testy akceptacyjne PoC – wyniki ścieżek powodzeń oraz negatywnych ścieżek niepowodzeń według historyjek użytkownika (moduł Zarządzanie dokumentami).

Retesty błędów PoC wykrytych podczas testów. Skupiono się przede wszystkim na znalezieniu i zaraportowaniu defektów, a nie sugestii (moduł Zarządzanie dokumentami).

Zakres testów obejmował moduły:

- typ i rodzaj/ format pliku,

- liczby stron pliku,

- rozpoznanie wartości pól,

- rozpoznanie wartości pól po rodzajach: statyczne, dynamiczne, tabelaryczne,

- budowanie szablonu,

- tworzenie szablonu,

- walidacji treści otrzymanych danych,

- zakończenie procesu ekstrakcji danych i procesu walidacji.

Postęp prac odbywał się zgodnie z planem, wszystkie zgłaszane błędy były naprawiane na bieżąco oraz zwiększało to jakość dostarczanych danych na wejściu dla testera przy kolejnych testach. Odnotowano brak czynników blokujących. Praca była płynna z zespołem deweloperskim. Podczas testów nie zostały wykryte błędy blokujące lub krytyczne. W związku z tym, jakość aplikacji testowanych funkcjonalności PoC w warunkach laboratoryjnych oceniono pozytywnie. W szczególności realizacja powyższych kamieni milowych pozwoliła osiągnąć cele badawcze do dalszego zastosowania w pracach badawczych i rozwojowych, w szczególności w komplementarnym etapie prac rozwojowych (ER2).

EB3: Opracowanie metody odtwarzania procesu na podstawie zarejestrowanych działań użytkownika

W ramach EB3 opracowano metodę rejestrowania zachowania użytkowników podczas pracy z systemami a następnie techniczny PoC (ang. Proof of Concept – dowód słuszności koncepcji) rozwiązania. Opracowane rozwiązanie przewiduje możliwość współdziałania z różnymi algorytmami sterującymi automatyzacją z wykorzystaniem języka robotycznego.

Jednym z podstawowych wyzwań tego etapu było rozpoznanie różnych metod zapisywania działań użytkownika komputera osobistego – od rejestrowania komunikacji elementów oprogramowania systemowego związanych z ruchami myszą i wciskanych klawiszy na klawiaturze, tempa działania i wyłapywania charakterystycznych miejsc „decyzyjnych" i skojarzenie tych danych z danymi sfederowanymi w postaci statycznych obrazów fragmentów ekranu pod kątem uzyskania zalgorytmizowanego opis procesu wykonywanego przez człowieka wykorzystującego format języka robotycznego, którego założenia wypracowano w poprzednich krokach procesu badawczego.

W oparciu o powyższe wyzwania technologiczne opracowano szereg elementów zgodnych z założeniami badawczymi. W szczególności dokonano oceny potencjału badanych sposobów rejestrowania zachowania człowieka pod kątem wykorzystania w automatyzacji procesów biznesowych przeglądu metody rejestrowania zachowania użytkowników pod kątem użyteczności do dalszych prac nad tworzeniem automatycznego opisu procesu w języku programowania robotów, a także metody wykrywania węzłów decyzyjnych i oceny poprawności procesu. W ramach przeprowadzonych badań udowodniono na bazie PoC (Proof of Concept) na przykładzie typowych procesów biznesowych, że automatyczny opis procesu jest wykonywalny.

Zaplanowane prace badawcze w ramach niniejszego etapu badawczego zostały zakończone sukcesem – szczegółowe omówienie przedstawiono poniżej.

Process Mining

Jako podstawowe uniwersalne podejście do generowania ścieżki użytkownika zdecydowano zastosować wykrywanie procesów (process mining) na podstawie zarejestrowanych danych użytkownika – logów w postaci RFL. W tym podejściu proces rejestratora zapisuje w formacie RFL-json, będącym implementacją uogólnionego modelu RFL z wcześniejszych etapów prac badawczych, dane, które wykorzystywane do modelowania procesu użytkownika.

RFL-json

Punktem wyjścia do modelowania procesów w ramach proces miningu jest RFL-json, czyli zmodyfikowany język RFL, który pozwala zebrać informacje niezbędne do dokumentowania przebiegu procesu w kontekście środowiska uruchomieniowego danego procesu przy jednoczesnym obniżeniu nakładów programistycznych niezbędnych do przeanalizowania procesu i jego kontekstu. W szczególności RFL-json jest to format zapisu logów aktywności użytkownika. W celu zweryfikowania działania systemu został zbudowany i przetestowany PoC biblioteki recordera, który posłużył następnie w ramach prac rozwojowych do zbudowania prototytpu podsystemu DPC Recordera. Celem tego formatu jest zebranie jak największej ilości danych, które są niezbędne dla osoby implementującej robota, w szczególności osoby, która nie jest zaawansowanym programistą. Głównymi elementami tego formatu są zdarzenia typu:

- Open browser,
- Click,
- Type text,
- Region of interest.

Na potrzeby niniejszego opracowania poniżej scharakteryzowana wspomniane główne zdarzenia.

1. Open browser

   Zdarzenie opisujące otwarcie przeglądarki bądź karty. Zawiera startowy URL oraz typ i wersję przeglądarki.

2. Click

   Charakteryzuje się zbiorem selektorów kontrolki, która została wciśnięta. Zapisany zostaje url, gdy dochodzi do wydarzenia, data wykonania oraz dane sfederowane typu screenshot, czyli fragment ekranu użytkownika w postaci bitmapy.

3. Type text

   Posiada te same cechy co Click, ale dodatkowo zapisuje tekst jaki został wprowadzony do kontrolki przed utratą skupienia na niej.

4. Region of interest

   Jest to typ pomocniczy. Nie zawsze dochodzi do interakcji z kluczowymi elementami procesu. Czasami wystarczy by osoba przeprowadzająca dany proces biznesowy spojrzała na określony element, który warto w tym celu zarejestrować. Takie zdarzenie jest rejestrowane jako Region of interest. Jego struktura wygląda jak Type text, tylko w miejscu tekstu wpisanego jest tekst z kontrolki.

Poniżej pokazano jak wyglądają logi RFL-json) dla przebiegu typowego procesu biznesowego użytego w ramach testów PoC. Szczegóły procesu opisano w dalszej części opracowania, w ramach opisu eksperymentalnych prac rozwojowych.

```
{
  "Documents": [
    {
      "LanguageVersion": "0.2",
      "MachineDetails": { "Resolution": "1440x1080" },
      "Steps": [
        {
          "Command": "OpenBrowser",
          "Data": {
            "BrowserType": "Chrome",
            "BrowserVersion": "87.0.4280",
            "StartUrl": "https://localhost:44390/",
            "BrowserId": "1"
          }
        },
        {
          "Command": "TypeText",
          "Data": {
            "BrowserAction": {
              "Selectors": [ "id=userName", "name=userName", "css=#userName", "xpath=//input[@id='userName']", "xpath=//input" ],
              "BrowserId": "1",
              "BrowserUrl": "https://localhost:44390/Account/Login?ReturnUrl=%2F",
              "Screenshot": "",
              "Date": "2021-01-20T08:51:19.1135447+01:00"
            },
            "InsertedText": "Admin"
          }
        },
        {
          "Command": "TypeText",
          "Data": {
            "BrowserAction": {
              "Selectors": [ "id=password", "name=password", "css=#password", "xpath=//input[@id='password']", "xpath=//div[2]/input" ],
              "BrowserId": "1",
              "BrowserUrl": "https://localhost:44390/Account/Login?ReturnUrl=%2F",
              "Screenshot": "",
              "Date": "2021-01-20T08:51:19.1151977+01:00"
            },
            "InsertedText": "adminRPA!"
          }
        },
        {
          "Command": "Click",
          "Data": {
            "ClickCommand": "LeftClick",
            "BrowserAction": {
              "Selectors": [ "css=.btn", "xpath=//button[@type='submit']", "xpath=//form/button", "xpath=//button[contains(.,'Zaloguj się')]" ],
              "BrowserId": "1",
              "BrowserUrl": "https://localhost:44390/Account/Login?ReturnUrl=%2F",
              "Screenshot": "",
              "Date": "2021-01-20T08:51:19.1152056+01:00"
            }
          }
        },
        {
          "Command": "Click",
          "Data": {
            "ClickCommand": "LeftClick",
            "BrowserAction": {
              "Selectors": [ "linkText=Zacznij raportowanie", "css=.btn-info", "xpath=//a[contains(text(),'Zacznij raportowanie')]", "xpath=//a[c
              "BrowserId": "1",
              "BrowserUrl": "https://localhost:44390/",
              "Screenshot": "",
              "Date": "2021-01-20T08:51:19.1156983+01:00"
```


RFL-json dla wybranego przebiegu we wzorcowym procesie biznesowym

```
    }
   }
  },
  {
   "Command": "TypeText",
   "Data": {
    "BrowserAction": {
     "Selectors": [ "id=ConsignmentNumber", "name=ConsignmentNumber", "css=#ConsignmentNumber", "xpath=//input[@id='ConsignmentNumber']", "xpath=//div/input" ],
     "BrowserId": "1",
     "BrowserUrl": "https://localhost:44390/Consignment/Add",
     "Screenshot": "",
     "Date": "2021-01-20T08:51:19.1157026+01:00"
    },
    "InsertedText": "1"
   }
  },
  {
   "Command": "TypeText",
   "Data": {
    "BrowserAction": {
     "Selectors": [ "id=ReasonCode", "name=ReasonCode", "css=#ReasonCode", "xpath=//input[@id='ReasonCode']", "xpath=//div[2]/input" ],
     "BrowserId": "1",
     "BrowserUrl": "https://localhost:44390/Consignment/Add",
     "Screenshot": "",
     "Date": "2021-01-20T08:51:19.1157038+01:00"
    },
    "InsertedText": ""
   }
  },
  {
   "Command": "TypeText",
   "Data": {
    "BrowserAction": {
     "Selectors": [ "id=Description", "name=Description", "css=#Description", "xpath=//input[@id='Description']", "xpath=//div[3]/input" ],
     "BrowserId": "1",
     "BrowserUrl": "https://localhost:44390/Consignment/Add",
     "Screenshot": "",
     "Date": "2021-01-20T08:51:19.115705+01:00"
    },
    "InsertedText": ""
   }
  },
  {
   "Command": "TypeText",
   "Data": {
    "BrowserAction": {
     "Selectors": [ "id=ExpectedDeliveryDate", "name=ExpectedDeliveryDate", "css=#ExpectedDeliveryDate", "xpath=//input[@id='ExpectedDeliveryDate'", "xpath=//div[4]/input" ],
     "BrowserId": "1",
     "BrowserUrl": "https://localhost:44390/Consignment/Add",
     "Screenshot": "",
     "Date": "2021-01-20T08:51:19.1157068+01:00"
    },
    "InsertedText": "02122020"
   }
  },
  {
   "Command": "TypeText",
   "Data": {
    "BrowserAction": {
     "Selectors": [ "id=DeliveryDate", "name=DeliveryDate", "css=#DeliveryDate", "xpath=//input[@id='DeliveryDate']", "xpath=//div[5]/input" ],
     "BrowserId": "1",
     "BrowserUrl": "https://localhost:44390/Consignment/Add",
     "Screenshot": "".

      "Screenshot": "",
      "Date": "2021-01-20T08:51:19.1157081+01:00"
     },
     "InsertedText": "02122020"
    }
   },
   {
    "Command": "TypeText",
    "Data": {
     "BrowserAction": {
      "Selectors": [ "id=DeliveryAddress", "name=DeliveryAddress", "css=#DeliveryAddress", "xpath=//input[@id='DeliveryAddress']", "xpath=//div[6]/input" ],
      "BrowserId": "1",
      "BrowserUrl": "https://localhost:44390/Consignment/Add",
      "Screenshot": "",
      "Date": "2021-01-20T08:51:19.115709+01:00"
     },
     "InsertedText": "Wołoska 5"
    }
   },
   {
    "Command": "Click",
    "Data": {
     "ClickCommand": "LeftClick",
     "BrowserAction": {
      "Selectors": [ "css=.btn-primary", "xpath=//button[@type='submit']", "xpath=//form/button", "xpath=//button[contains(.,'Dodaj')]" ],
      "BrowserId": "1",
      "BrowserUrl": "https://localhost:44390/Consignment/Add",
      "Screenshot": "",
      "Date": "2021-01-20T08:51:19.1157098+01:00"
     }
    }
   }
  ]
 },
```


RFL-json dla wybranego przebiegu we wzorcowym procesie biznesowym cd.

Opracowanie modelu proces miningu oraz zbudowany na tej podstawie PoC, który posłużył do dalszych prac nad prototypem docelowego recordera w komplementarnym etapie ER3 pozwolił odpowiedzieć na szereg kluczowych kwestii dotyczących:

• generowania modelu na podstawie logów,

• odwzorowywania procesu w postaci diagramu (przykład w komplementarnej części dotyczącej eksperymentalnych prac rozwojowych ER3),

• identyfikacji wąskich gardeł procesu,

• weryfikowania poprawności wykonywania procesu biznesowego przez operatora,

• identyfikacji głównych ścieżek procesu,

• zoptymalizowacji procesu.

