Politechnika Wrocławska

SSDL_SIG

Samodzielny Interfejs Graficzny narzędzie kompozycji usług złożonych

Opis produktu wytworzonego
w ramach Zespołowego Przedsięwzięcia
Inżynierskiego

skład grupy ZPI:
Dorota Kawalec
Jacek Kowalczyk
Błażej Wolańczyk
Kamil Kopryk
Mateusz Brząkała

Promotor: dr inż. Grzegorz Kołaczek

Praca realizowana w ramach projektu Programu Operacyjnego Innowacyjna Gospodarka, grant numer POIG.01.03.01-00-008/08 (Nowe technologie informacyjne dla elektronicznej gospodarki i społeczeństwa informacyjnego oparte na paradygmacie SOA).

Spis treści

Wstęp

SSDL_SIG (Samodzielny Interfejs Graficzny) to narzędzie wytworzone w ramach projektu IT-SOA, mające na celu umożliwienie łatwej konstrukcji usług złożonych w przyjaznym dla użytkownika środowisku graficznym. Niniejszy dokument ma na celu prezentację możliwości, architektury i działania narzędzia.

Aby ułatwić lekturę dokumentu Czytelnikowi, który nie zna kontekstu powstania tej aplikacji, w ramach niniejszego wstępu wyjaśnione zostaną pojęcia i nazwy własne, które zostaną w ramach prezentacji narzędzia wielokrotnie użyte.

Service Oriented Architecture (SOA)

Architektura rozproszona, w której działanie odbywa się w oparciu o komunikację pomiędzy niezależnymi od siebie usługami, zdefiniowanymi tak, by ich (współ)działanie pozwoliło na zaspokojenie potrzeb użytkownika. Mogą one być napisane z wykorzystaniem różnych technologii, a jedynym sposobem ich komunikacji jest protokół komunikacyjny, umożliwiający wymianę danych.

Usługi

W kontekście architektury SOA usługa nazywa się działający niezależnie od innych element oprogramowania o określonym interfejsie, za pomocą którego udostępnia swoją określoną funkcję. Szczegóły implementacji usługi nie są znane klientowi – działanie usługi definiuje jedynie jej zdefiniowany interfejs wejść/wyjść.

Usługi złożone

Usługi złożone (*smart services*) to usługi powstałe poprzez złożenie ze sobą innych usług w celu dostarczenia bardziej złożonej funkcjonalności.