Każde zdarzenie opisane w RFL-json w process miningu jest węzłem (rekordem) typu event data. Każdy rekord tego typu musi zawierać trzy obowiązkowe pola:

- Id wydarzenia,
- Nazwa wydarzenia,
- Data.

RFL-json przewiduje miejsce na pola opcjonalne. Zbiór zdarzeń zostanie jest umieszczany w pliku o nazwie event log. Jest on zapisywany w formacie XES.

Format XES

XES Standard (Extensible Event Stream) powstał po to by wymieniać się informacjami o zdarzeniach pomiędzy systemami. Bazuje na formacie XML. Poniżej zaprezentowano, jak wygląda jedno z zarejestrowanych zdarzeń w formacie XES dla procesu wzorcowego w ramach PoC.

```
<event>
  <string key="concept:name" value="Type:TypeText #!^5 Url:https://localhost:44390/Account/Login?ReturnUrl=%2F
  #!^5 MainSelector:xpath=//input[@id='password'] #!^5 " />
  <date key="time:timestamp" value="2021-07-19T12:12:20.168Z" />
</event>
```

Pojedynczy log zdarzenia w formacie XES.

Element ten charakteryzuje wartość dla klucza concept:name. W tym miejscu jest to umieszczone ID zdarzenia składające się z typu zdarzenia, url-a i głównego selektora. Dodatkowo niezbędna jest data by móc później uporządkować te elementy.

Directly-Follows Graph

Uzyskawszy logi w formacie XES, można rozpocząć zasadniczą analizę procesu. Na potrzeby PoC wykorzystano do tego celu bibliotekę pm4py, która potrafi przetworzyć zapis formatu xes do postaci grafu w fromacie svg. Graf ten jest typu Direct Follower Graph i służy do reprezentacji procesu w ramach process miningu. Format ten jest zbliżony do dobrze znanych formatów BPMN. Reprezentację graficzną poniższego zapisu z PoC umieszczono w rozdziale opisującym komplementarny etap prac rozwojowych ER3.

```
<!-- &#45;6848614562303261895 -->
<g id="node10" class="node">
<title>&#45;6848614562303261895</title>
<polygon fill="#cdcdff" stroke="black" points="1111.5,-819 513.5,-819 513.5,-783 1111.5,-783 1111.5,-819"/>
<text text-anchor="middle" x="812.5" y="-797.3" font-family="Times New Roman,serif" font-size="14.00">
Type:TypeText #!^5 Url:https://localhost:44345/Consignment #!^5 MainSelector:xpath:position #!^5  (2)</text>
</g>
```

Pojedynczy log zdarzenia w formacie svg.

Podsumowując w ramach niniejszego etapu badawczego (EB3) opracowano metodę rejestrowania zachowania użytkowników podczas pracy z systemami z użyciem RFL-json, a następnie techniczny PoC (ang. Proof of concept – dowód słuszności koncepcji) rozwiązania. Opracowane rozwiązanie przewiduje możliwość współdziałania z różnymi algorytmami sterującymi automatyzacją z wykorzystaniem języka robotycznego. W związku z tym, jakość aplikacji testowanych funkcjonalności PoC w warunkach laboratoryjnych oceniono pozytywnie. W szczególności realizacja powyższych kamieni milowych pozwoliła

osiągnąć cele badawcze do dalszego zastosowania w pracach badawczych i rozwojowych, w szczególności w komplementarnym etapie prac rozwojowych (ER3).

ER1: Budowa prototypu i środowiska robotycznego

Na podstawie przeprowadzonych badań przemysłowych w etapie 1 (EB1) w komplementarnym do niego pierwszym etapie prac rozwojowych (ER1) zaimplementowano interpreter języka robotycznego oraz oprogramowanie potrafiące rozpoznawać i wykonywać polecenia ze specyfikacji języka dla robotów. W ramach prac rozwojowych przeprowadzono również prototypowe uruchomienie w środowisku klienta firmy DPC, na wybranym procesie biznesowym. Opracowano również prototyp oprogramowania monitorującego i zarządzającego pracą robotów.

Jednym z podstawowych wyzwań tego etapu było opracowanie i implementacja interpretera języka dla RFL na podstawie specyfikacji wypracowanej w etapie badań przemysłowych wraz ze zintegrowanego z językiem środowiska uruchomieniowego pozwalającego na monitorowanie i zarządzanie pracą robotów. W oparciu o powyższe wyzwania technologiczne opracowano szereg elementów zgodnych z założeniami badawczymi. W szczególności opracowano interpreter składni i środowisko uruchomieniowe robotów, umożliwiające kontrolę poprawności i kompletności poleceń, zwłaszcza dotyczących obsługi wyjątków i automatyzacji z użyciem uczenia maszynowego oraz możliwości uczenia eksperckiego w wyspecyfikowanych wcześniej obszarach, a także zaimplementowano prototyp oprogramowania, który umożliwił zebranie wniosków z implementacji i wdrożenia prototypu do opracowania rozwiązania docelowego.

Zaplanowane prace badawcze w ramach niniejszego etapu badawczego zostały zakończone sukcesem – szczegółowe omówienie przedstawiono poniżej.

Wykorzystując specyfikację wyjściową języka wypracowaną w komplementarnym etapie badawczym (EB1) w toku eksperymentalnych prac rozwojowych w etapie ER1 wytworzony został docelowy język RFL, który jest interpretowany i zapisywany w formacie XML, ściśle powiązany z ekosystemem środowiska .NET.

Język ten ma charakter uniwersalny i umożliwia zapisu dowolnego przebiegu procesu przeznaczonego do automatyzacji biznesowej. Zawiera szereg kluczowych elementów jak deklaracja zmiennych, wywoływanie funkcji środowiskowych .NET oraz kluczowych elementów niezbędnych do wykonywania procesów, takich jak:

- otwarcie przeglądarki,
- otworzenie odpowiedniej strony,
- odświeżenie przeglądarki,
- wpisanie na stronie internetowej odpowiedniej wartości tekstowej,
- aktywacja elementu na stronie internetowej.

Język jest rozszerzony o niezbędne pola, potrzebne przy robotyzacji procesu. W szczególności funkcja otwierania przeglądarki pozwala na otwarcie jej w trybie headless z domyślną maksymalizacją. Funkcje w przeglądarce posiadają opcje uodpornienia implementacji na zmiany. Można również ustawić czas oczekiwania. Jest to przydatne z uwagi na zmienność środowiska uruchomieniowego robotów, gdy pewne elementy strony ładują się dłużej i chcemy aktywnie sprawdzać czy się już one pojawiły. Zmienność w widoku strony również została uwzględniona. Przy szukaniu elementów na stronie poszukiwany jest nie jeden selektor wyznaczający element, a wszystkie podane. Dzięki temu, jeśli istnieje selektor zawierający tekst wypisywany na stronie i ulegnie on zmianie, wówczas automat będzie nadal działał, o ile inne kluczowe elementy, np. id elementu, nie uległy zmianie.

Interface programistyczny języka RFL

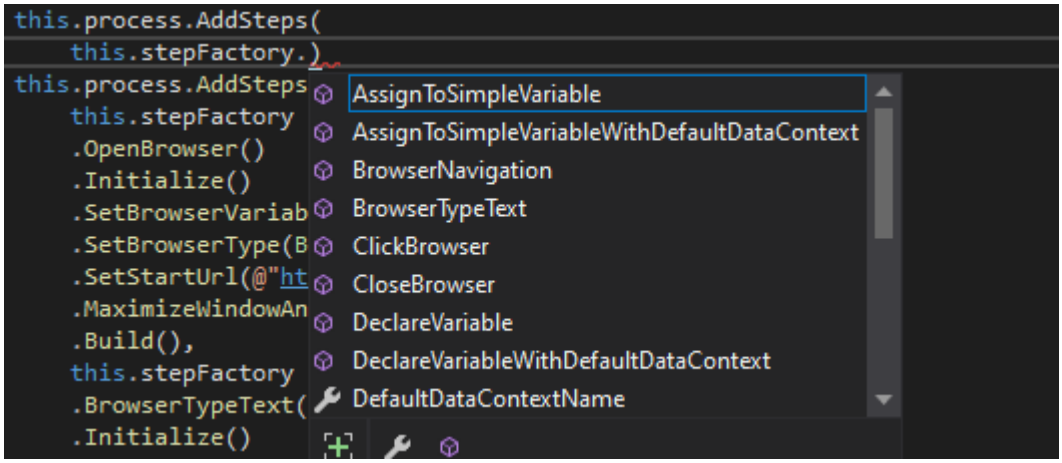
Z powodu trudu jaki może wystąpić w samodzielnym pisaniu procesów w RFL-u, powstał interfejs programistyczny do budowania takich plików. Dołożono wszelkich starań by generowanie kodu w taki sposób było płynne i bezproblemowe z punktu widzenia operatora systemu. Z tego powodu na etapie implementacji w ramach prac rozwojowych

skorzystano ze wzorca budowniczego z dodatkową funkcją zabezpieczającą przed błędnie zapisanym RFL-em niezgodnym ze specyfikacją.

```
18 references
public interface IOpenBrowser:IStep
{
    6 references
    bool MaximizeWindow { get; }
    6 references
    BrowserTypeEnum BrowserType { get; }
    7 references
    string Url { get; }
    6 references
    string TargetObjectName { get; }
    3 references
    public interface IBuilder: IBuild, IInitializer, IBrowserType, ITargetObjectName, IStartUrl, IMaximizeWindow, IParams
    {

    }
    7 references
    interface IBuild
    {
        12 references | 0/6 passing
        IOpenBrowser Build();
    }
    3 references
    public interface IInitializer
    {
        12 references | 0/6 passing
        ITargetObjectName Initialize();
    }
    4 references
    interface ITargetObjectName
    {
        12 references | 0/6 passing
        IBrowserType SetBrowserVariableName(string name);
    }
    3 references
    public interface IBrowserType
    {
        12 references | 0/6 passing
        IStartUrl SetBrowserType(BrowserTypeEnum browserType);
    }

    3 references
    interface IStartUrl
    {
        12 references | 0/6 passing
        IMaximizeWindow SetStartUrl(string url);
    }
    3 references
    interface IMaximizeWindow
    {
        3 references | 0/1 passing
        IParams MaximizeWindow(bool maximize);
        10 references | 0/5 passing
        IBuild MaximizeWindowAndBuildWithNoParams(bool maximize);
    }
    5 references
    interface IParams
    {
        2 references | 0/1 passing
        IParams HeadelessBrowser();
        2 references | 0/1 passing
        IBuild NoMoreParameters();
        2 references
        IBuild AddSeleniumParams(params string[] parameters);
    }
}
```

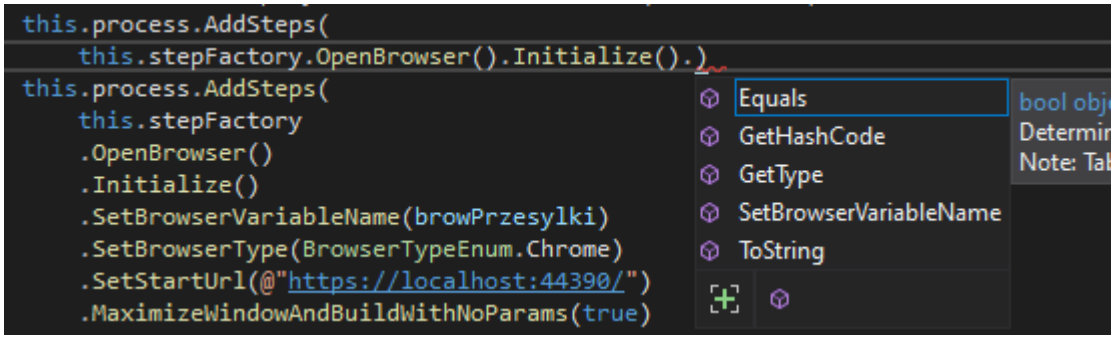
Przykładowa implementacja interfejsu do budowania funkcji otwierania przeglądarki.

Poniżej przedstawiono skuteczną formę implementacji interfejsów, która nie daje szansy na pomyłkę w utworzeniu błędnej struktury w obrębie RFL.

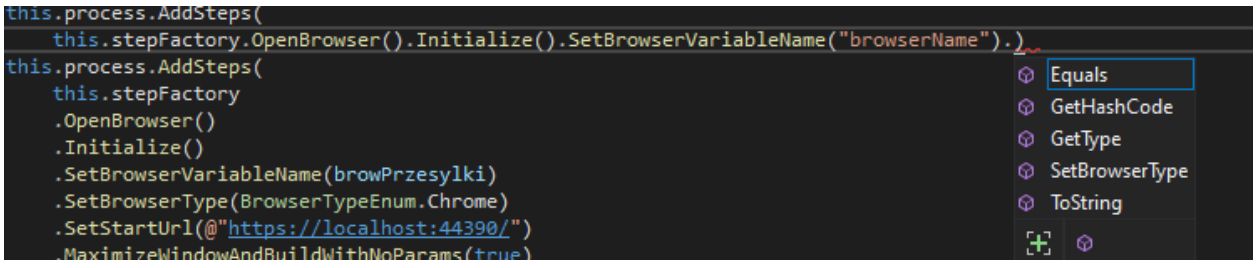


Obiekt stepFactory pokazuje na samym początku jaki element RFL-a można dodać do przebiegu.

W szczególności zaimplementowane środowisko pozwala po wybraniu OpenBrowser na tylko jedną opcję pójścia krok dalej przy budowaniu obiektu. Na początku nadajemy przeglądarce nazwę by móc się do niej później odwoływać.

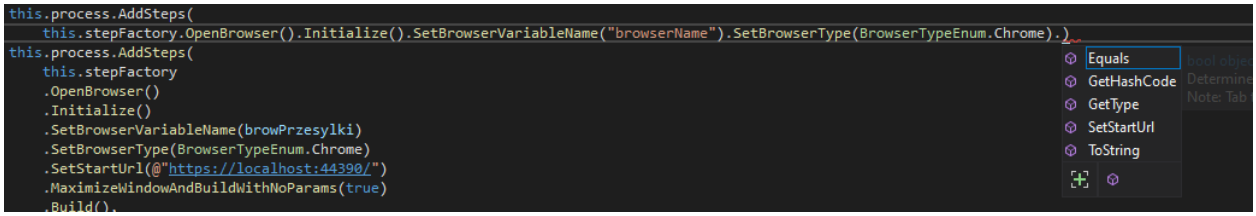


Możliwości kontynuacji dla OpenBrowser.



Możliwość kontynuacji dla OpenBrowser – deklaracja nazwy karty przeglądarki.

W kolejnych krokach następuje deklaracja typu przeglądarki zilustrowana poniżej.



Możliwość kontynuacji dla OpenBrowser – deklaracja typu przeglądarki.

Możliwość kontynuacji dla OpenBrowser – deklaracja startowego URL-a.

Zbudowany obiekt OpenBrowser.

Powyższe zrzuty ekranowe ilustrują proces zdefiniowania i zadeklarowania otwarcia konkretnego typu przeglądarki, w tym wypadku Chrome ze adresem strony startowej *google.com* z początkową maksymalizacją okna z jednoczesnym zdefiniowaniem i nadaniem identyfikatora, aby móc się później do niej odwoływać w dalszych krokach procesu. Zaprezentowane powyżej przykładowe użycie interfejsu jest możliwe do wykorzystania przy budowaniu każdego elementu języka RFL.

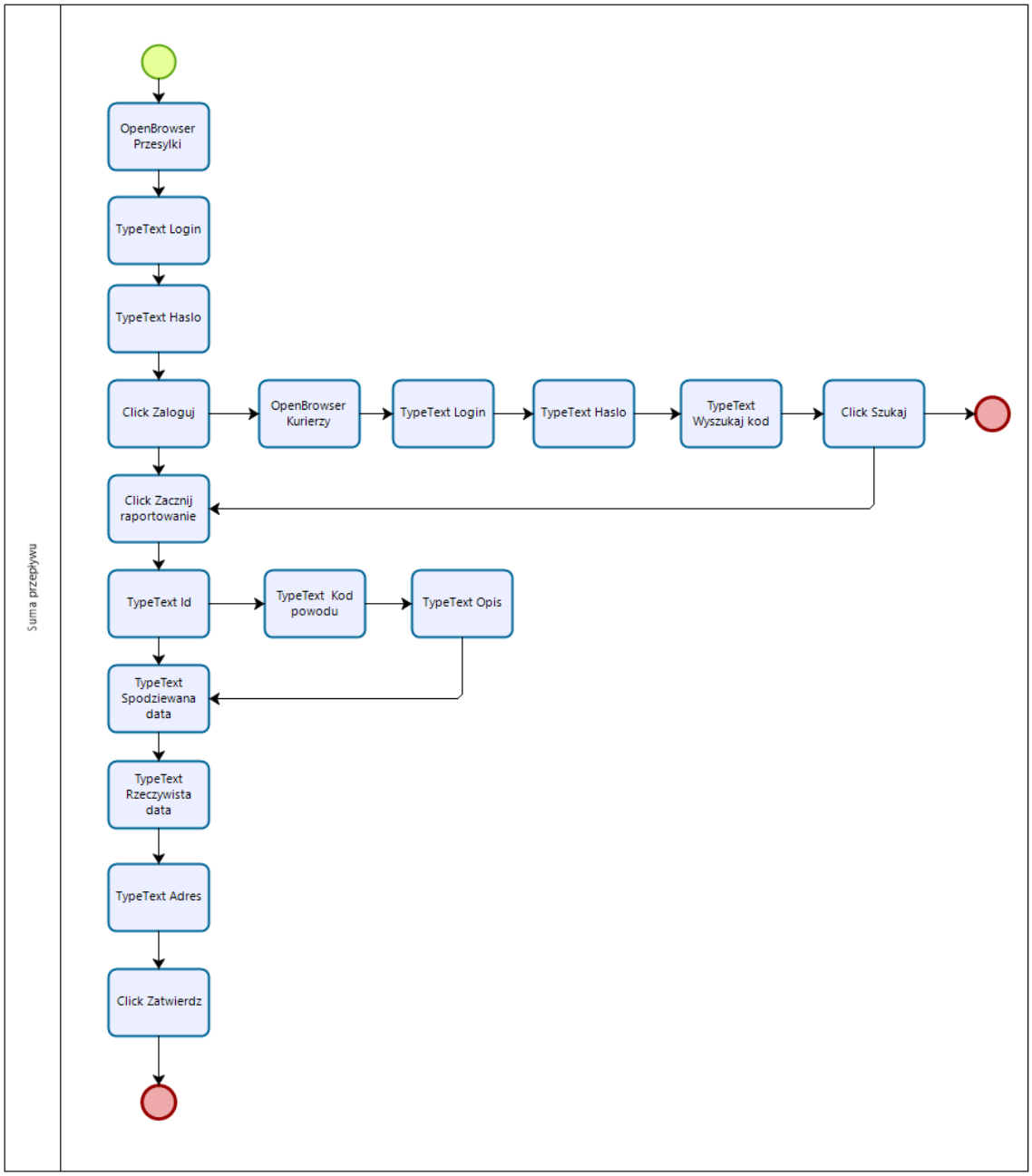
Proces wzorcowy

Na potrzeby implementacji i testów środowiska w ramach eksperymentalnych prac rozwojowych został opracowany wzorcowy proces biznesowy zawierający wystandaryzowane problemy związane z automatyzacją i robotyzacją. Proces wzrocowy charakteryzuje się w szczególności:

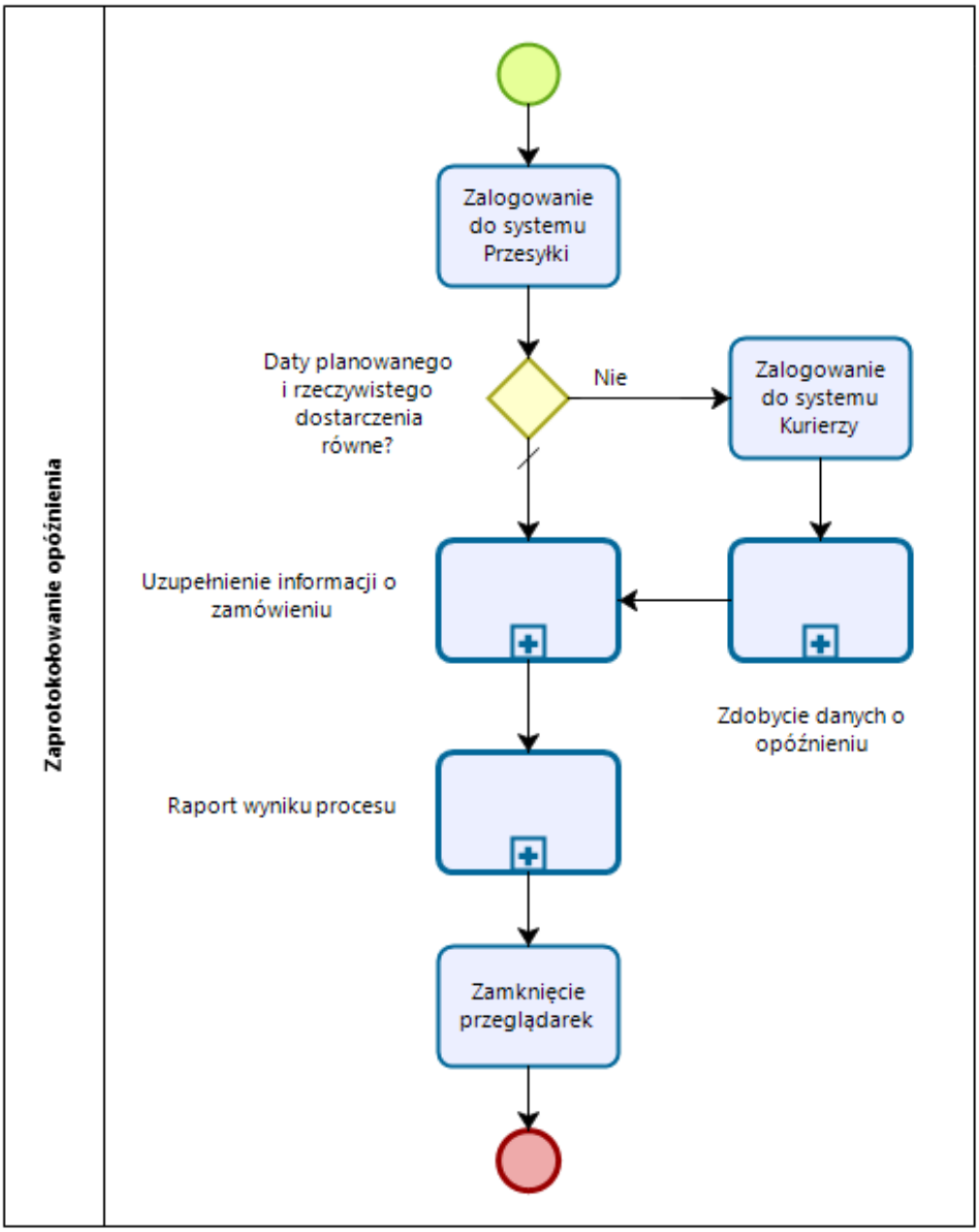
- Korzystaniem z dwóch systemów na dwóch przeglądarkach,
- Pracą z tabelą,
- Obsługą błędów walidacji,
- Pracą z kalendarzowymi kontrolkami,
- Różnymi przebiegami pozytywnymi.

Na proce wzorcowy składają się dwie wzorcowe aplikacje webowe, które zostały utworzone w toku prac rozwojowych: *Przesyłki* oraz *Kurierzy*. Wzorcowy proces polega na odtworzeniu w środowisku robotycznym opóźnienia dostarczenia przesyłki w systemie Przesyłki, wejścia do systemu Kurierzy w celu ustalenia wyciągnąć powodu opóźnienia, a następnie zaraportować zidentyfikowanego powodu w systemie Przesyłki. Poniżej przedstawiono za pomocą diagramów BPMN szczegółowy opis procesu wzorcowego.

Model wzorcowy

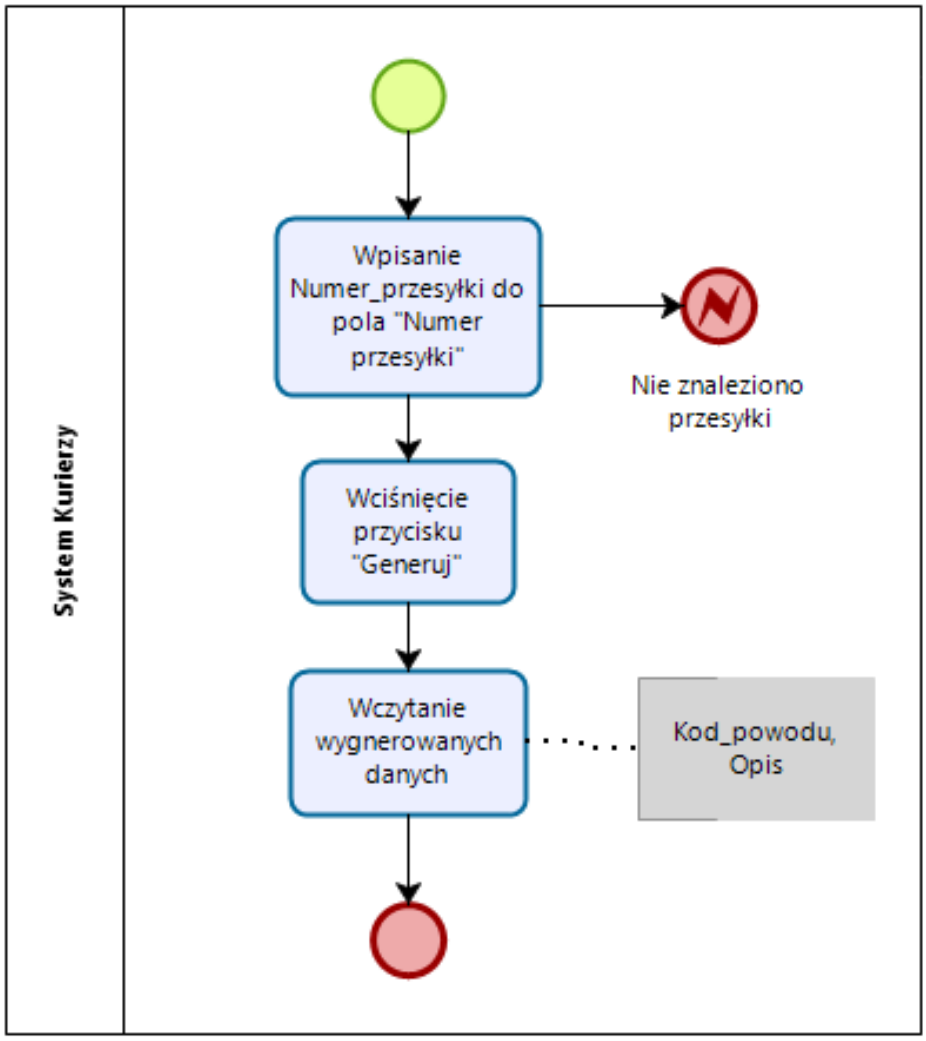


Uogólniony model Procesu Wzorcowego.