Projekt IT-SOA

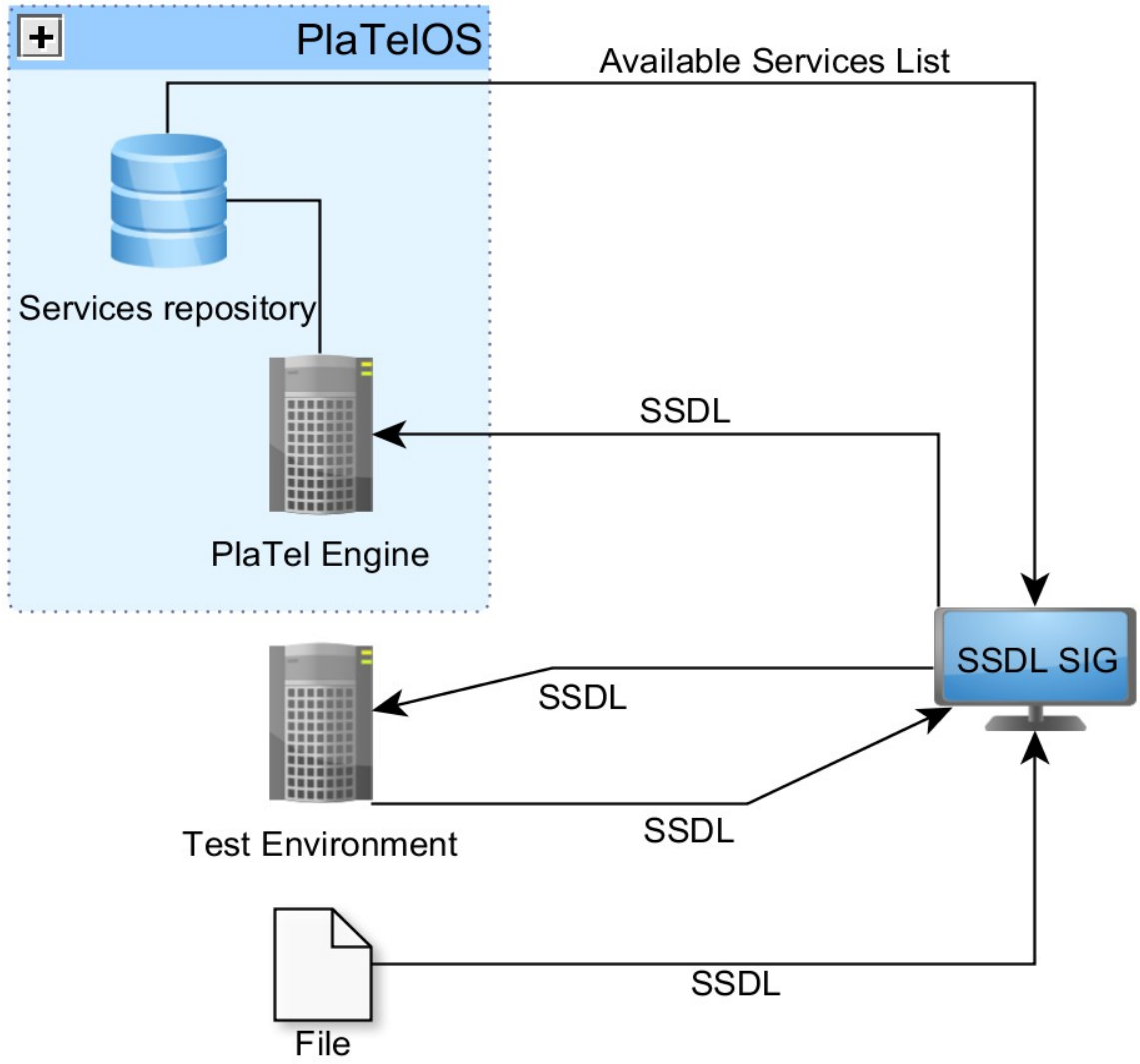
Pełna nazwa projektu to „Nowe technologie informacyjne dla elektronicznej gospodarki i społeczeństwa informacyjnego oparte na paradygmacie SOA”.

Cytując informacje umieszczone na stronie www projektu (https://www.soa.edu.pl/) przez jego organizatorów, „Projekt dotyczy współczesnych technologii informacyjnych działających w systemach rozproszonych. Technologie te, oparte na paradygmacie SOA (ang. Service Oriented Architecture), są stosowane do informatyzacji procesów biznesowych, platform usługowych oraz infrastruktury informatycznej dla e-Science. Istotą tego podejścia jest automatyzacja zarówno uruchamiania pojedynczych usług jak i integracji usług w złożone procesy biznesowe. Zasadniczymi problemami występującymi w tym obszarze są: język opisu usług, proces publikacji usługi przez jej dostarczyciela oraz wyszukiwanie potrzebnych usług przez klienta, a także wykorzystujące semantykę protokoły kompozycji usług w złożone i często długotrwałe procesy biznesowe.”

PlaTel

Platforma Planowania i Monitorowania Usług Teleinformatycznych. System wspierający planowanie i monitoring usług teleinformatycznych. Znajduje on zastosowanie w systemach, w których realizacja różnych procesów biznesowych zależy od liczby i jakości usług teleinformatycznych, usługi teleinformatyczne są dostarczane przez różnych dostawców i operatorów (różne taryfy, różny zasięg przestrzenny, dostępność, bezpieczeństwo) i usługi dziedzinowych systemów usługowych bazujących na dostępnych usługach teleinformatycznych (zdrowie, edukacja, sieci domowe i samochodowe, inteligentne systemy monitorowania, itd.). PlaTel jest zintegrowaną platformą wspierającą procesy biznesowe w dystrybuowanych technologiach teleinformatycznych opartych o paradygmat SOA, częścią projektu IT-SOA w którym biorą udział inne uczelnie takie jak: AGH, Uniwerstet Ekonomiczny w Poznaniu, Politechnika Poznańska, Instytut Podstaw Informatyki PAN. Więcej informacji na http://www.platel.com.pl/ .

SSDL_SIG zaprojektowane jest jako aplikacja wspierająca tworzenie usług złożonych w ramach funkcjonowania PlaTelu.