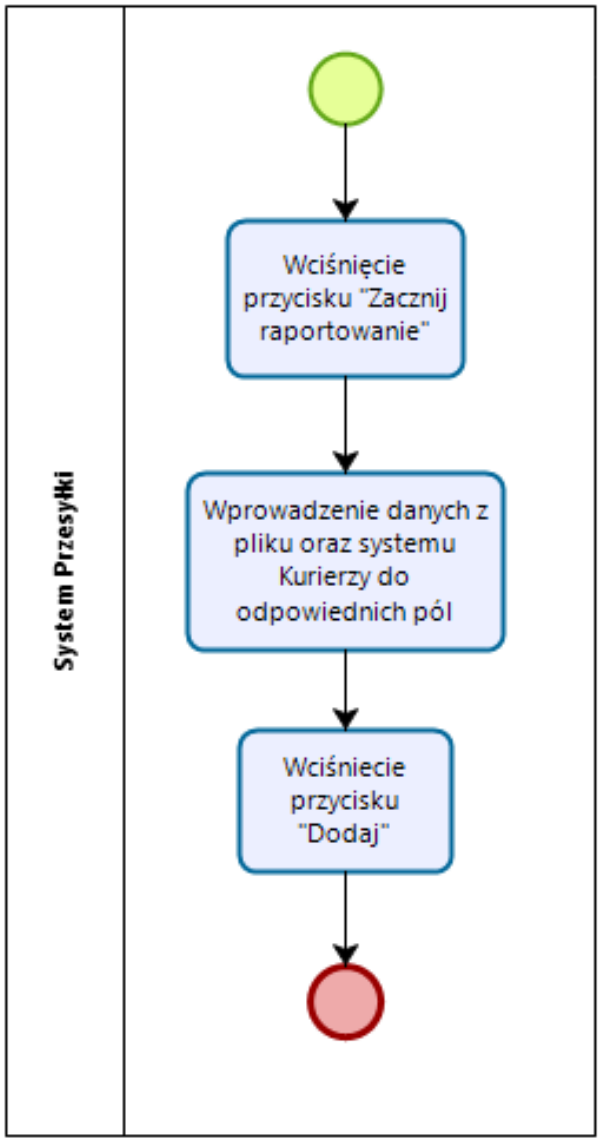


Uszczegółowiony diagram Procesu Wzorcowego.

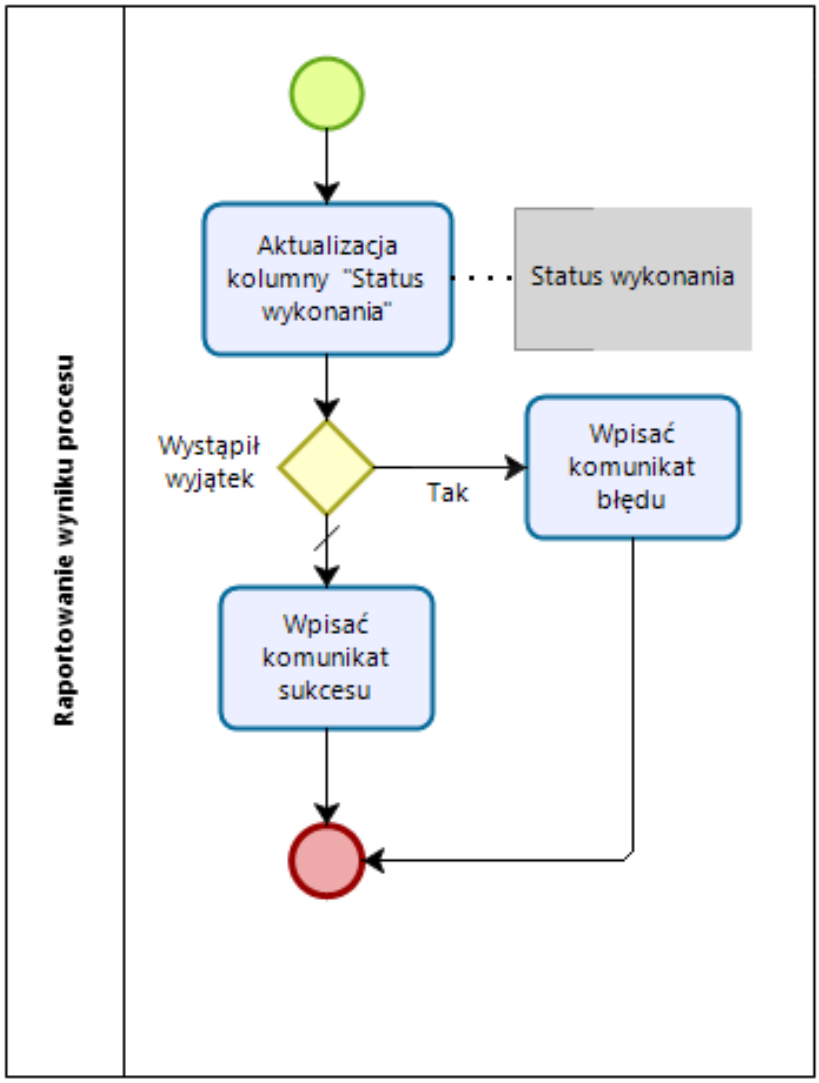
W szczególności powyższy uszczegółowiony diagram procesu wzorowego ilustruje fragment decyzyjny procesu wskazujący na potrzebę wejścia do systemu *Kurierzy*.



Praca z systemem Kurierzy – pozyskanie danych o opóźnieniu.



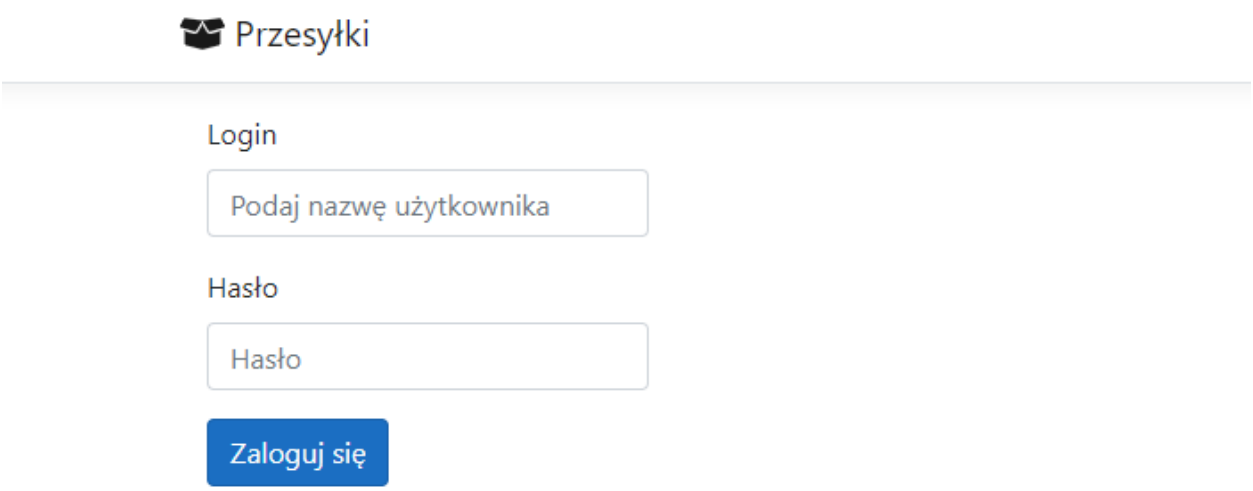
Uzupełnienie informacji o zamówieniu.



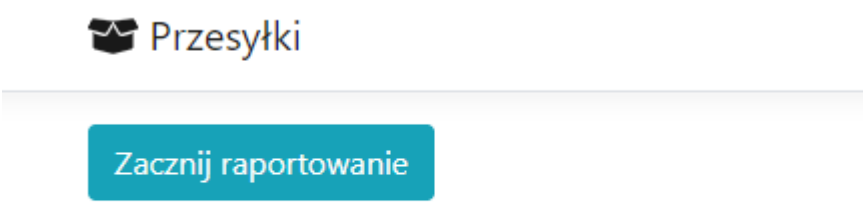
Raportowanie wyniku procesu w systemie Przesyłki.

Implementacja wzorcowych podsystemów

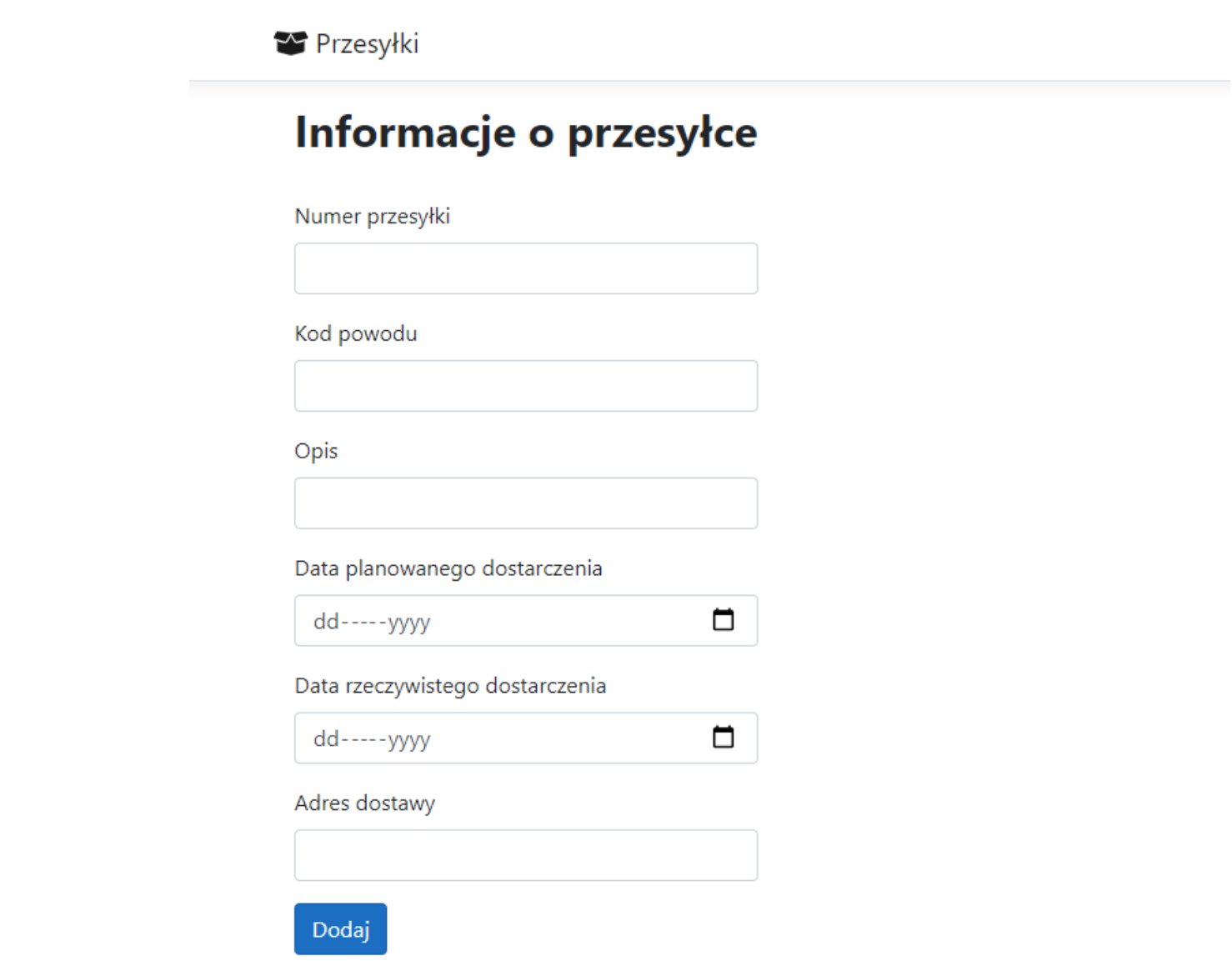
Poniżej zaprezentowano zrzuty ekranowe dokumentujące implementację wzorcowego podsystemu *Przesyłki*.



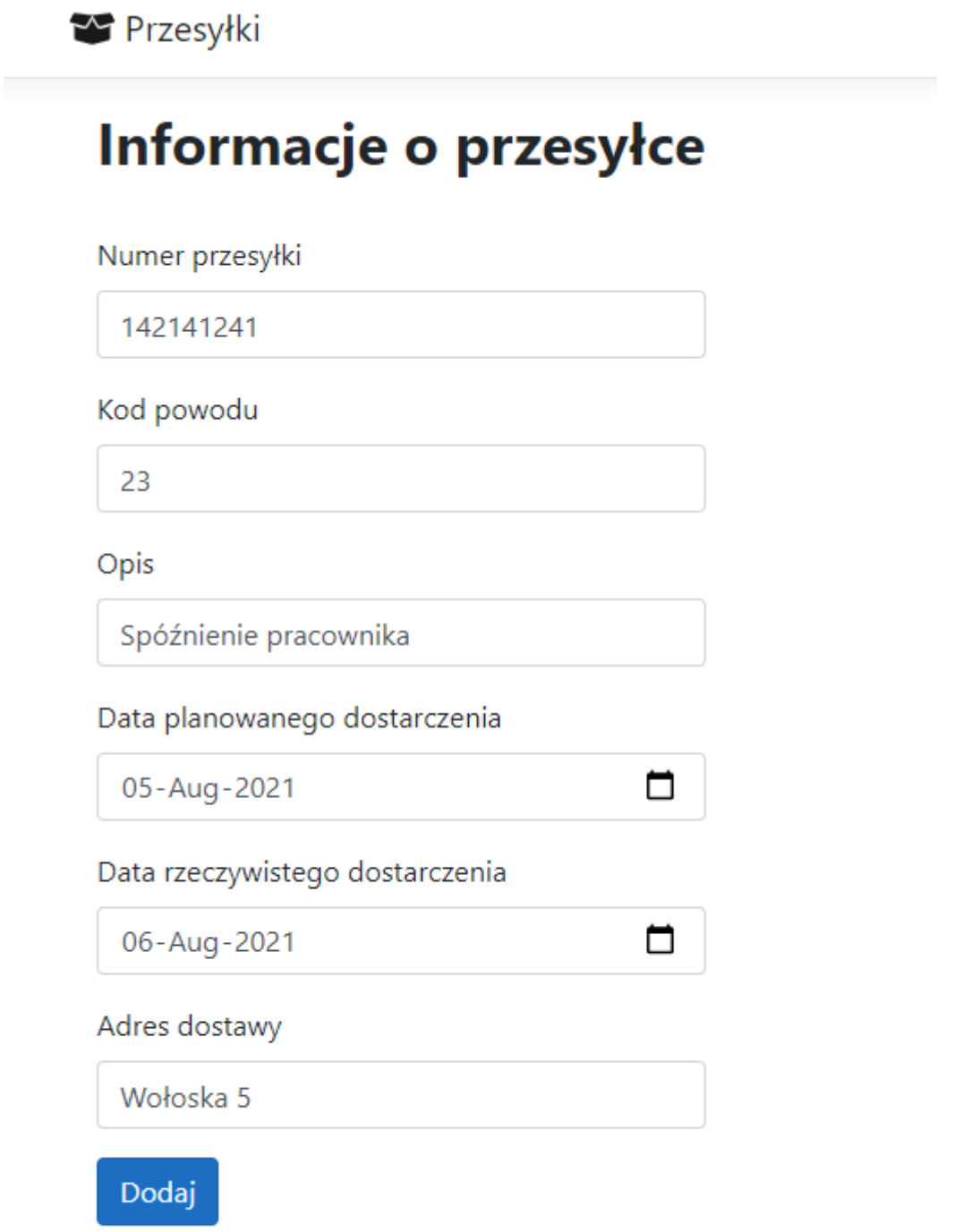
Ekran logowania do sytemu Przesyłki.



Klikając na przycisk przedstawiony na powyższej ilustracji można przejść do ekranu raportowania.

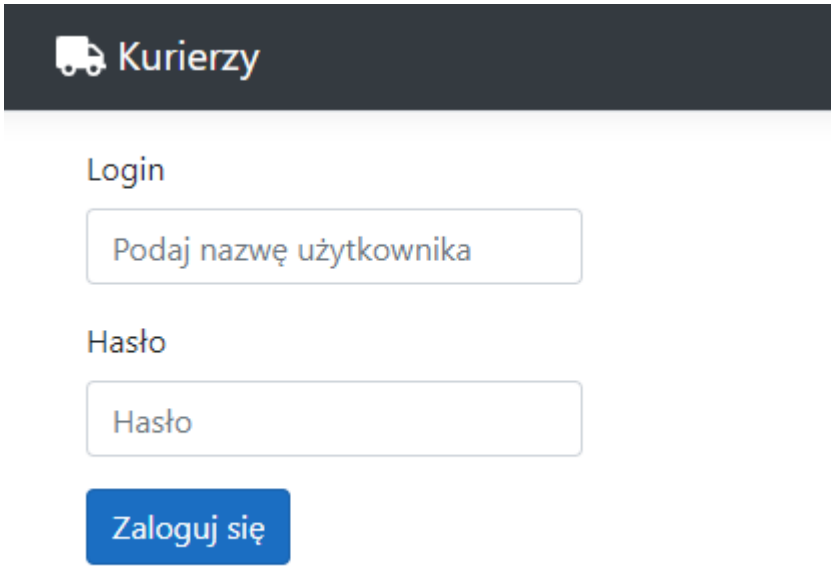


Zainicjowany pusty formularz podsystemu wzorcowego.



Formularz podsystemu wzorcowego uzupełniony o przykładowe dane.

Poniżej zaprezentowano zrzuty ekranowe dokumentujące implementację wzorcowego podsystemu *Kurierzy*.



Ekran logowania do systemu Kurierzy.

Kurierzy | Wyloguj się

Wyszukaj po numerze przesyłki: numer przesyłki | Szukaj | Wyświetl wszystkie przesyłki

Numer przesyłki	Kod powodu	Opis	Data planowanego dostarczenia	Data rzeczywistego dostarczenia	Adres dostawy
1	1	test	05-Dec-20	05-Dec-20	Warszawa ul. Idzikowskiego 16
2	2	test desc	31-Dec-20	31-Dec-20	Warszawa ul. Idzikowskiego 16
3	4	description	01-Nov-20	05-Dec-20	Warszawa ul. Idzikowskiego 16
4	5	problem w sortowni	02-Nov-20	03-Nov-20	Łódź ul. Piotrkowska 33
5	5	nie odebrano paczki	01-Nov-20	02-Nov-20	Łódź ul. Piotrkowska 1
6	10		03-Nov-20	06-Nov-20	Warszawa ul. Idzikowskiego 20
7	11	zniszczona paczka	07-Nov-20	10-Nov-20	Warszawa ul. Idzikowskiego 22
8	13	paczka zgubiona	09-Nov-20	13-Nov-20	Warszawa plac Defilad 1
9	20	dostarczona	09-Nov-20	09-Nov-20	Warszawa plac Defilad 1

Tabela łącząca kod powodu opóźnienia z numerem przesyłki.

Kurierzy | Wyloguj się

Wyszukaj po numerze przesyłki: 7 | Szukaj | Wyświetl wszystkie przesyłki

Numer przesyłki	Kod powodu	Opis	Data planowanego dostarczenia	Data rzeczywistego dostarczenia	Adres dostawy
7	11	zniszczona paczka	07-Nov-20	10-Nov-20	Warszawa ul. Idzikowskiego 22

Wyfiltrowanie tabeli o wybranym numerze przesyłki.

Implementacja przebiegu Procesu Wzorcowego przy użyciu interfejsu do tworzenia RFL

Poniżej został przedstawiony wzorcowy kod programistyczny, który może posłużyć do wygenerowania kodu w języku RFL. Wykorzystywany jest opisany powyżej przypadek testowy, w którym nie doszło do opóźnienia w przesyłce.

```
211         public void RegisterWzorcowyProces1()
212         {
213             string browPrzesylki = "Przegladarka Przesylki";
214             string login = "Admin";
215             string password = "adminRPA!";
216             string ParcelId1 = "1";
217             string plannedDays1 = "02";
218             string plannedMonths1 = "12";
219             string plannedYears1 = "2020";
220             string address1 = "Wołoska 5";
221             this.doc.ProcessSpecyfication.SaveResolution("1440x1080");
222             this.process.AddSteps(
223                 this.stepFactory
224                 .OpenBrowser()
225                 .Initialize()
226                 .SetBrowserVariableName(browPrzesylki)
227                 .SetBrowserType(BrowserTypeEnum.Chrome)
228                 .SetStartUrl(@"https://localhost:44390/")
229                 .MaximizeWindowAndBuildWithNoParams(true)
230                 .Build(),
231                 this.stepFactory
232                 .BrowserTypeText()
233                 .Initialize()
234                 .BrowserObjectName(browPrzesylki)
235                 .SetText(login)
236                 .Selector("userName")
237                 .SelectorType(SelectorByEnum.Id)
238                 .AddSelector("userName")
239                 .SelectorType(SelectorByEnum.Name)
240                 .AddSelector("userName")
241                 .SelectorType(SelectorByEnum.CssSelector)
242                 .AddSelector("//input[@id='userName']")
243                 .SelectorType(SelectorByEnum.XPath)
244                 .NoMoreSelectors()
245                 .ClearBeforeType(true)
246                 .AddPathForScreenshot(@"C:\Users\BlueCompany\Downloads")
247                 .TimePastFromLastStep(TimeSpan.FromSeconds(5))
248                 .Build(),
249                 this.stepFactory
250                 .BrowserTypeText()
251                 .Initialize()
252                 .BrowserObjectName(browPrzesylki)
253                 .SetText(password)
254                 .Selector("password")
255                 .SelectorType(SelectorByEnum.Id)
256                 .NoMoreSelectors()
257                 .ClearBeforeTypeAndNoMoreOptions(true)
258                 .Build(),
259                 this.stepFactory
260                 .ClickBrowser()
261                 .Initialize()
262                 .BrowserObjectName(browPrzesylki)
263                 .Selector("//button[@type='submit']")
264                 .SelectorType(SelectorByEnum.XPath)
265                 .NoMoreSelectors()
266                 .LeftClickAndNoMoreOptions()
267                 .Build(),
```

Implementacja wybranego przebiegu dla Procesu Wzorcowego.

```
268                 this.stepFactory
269                 .ClickBrowser()
270                 .Initialize()
271                 .BrowserObjectName(browPrzesylki)
272                 .Selector("Zacznij raportowanie")
273                 .SelectorType(SelectorByEnum.LinkText)
274                 .NoMoreSelectors()
275                 .LeftClickAndNoMoreOptions()
276                 .Build(),
277                 this.stepFactory
278                 .BrowserTypeText()
279                 .Initialize()
280                 .BrowserObjectName(browPrzesylki)
281                 .SetText(ParcelId1)
282                 .Selector("ConsignmentNumber")
283                 .SelectorType(SelectorByEnum.Id)
284                 .NoMoreSelectors()
285                 .ClearBeforeTypeAndNoMoreOptions(true)
286                 .Build(),
287                 this.stepFactory
288                 .BrowserTypeText()
289                 .Initialize()
290                 .BrowserObjectName(browPrzesylki)
291                 .SetText(plannedDays1)
292                 .Selector("ExpectedDeliveryDate")
293                 .SelectorType(SelectorByEnum.Id)
294                 .NoMoreSelectors()
295                 .ClearBeforeTypeAndNoMoreOptions(false)
296                 .Build(),
297                 this.stepFactory
298                 .BrowserTypeText()
299                 .Initialize()
300                 .BrowserObjectName(browPrzesylki)
301                 .SetText(plannedMonths1)
302                 .Selector("ExpectedDeliveryDate")
303                 .SelectorType(SelectorByEnum.Id)
304                 .NoMoreSelectors()
305                 .ClearBeforeTypeAndNoMoreOptions(false)
306                 .Build(),
307                 this.stepFactory
308                 .BrowserTypeText()
309                 .Initialize()
310                 .BrowserObjectName(browPrzesylki)
311                 .SetText(plannedYears1)
312                 .Selector("ExpectedDeliveryDate")
313                 .SelectorType(SelectorByEnum.Id)
314                 .NoMoreSelectors()
315                 .ClearBeforeTypeAndNoMoreOptions(false)
316                 .Build(),
317                 this.stepFactory
318                 .BrowserTypeText()
319                 .Initialize()
320                 .BrowserObjectName(browPrzesylki)
321                 .SetText(plannedDays1)
322                 .Selector("DeliveryDate")
323                 .SelectorType(SelectorByEnum.Id)
324                 .NoMoreSelectors()
325                 .ClearBeforeTypeAndNoMoreOptions(false)
326                 .Build(),
```

Implementacja wybranego przebiegu dla Procesu Wzorcowego cd.

```
                this.stepFactory
                .BrowserTypeText()
                .Initialize()
                .BrowserObjectName(browPrzesylki)
                .SetText(plannedMonths1)
                .Selector("DeliveryDate")
                .SelectorType(SelectorByEnum.Id)
                .NoMoreSelectors()
                .ClearBeforeTypeAndNoMoreOptions(false)
                .Build(),
                this.stepFactory
                .BrowserTypeText()
                .Initialize()
                .BrowserObjectName(browPrzesylki)
                .SetText(plannedYears1)
                .Selector("DeliveryDate")
                .SelectorType(SelectorByEnum.Id)
                .NoMoreSelectors()
                .ClearBeforeTypeAndNoMoreOptions(false)
                .Build(),
                this.stepFactory
                .BrowserTypeText()
                .Initialize()
                .BrowserObjectName(browPrzesylki)
                .SetText(address1)
                .Selector("DeliveryAddress")
                .SelectorType(SelectorByEnum.Id)
                .NoMoreSelectors()
                .ClearBeforeTypeAndNoMoreOptions(false)
                .Build(),
                this.stepFactory
                .ClickBrowser()
                .Initialize()
                .BrowserObjectName(browPrzesylki)
                .Selector("//button[@type='submit']")
                .SelectorType(SelectorByEnum.XPath)
                .NoMoreSelectors()
                .LeftClickAndNoMoreOptions()
                .Build()
                );
            string path = this.CreateRflFileName(nameof(RegisterWzorcowyProces1));
            this.SaveLoadRunProcess(this.process, path);
        }
```

Implementacja wybranego przebiegu dla Procesu Wzorcowego cd.