Ryc. 1 - Współpraca SSDL-SIG z PlaTelOS

SSDL

SSDL, czyli *Smart Service Description Language*, to oparty o XML język opisu usług złożonych powstały na Politechnice Wrocławskiej w ramach projektu naukowego „Nowe technologie informacyjne dla elektronicznej gospodarki i społeczeństwa informacyjnego oparte na paradygmacie SOA”. Rozszerza on możliwości używanego zazwyczaj WSDL o reprezentację wymagań niefunkcjonalnych oraz usług złożonych.

Problemy związane z tworzeniem usług złożonych

Aplikacja SSDL_SIG opracowana została z myślą o rozwiązaniu następujących problemów, z którymi borykali się twórcy platformy PlaTel:

- **język SSDL opiera się na XML-u**
  - co za tym idzie, tworzenie usług złożonych jest procesem żmudnym i czasochłonnym, nawet dla programisty
  - jednocześnie zapis XML-owy jest mało czytelny dla człowieka i długo trwa analiza struktury zapisanego w SSDL grafu
  - potrzebna jest walidacja XML ze schemą
- **z powyższego wynika - brak odpowiedniego narzędzia umożliwiającego łatwe tworzenie lub modyfikację usług złożonych**
- **brak połączenia pomiędzy interfejsem graficznym, a mediatorami ontologii i usług atomowych, czyli problem cross-domain**
- **brak aplikacji do wizualizacji dla środowiska do testów**

Dostarczane funkcjonalności

- **czytelna wizualizacja i automatyczne rozmieszczanie grafu (przy pomocy algorytmu genetycznego) po sparsowaniu SSDL z pliku .xml**
- **dwa tryby pracy:**
  - CXSE – budowanie i edycja usług złożonych
  - BSPE – oparty o zapis w SSDL edytor procesów biznesowych
- **edytor usług złożonych** o następujących możliwościach:
  - możliwość podglądu zawartości SSDL w formie graficznej
  - edycji usługi złożonej
  - budowa grafu nowej usługi złożonej
  - eksport wyników pracy do SSDL, wraz z walidacją
- **edytor procesów biznesowych** o następujących możliwościach:
  - podglądu procesu biznesowego w formie graficznej
  - edycji procesu biznesowego
  - budowa nowego procesu biznesowego
  - eksport wyników pracy do SSDL
- **konfigurowalność i możliwość nieskończonej rozbudowy aplikacji**
- **połączenia SOAP z serwerami dostarczającymi:**
  - usługi atomowe
  - ontologię konceptów
  - ontologię nazw dla wymagań QoS
  - pary działań i obiektów dla słów kluczowych w procesach biznesowych
- **dwie wersje językowe**
  - polska
  - angielska
  - możliwość dopisania dodatkowych wersji językowych, o ile użytkownik zna javaScript