Prezentacja przebiegu Wzorcowego Procesu w języku RFL

Po uruchomieniu powyższego kodu zostanie wytworzony poniższy plik RFL.

```xml
<?xml version="1.0" encoding="utf-8"?>
<Steps>
  <ProcessInfo>
    <RFLVersion>1.0</RFLVersion>
    <Resolution>1440x1080</Resolution>
    <Browser>Chrome</Browser>
    <BrowserVersion>87.0.4280</BrowserVersion>
  </ProcessInfo>
  <OpenBrowser>
    <BrowserType>Chrome</BrowserType>
    <MaximizeWindow>true</MaximizeWindow>
    <TargetObjectName>Przegladarka Przesylki</TargetObjectName>
    <Url>https://localhost:44390/</Url>
    <Options />
  </OpenBrowser>
  <BrowserTypeText>
    <BrowserObjectName>Przegladarka Przesylki</BrowserObjectName>
    <Selector>Id=userName</Selector>
    <Selector>Name=userName</Selector>
    <Selector>CssSelector=userName</Selector>
    <Selector>XPath=//input[@id='userName']</Selector>
    <ClearBeforeType>true</ClearBeforeType>
    <Text>Admin</Text>
    <PathToScreenshot>C:\Users\BlueCompany\Downloads</PathToScreenshot>
    <SecondsPast>5</SecondsPast>
  </BrowserTypeText>
  <BrowserTypeText>
    <BrowserObjectName>Przegladarka Przesylki</BrowserObjectName>
    <Selector>Id=password</Selector>
    <ClearBeforeType>true</ClearBeforeType>
    <Text>adminRPA!</Text>
    <PathToScreenshot />
    <SecondsPast>0</SecondsPast>
  </BrowserTypeText>
  <ClickBrowser>
    <BrowserObjectName>Przegladarka Przesylki</BrowserObjectName>
    <Selector>XPath=//button[@type='submit']</Selector>
    <Command>LeftClick</Command>
    <PathToScreenShot />
    <SecondsPast>0</SecondsPast>
    <InnerText />
  </ClickBrowser>
  <ClickBrowser>
    <BrowserObjectName>Przegladarka Przesylki</BrowserObjectName>
    <Selector>LinkText=Zacznij raportowanie</Selector>
    <Command>LeftClick</Command>
    <PathToScreenShot />
    <SecondsPast>0</SecondsPast>
    <InnerText />
  </ClickBrowser>
  <BrowserTypeText>
    <BrowserObjectName>Przegladarka Przesylki</BrowserObjectName>
    <Selector>Id=ConsignmentNumber</Selector>
    <ClearBeforeType>true</ClearBeforeType>
    <Text>1</Text>
    <PathToScreenshot />
    <SecondsPast>0</SecondsPast>
  </BrowserTypeText>
  <BrowserTypeText>
    <BrowserObjectName>Przegladarka Przesylki</BrowserObjectName>
    <Selector>Id=ExpectedDeliveryDate</Selector>
    <ClearBeforeType>false</ClearBeforeType>
    <Text>02</Text>
    <PathToScreenshot />
    <SecondsPast>0</SecondsPast>
```

Kod RFL dla powyższego przypadku procesu wzorcowego.

```
    <SecondsPast>0</SecondsPast>
  </BrowserTypeText>
  <BrowserTypeText>
    <BrowserObjectName>Przegladarka Przesylki</BrowserObjectName>
    <Selector>Id=ExpectedDeliveryDate</Selector>
    <ClearBeforeType>false</ClearBeforeType>
    <Text>12</Text>
    <PathToScreenshot />
    <SecondsPast>0</SecondsPast>
  </BrowserTypeText>
  <BrowserTypeText>
    <BrowserObjectName>Przegladarka Przesylki</BrowserObjectName>
    <Selector>Id=ExpectedDeliveryDate</Selector>
    <ClearBeforeType>false</ClearBeforeType>
    <Text>2020</Text>
    <PathToScreenshot />
    <SecondsPast>0</SecondsPast>
  </BrowserTypeText>
  <BrowserTypeText>
    <BrowserObjectName>Przegladarka Przesylki</BrowserObjectName>
    <Selector>Id=DeliveryDate</Selector>
    <ClearBeforeType>false</ClearBeforeType>
    <Text>02</Text>
    <PathToScreenshot />
    <SecondsPast>0</SecondsPast>
  </BrowserTypeText>
  <BrowserTypeText>
    <BrowserObjectName>Przegladarka Przesylki</BrowserObjectName>
    <Selector>Id=DeliveryDate</Selector>
    <ClearBeforeType>false</ClearBeforeType>
    <Text>12</Text>
    <PathToScreenshot />
    <SecondsPast>0</SecondsPast>
  </BrowserTypeText>
  <BrowserTypeText>
    <BrowserObjectName>Przegladarka Przesylki</BrowserObjectName>
    <Selector>Id=DeliveryDate</Selector>
    <ClearBeforeType>false</ClearBeforeType>
    <Text>2020</Text>
    <PathToScreenshot />
    <SecondsPast>0</SecondsPast>
  </BrowserTypeText>
  <BrowserTypeText>
    <BrowserObjectName>Przegladarka Przesylki</BrowserObjectName>
    <Selector>Id=DeliveryAddress</Selector>
    <ClearBeforeType>false</ClearBeforeType>
    <Text>Wołoska 5</Text>
    <PathToScreenshot />
    <SecondsPast>0</SecondsPast>
  </BrowserTypeText>
  <ClickBrowser>
    <BrowserObjectName>Przegladarka Przesylki</BrowserObjectName>
    <Selector>XPath=//button[@type='submit']</Selector>
    <Command>LeftClick</Command>
    <PathToScreenShot />
    <SecondsPast>0</SecondsPast>
    <InnerText />
  </ClickBrowser>
</Steps>
```

Kod RFL dla powyższego przypadku procesu wzorcowego cd.

Podsumowując w ramach niniejszego etapu prac rozwojowych (ER1) na podstawie przeprowadzonych badań przemysłowych w komplementarnym etapie 1 (EB1) skutecznie zaimplementowano interpreter języka robotycznego oraz oprogramowanie potrafiące rozpoznawać i wykonywać polecenia ze specyfikacji języka dla robotów. W ramach prac rozwojowych przeprowadzono również prototypowe uruchomienie w środowisku klienta firmy DPC, na wybranym procesie biznesowym. Opracowano również prototyp oprogramowania monitorującego i zarządzającego pracą robotów. W szczególności realizacja powyższych kamieni milowych pozwoliła osiągnąć cele istotne do dalszego zastosowania w pracach rozwojowych w kontekście implementacji środowiska robotycznego.

ER2: Implementacja prototypowych modułów systemu przetwarzania danych wejściowych i dostosowaniu do środowiska wykonawczego

W ramach realizacji prac rozwojowych drugiej fazy projektu (ER2) komplementarnych do EB2 wykonano prototyp rozwiązania na podstawie wybranych założeń i elementów PoC z EB2 realizujący dostosowywanie się do zmiennych formatów dokumentów w firmach przetwarzających duże ilości dokumentów, w szczególności finansowych. Prototyp został zweryfikowany w kontekście środowiska biznesowego klienta DPC. Prototyp oprogramowania oparto o zmienny algorytm działania robota pod wpływem decyzji związanych ze zmienionym środowiskiem wykonawczym.

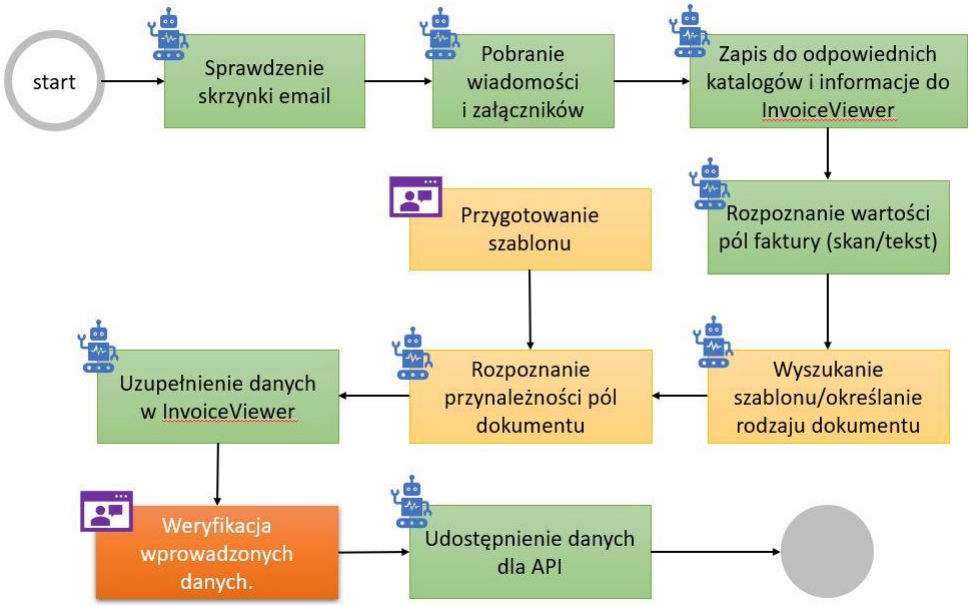
Jednym z podstawowych wyzwań tego etapu było opracowanie oprogramowania pozwalającego na dostosowywanie robota do zmian we wprowadzanych dokumentach, w tym przygotowanie do implementacji na bazie algorytmów najlepiej rokujących wskazanych na etapie badań przemysłowych, w tym modyfikowanie zachowania robota poprzez wprowadzanie zmian w plikach instrukcji i szablonów.

W oparciu o powyższe wyzwania technologiczne opracowano szereg elementów zgodnych z założeniami badawczymi. W szczególności poprzez implementację interpretera RFL potwierdzono możliwości zbudowania oprogramowania, które będzie umożliwiało dostosowywanie w określonym w pracach badawczych zakresie do zmiennych formatów danych wejściowych. W ramach implementacji rozwiązania wykonano w oparciu o opracowane w komplementarnym etapie badań przemysłowych EB2 typologie funkcjonalności dziedzinowych dokonano implementacji prototypu oprogramowania, który pozwolił na zebranie doświadczeń w celu opracowania docelowego rozwiązania.

Zaplanowane prace badawcze w ramach niniejszego etapu badawczego zostały zakończone sukcesem – szczegółowe omówienie przedstawiono poniżej.

Interpreter RFL-a

Interpreter został zaimplementowany przy użyciu języka C#. Potrafi on wykonać dowolny plik zapisany w języku RFL oparty o poniższy uogólniony model procesu biznesowego podlegającego automatyzacji z użyciem środowiska robotycznego.



Interpreter działa w formie aplikacji konsolowej. Jedynym parametrem jaki potrzebuje jest ścieżka do pliku RFL.

Etapy procesu

Na początku plik zostaje sprawdzony pod względem poprawności zapisu. Interpreter sprawdza czy wszystkie główne elementy pliku są rozpoznane z nazwy. Gdy jest to prawdą, rozpoczyna się etap generowania elementów RFL-a. Wszelkie deklaracje zmiennych są umieszczane wewnątrz słownika i inicjalizowane. Każda funkcja jest w sposób dynamiczny kreowana i ma dostęp do słownika zmiennych (kontekstu programu). Przykładowo: Deklarowana jest zmienna typu string i funkcja otwarcia przeglądarki na tej stronie co wcześniejszy string. Zmienna typu string pojawi się w kontekście, a funkcja wywoływania funkcji odnajdzie ją po nazwie i wykorzysta by wstawić do URL-a.

Podsumowując w ramach realizacji prac rozwojowych drugiej fazy projektu (ER2) komplementarnych do EB2 wykonano prototyp rozwiązania na podstawie wybranych założeń i elementów PoC z EB2 realizujący dostosowywanie się do zmiennych formatów dokumentów w firmach przetwarzających duże ilości dokumentów, w szczególności finansowych. Prototyp został zweryfikowany w kontekście środowiska biznesowego klienta DPC. Prototyp oprogramowania oparto o zmienny algorytm działania robota pod wpływem decyzji związanych ze zmienionym środowiskiem wykonawczym. Realizacja powyższych kamieni milowych pozwoliła osiągnąć cele istotne do dalszego zastosowania w pracach rozwojowych w kontekście implementacji prototypu zintegrowanego środowiska robotycznego.

ER3: Implementacja oprogramowania realizującego rejestrowanie procesów

W ramach etapu eksperymentalnych prac rozwojowych (ER3), komptelmentarnego do trzeciego etapu prac badawczych EB3 opracowano prototyp silnika do analizy procesów, wytwarzający zapis procesu w postaci plików poleceń języka dla robotów. Prototyp recordera opracowano na bazie rzeczywistych przykładów procesów biznesowych z obszaru objętego planowanymi wdrożeniami u klientów DPC. Prototyp został zweryfikowany w kontekście środowiska biznesowego jednego z kluczowych klientów korporacyjnych firmy DPC.

Jednym z podstawowych wyzwań tego etapu było przygotowanie na bazie metod opracowanych w EB.3 implementacji rozwiązania pozwalającego na zarejestrowanie przebiegu procesu w komputerze użytkownika i opisanie go przy pomocy języka robotów opracowanego we wcześniejszych etapach badawczych i rozwojowych.

W oparciu o powyższe wyzwania technologiczne opracowano szereg elementów zgodnych z założeniami badawczymi. W szczególności w zakresie możliwości implementacji algorytmów rozpoznawania obrazów i czynności zarejestrowanych innymi metodami, wykonywanych przez człowieka podczas realizacji procesu, a także implementacji rozpoznawania kroków algorytmu na podstawie zgromadzonych danych i przyjętych w EB.3 metod, w tym rozpoznawania węzłów decyzyjnych podejmowania decyzji oraz wykonanie całościowego prototypu aplikacji realizującej procesu generowania zapis czynności w postaci komend języka RFL przy minimalizacji ingerencji operatora po przygotowaniu pliku.

Zaplanowane prace badawcze w ramach niniejszego etapu badawczego zostały zakończone sukcesem – szczegółowe omówienie przedstawiono poniżej.

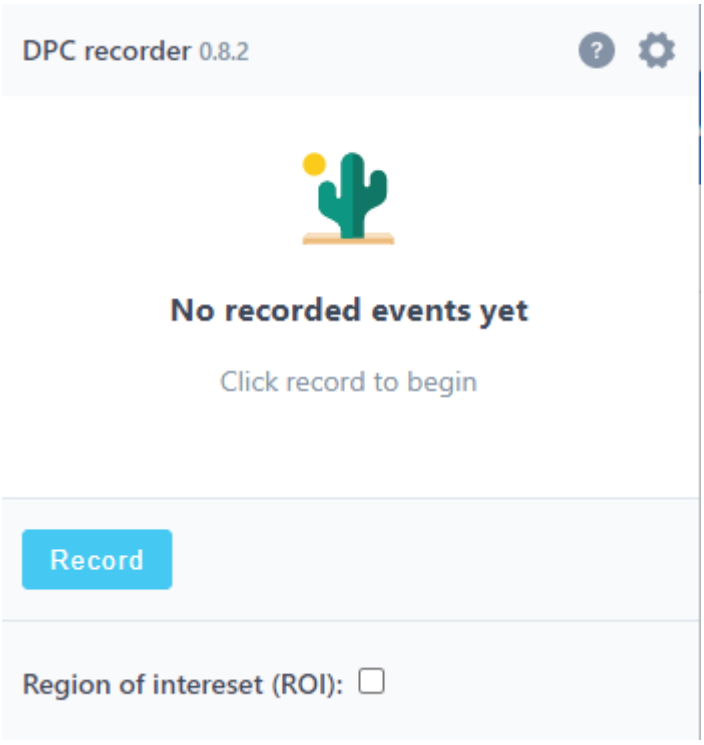
Implementacja modułu recordera

Narzędzie do nagrywania ma za zadanie wygenerować automatycznie logi w formacie RFL-json. Jego użytkownikiem jest pracownik wykonujący proces. Dzięki temu programista nie musi samodzielnie przechodzić wielokrotnie danego procesu by poznać jego szczegóły a wystarczy, że otrzyma przeprocesowane logi. Sam projekt jest forkiem projektu Headless Recorder (https://github.com/checkly/headless-recorder).

Obsługa procesów wzorcowych

Aplikacja działa jako wtyczka przeglądarkowa. Jest kompatybilna z przeglądarkami wykorzystujących Chromium, tzn. takie jak Google Chrome, Edge, Opera.

Nagrywanie rozpoczyna się od wciśnięcia przycisku Record.

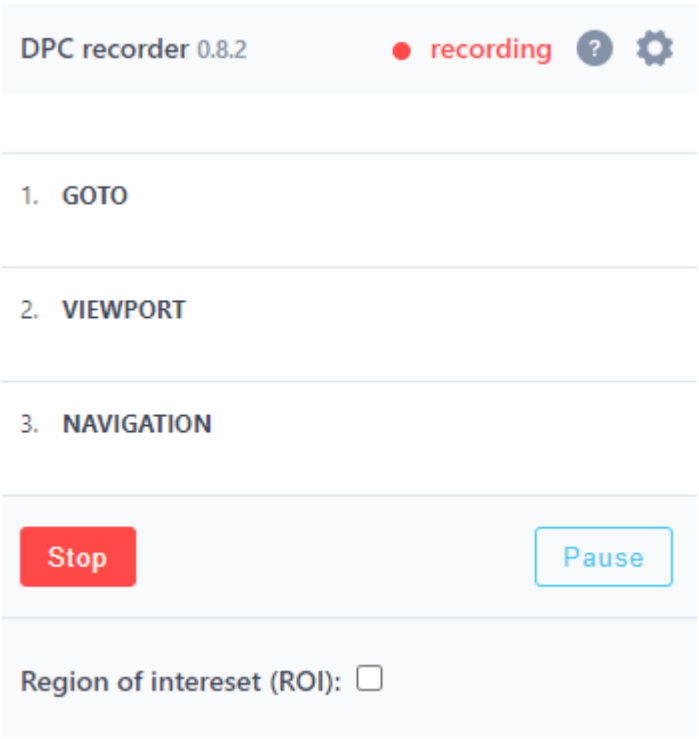


Ekran początkowy nagrywarki.

Przycisk Region of interest jest wykorzystywany, gdy użytkownik patrzy się na dany element w przeglądarce, ale nie wchodzi z nim w interakcję w trakcie procesu.

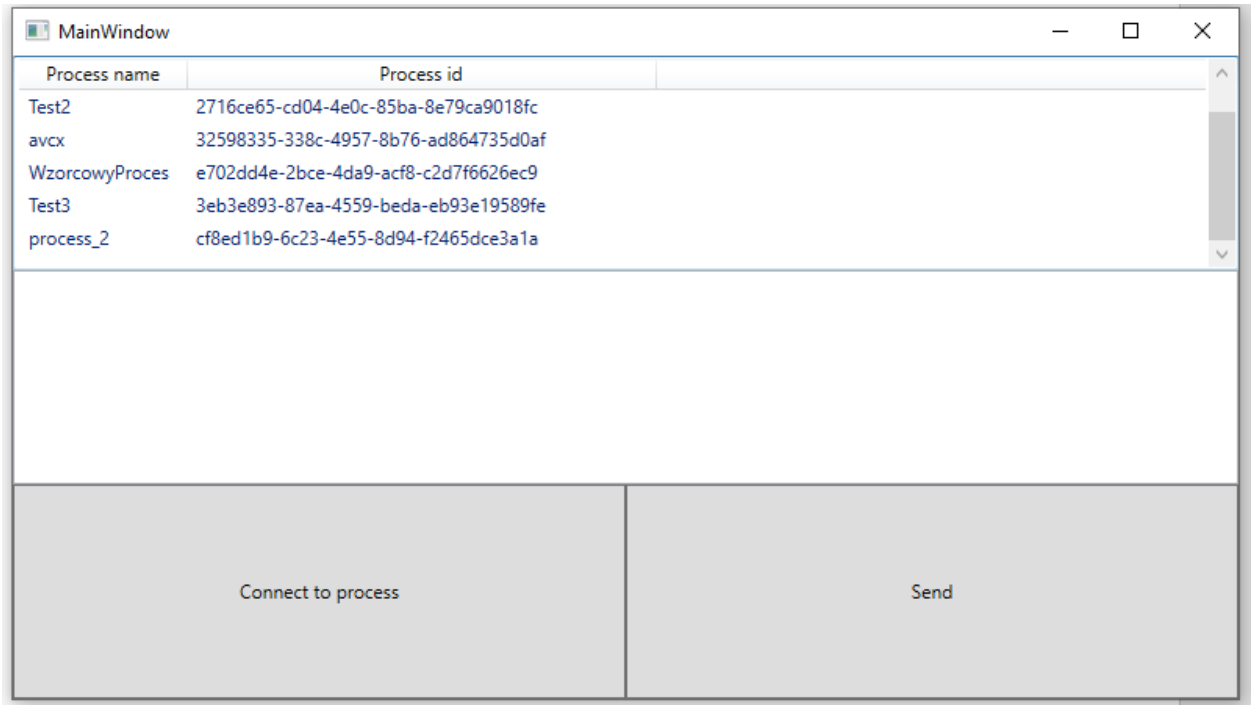
Przykładowe nagranie dwóch zdarzeń przy pomocy recordera.

Po wykonaniu procesu należy wcisnąć przycisk Stop.



Nagrywarka w trakcie działania.

To co znajdzie się w zakładce Recording Result należy następnie skopiować. By przekazać do bazy danych wykonane nagranie należy skorzystać z aplikacji desktopowej.



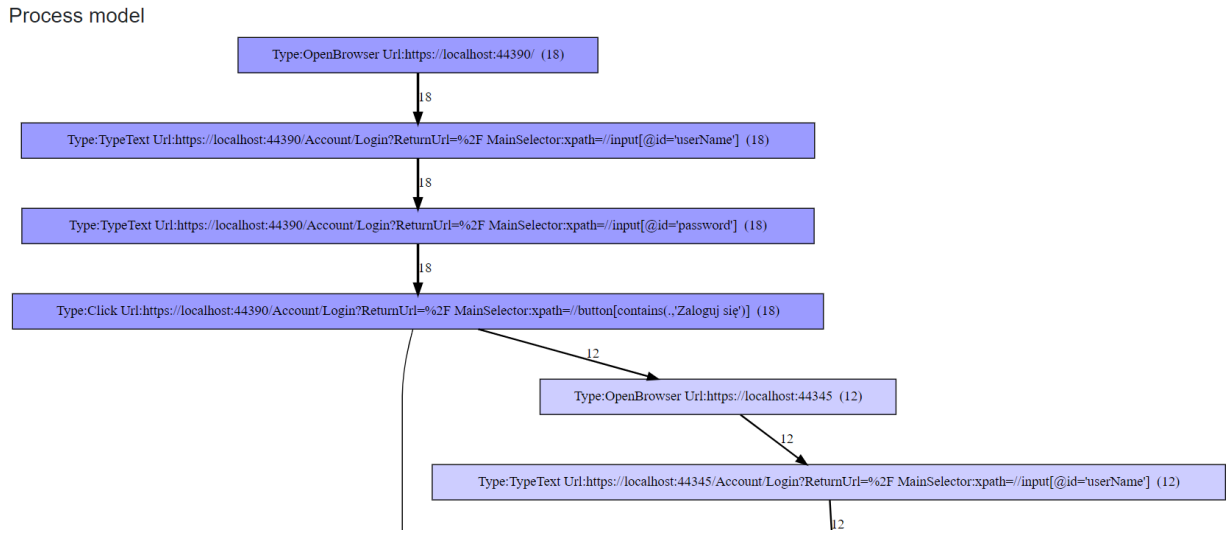
Ekran startowy aplikacji do przesyłania nagrań RFL-json.

Należy wybrać process do którego należy wykonane nagranie, a sam RFL-json opisany szczegółowo we wcześniejszych sekcjach niniejszego opracowania, należy skopiować do textboxa pod tabelką z dostępnymi procesami. Całą operację kończy przycisk Send.