Zalety naszego rozwiązania

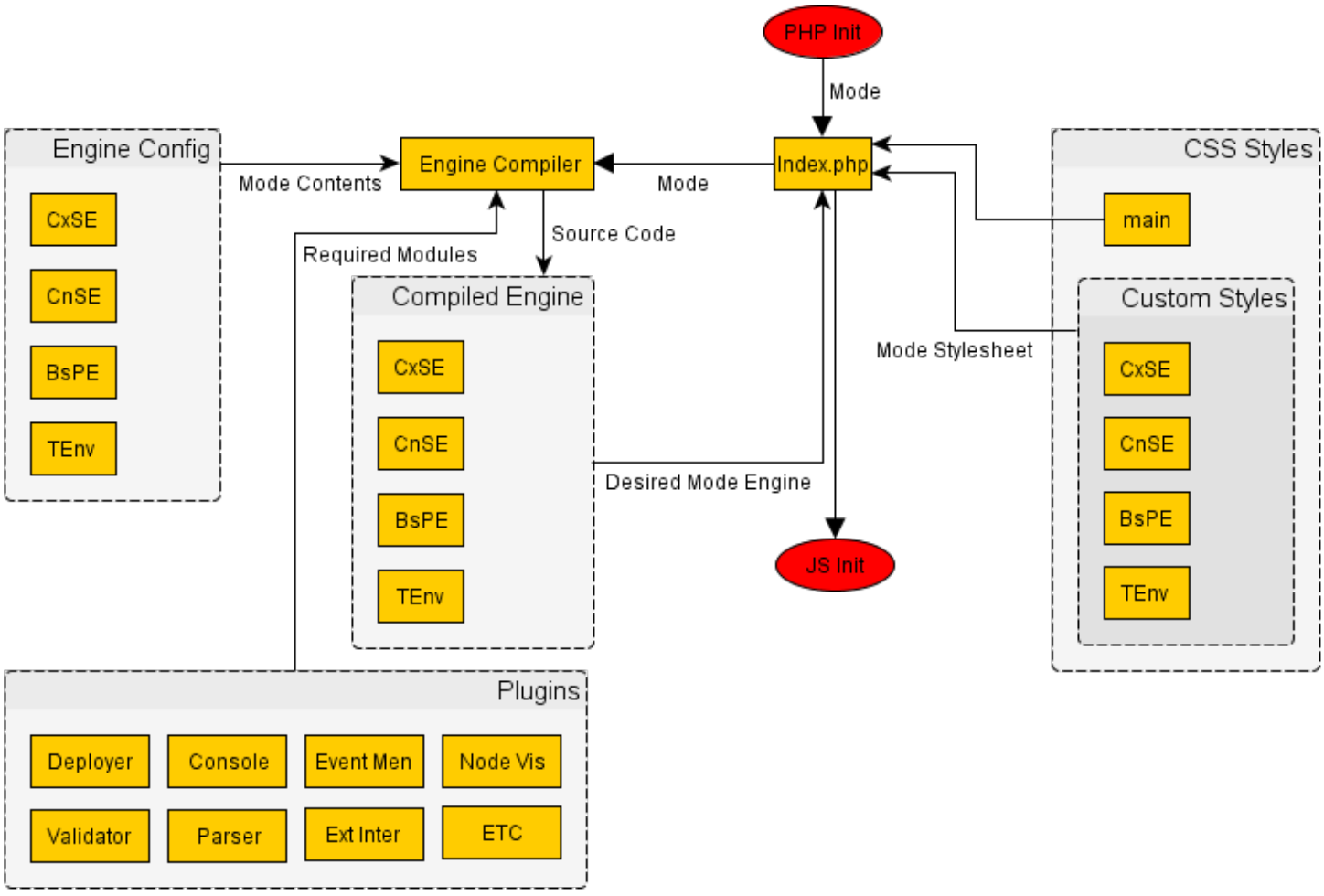
- **aplikacja SSDL_SIG jest wynikiem poprzednich doświadczeń naszego zespołu w pracy nad SSDL_GUI, poprzednią iteracją narzędzia – graficznego edytora usług złożonych**
- **działanie niezależne od systemu operacyjnego, w oknie przeglądarki**
- **obsługiwane przeglądarki:**
  - Google Chrome – zalecane ze względu na wydajność SVG
  - Internet Explorer
  - Mozilla Firefox
  - Safari
  - Opera - stabilna obsługa części funkcjonalności, niezalecana
- **modułowa budowa i architektura oparta o paradygmat sterowany zdarzeniami:**
  - modyfikacja kilku plików i dopisanie własnych modułów pozwala na dodanie nowego trybu
  - edycja plików .css i odpowiedzialnych za wygląd plików .js pozwala na modyfikację kolorystyki i układu aplikacji
  - możliwość dodawania nowych typów wierzchołków do grafu opisującego usługę złożoną
- **każdy element aplikacji może być zmieniany i rozbudowywany, bez ponoszenia dodatkowych kosztów związanych z przebudową innych modułów**
- **intuicyjny interfejs:**
  - gdzie użytkownikowi przedstawiane są tylko najważniejsze informacje
  - dymki z podpowiedziami
  - ikony pełniące funkcję informacyjną
  - graficzne menu
  - menu kontekstowe
  - skróty klawiszowe
- **aplikacja jest skalowalna, rozszerzalna, i pozwala na obniżenie kosztów rozwoju oprogramowania usługowego.**

Architektura aplikacji

SSDL_SIG składa się z back-endu napisanego w PHP oraz front-endu w javaScript. Elementy javaScriptowe funkcjonują jako moduły, których działaniem steruje zarządca zdarzeń (*Event Manager*).