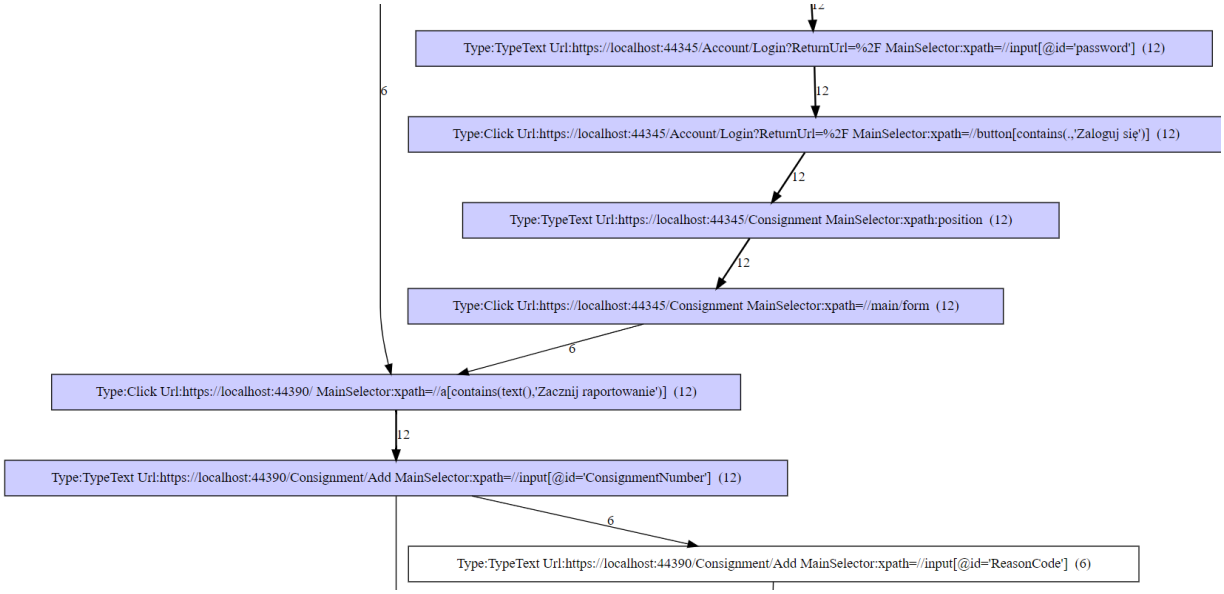
Wykorzystanie zebranych danych

Dzięki równoczesnej analizie logów oraz modelu procesu można dostarczyć operatorowi systemu robotycznego wszystkie najważniejsze informacje dotyczące procesu. Kluczowymi zaletami jest to, że opracowane środowisko umożliwia wygenerowanie wszystkich kroków modelu procesu oraz wzbogacić je o takie informacje jak:

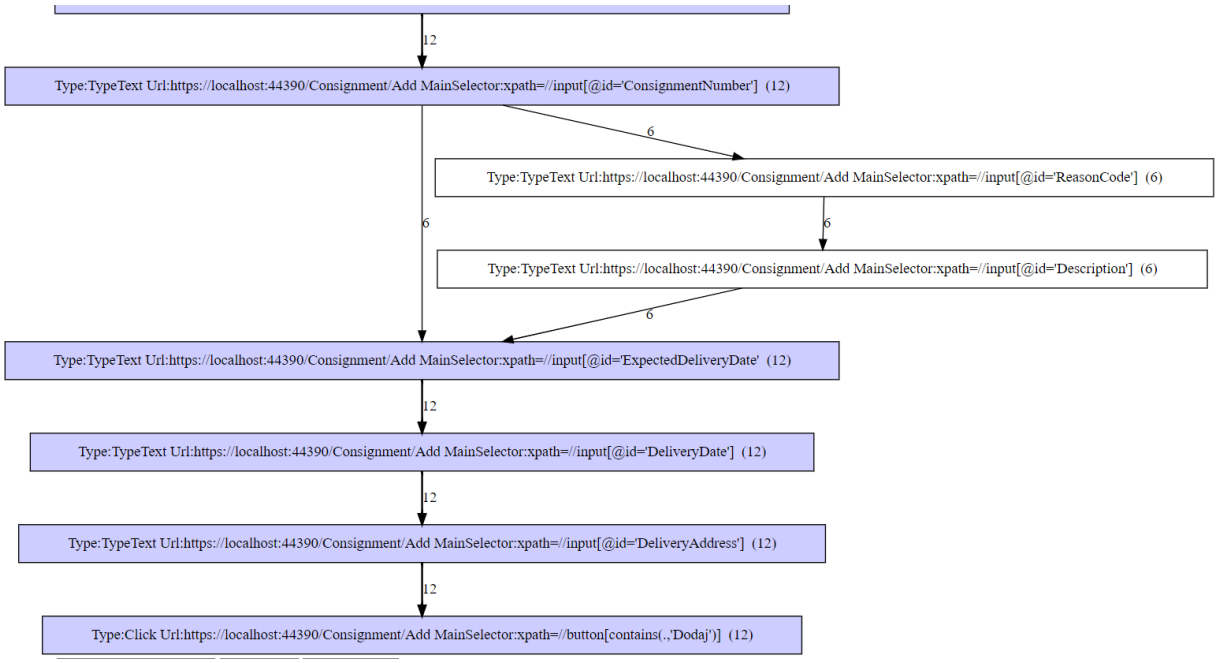
- Screenshot ze zdarzenia,
- Wiele selektorów pozwalających na pisanie bardziej odpornych na zmiany robotów,
- Czas wykonania kroku, dzięki czemu można optymalizować działanie robota,
- Wszystkie dane wejściowe.



Pierwszy etap Procesu Wzorcowego.



Kolejny etap Procesu Wzorcowego.



Końcowy etap Procesu Wzorcowego.

Objaśnienia modelu reprezentacji procesu

Każdy prostokąt to dane zdarzenie z procesu. Na początku zostaje podany typ zdarzenia oraz URL oraz jeśli istnieje główny selektor (wyznaczony przez zaimplementowany algorytm). Na samym końcu zawarta jest wartość liczbowa wskazująca liczbę instancji danego wydarzenia.

Pomiędzy prostokątami są strzałki, stanowiące krawędzie grafu, które wskazują na przepływ procesu. Przy każdej krawędzi jest podana liczba przejść użytkownika przez daną ścieżkę. W końcowym etapie Procesu Wzorcowego zamieszczonego przykładu można zauważyć, że w 50% przypadkach była potrzeba uzupełniania w formularzu kodu opóźnienia oraz jego opisu. Każde zdarzenie posiada pewne szczegóły. Poniżej pokazano, jak wygląda jeden z kroków dla Wzorcowego Procesu.

TypeText element

Browser id: 1

Url: https://localhost:44390/Consignment/Add

Min time: 0.0008

Avg time: 0.0010916666666666668

Max time: 0.0019

Selectors
id=ExpectedDeliveryDate
name=ExpectedDeliveryDate
css=#ExpectedDeliveryDate
xpath=//input[@id='ExpectedDeliveryDate'
xpath=//div[4]/input

Inserted values
02122020
10122020
10122020
02122020
02122020
02122020
02122020
10122020

Szczegóły kroku dla Wzorcowego Procesu.

Można w nim wyróżnić szereg elementów, w tym:

- Ścieżka Url na jakim znajduje się przycisk,
- Najszybszy, średni i najdłuższy zarejestrowany czas do kolejnego zdarzenia,
- Zbiór selektorów,
- Wpisane wartości do kontrolki.

Dzięki takim informacjom operator systemu robotycznego może zaimplementować blisko 100% robota bez potrzeby bezpośredniego ręcznego dostępu do systemu.

Generowanie kodu pośredniczącego

Ponieważ RFL jest specyficznym językiem specjalnego zastosowania (DSL – Domain Specific Language), zostało zaimplementowane generowanie kodu do ogólnie znanych języków, w szczególności C# stanowiącego podstawę środowiska .NET.

```
IWebDriver browser1 = new ChromeDriver();
                    browser1.Manage().Window.Maximize();
                    browser1.Navigate().GoToUrl(@"https://localhost:44390/");
                    browser1.FindElement(By.XPath("//input[@id='userName']")).SendKeys("Admin");
                browser1.FindElement(By.XPath("//input[@id='password']")).SendKeys("adminRPA!");
                browser1.FindElement(By.XPath("//button[contains(.,'Zaloguj się')]")).Click();
            bool condition0 = true;
bool condition1 = true;
if(condition0){
browser1.FindElement(By.XPath("//a[contains(text(),'Zacznij raportowanie')]")).Click();
            string variable0 = "";
                browser1.FindElement(By.XPath("//input[@id='ConsignmentNumber']")).SendKeys(variable0);
                bool condition2 = true;
bool condition3 = true;
if(condition2){
string variable1 = "";
                browser1.FindElement(By.XPath("//input[@id='ExpectedDeliveryDate'")).SendKeys(variable1);
                string variable2 = "";
                browser1.FindElement(By.XPath("//input[@id='DeliveryDate']")).SendKeys(variable2);
                string variable3 = "";
                browser1.FindElement(By.XPath("//input[@id='DeliveryAddress']")).SendKeys(variable3);
                browser1.FindElement(By.XPath("//button[contains(.,'Dodaj')]")).Click();
            return;
}
else if(condition3){
browser1.FindElement(By.XPath("//input[@id='ReasonCode']")).SendKeys("2");
                browser1.FindElement(By.XPath("//input[@id='Description']")).SendKeys("test desc");
                string variable4 = "";
                browser1.FindElement(By.XPath("//input[@id='ExpectedDeliveryDate'")).SendKeys(variable4);
                string variable5 = "";
                browser1.FindElement(By.XPath("//input[@id='DeliveryDate']")).SendKeys(variable5);
                string variable6 = "";
                browser1.FindElement(By.XPath("//input[@id='DeliveryAddress']")).SendKeys(variable6);
                browser1.FindElement(By.XPath("//button[contains(.,'Dodaj')]")).Click();
            return;
}
}
else if(condition1){
```

Kod wygenerowany na podstawie modelu.

```
IWebDriver browser2 = new ChromeDriver();
                    browser2.Manage().Window.Maximize();
                    browser2.Navigate().GoToUrl(@"https://localhost:44345");
                    browser2.FindElement(By.XPath("//input[@id='userName']")).SendKeys("Admin");
                browser2.FindElement(By.XPath("//input[@id='password']")).SendKeys("adminRPA!");
                browser2.FindElement(By.XPath("//button[contains(.,'Zaloguj się')]")).Click();
            string variable7 = "";
                browser2.FindElement(By.XPath("position")).SendKeys(variable7);
                browser2.FindElement(By.XPath("//main/form")).Click();
            browser1.FindElement(By.XPath("//a[contains(text(),'Zacznij raportowanie')]")).Click();
            string variable8 = "";
                browser1.FindElement(By.XPath("//input[@id='ConsignmentNumber']")).SendKeys(variable8);
                bool condition4 = true;
bool condition5 = true;
if(condition4){
string variable9 = "";
                browser1.FindElement(By.XPath("//input[@id='ExpectedDeliveryDate'")).SendKeys(variable9);
                string variable10 = "";
                browser1.FindElement(By.XPath("//input[@id='DeliveryDate']")).SendKeys(variable10);
                string variable11 = "";
                browser1.FindElement(By.XPath("//input[@id='DeliveryAddress']")).SendKeys(variable11);
                browser1.FindElement(By.XPath("//button[contains(.,'Dodaj')]")).Click();
            return;
}
else if(condition5){
browser1.FindElement(By.XPath("//input[@id='ReasonCode']")).SendKeys("2");
                browser1.FindElement(By.XPath("//input[@id='Description']")).SendKeys("test desc");
                string variable12 = "";
                browser1.FindElement(By.XPath("//input[@id='ExpectedDeliveryDate'")).SendKeys(variable12);
                string variable13 = "";
                browser1.FindElement(By.XPath("//input[@id='DeliveryDate']")).SendKeys(variable13);
                string variable14 = "";
                browser1.FindElement(By.XPath("//input[@id='DeliveryAddress']")).SendKeys(variable14);
                browser1.FindElement(By.XPath("//button[contains(.,'Dodaj')]")).Click();
            return;
}
```

Kod wygenerowany na podstawie modelu cd.

Tak powstały kod wymaga jedynie obsługi instrukcji warunkowych w celu doprecyzowania zidentyfikowanych węzłów przejścia w oparciu o ścieżki użytkowników.

Podsumowując, w ramach niniejszego etapu eksperymentalnych prac rozwojowych (ER3), komptelmentarnego do trzeciego etapu prac badawczych (EB3) opracowano prototyp silnika do analizy procesów (recordera), wytwarzający zapis procesu w postaci plików poleceń języka dla robotów. Prototyp opracowano na bazie rzeczywistych przykładów procesów biznesowych z obszaru objętego planowanymi wdrożeniami u klientów DPC. Ralizacja powyższych kamieni milowych pozwoliła dopiąć proces implementacji w ramach projektu prototypu zintegrowanego prototypu środowiska robotycznego, który został

zweryfikowany w kontekście środowiska biznesowego jednego z kluczowych klientów korporacyjnych firmy DPC.