Podział na front-end i back-end

Po stronie serwera znajduje się back-end w PHP, natomiast po stronie klienta front-end w javaScript. W zależności od wybranego trybu pracy (np. edytor usług złożonych lub edytor procesów biznesowych) silnik w PHP wybiera odpowiedni dla danego trybu plik konfiguracyjny i kompiluje silnik javaScriptowy z plików modułów, które zostały wskazane w tym pliku konfiguracyjnym.



Ryc. 2 - architektura SSDL_SIG

Taka budowa pozwala na maksymalną elastyczność aplikacji, umożliwiając jej nieskończoną rozbudowę – jeżeli w przyszłości pojawiłoby się zapotrzebowanie na nowy tryb pracy, wystarczyłoby napisanie specyficznych dla niego modułów oraz pliku konfiguracyjnego, a silnik w PHP skompilowałby z nich adekwatną do potrzeb aplikację w javaScript.

Moduły

Moduł jest to niezależny byt programistyczny dostarczający określoną funkcjonalność, spełniający określony interfejs. Front-end SSDL_SIG zbudowany jest z modułów, które komunikują się ze sobą za pomocą zdarzeń.

Co daje zastosowanie podziału aplikacji na moduły?

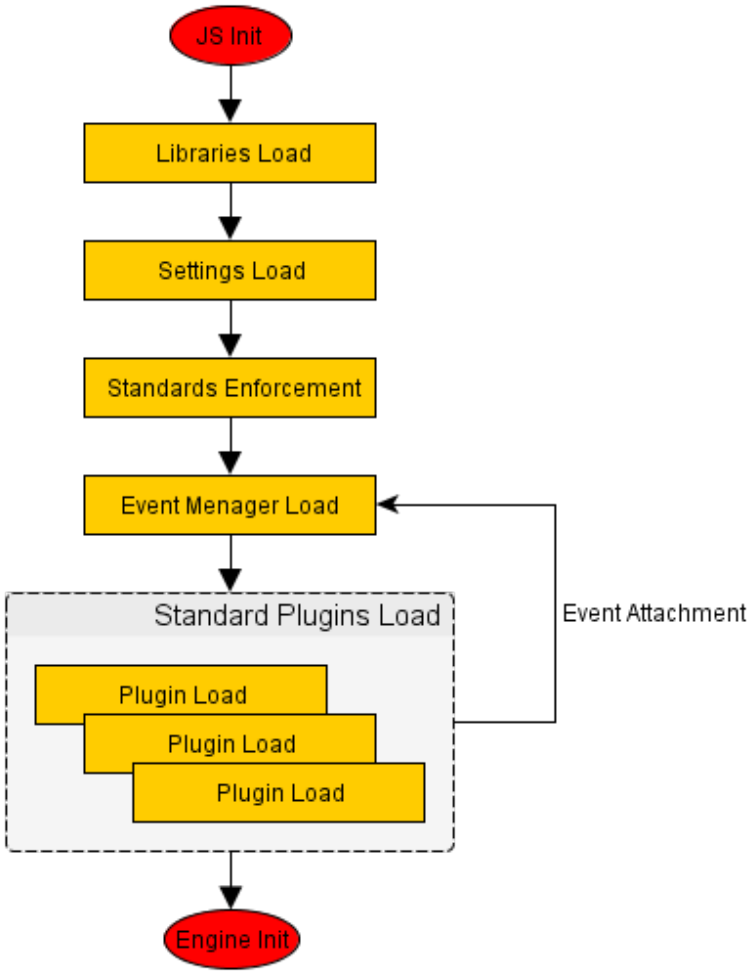
- łatwość dodawania kolejnych funkcjonalności
- łatwość modyfikacji istniejących funkcjonalności
- łatwość konfiguracji, jakie funkcjonalności potrzebujemy realizować.

Dzięki podziałowi na moduły SSDL_SIG jest aplikacją łatwo skalowalną, o dużej elastyczności i rozszerzalności, co pozwala na obniżenie kosztów dalszego rozwoju tego oprogramowania.

Kolejne działania podczas uruchamiania javaScriptowego silnika SSDL_SIG:

- załadowanie bibliotek
- załadowanie ustawień
- narzucenie standardów (w celu uzyskania kompatybilności z możliwie największą liczbą przeglądarek)
- załadowanie zarządcy zdarzeń
- ładowanie kolejnych modułów; obsługiwane przez nie zdarzenia są rejestrowane przez zarządcę

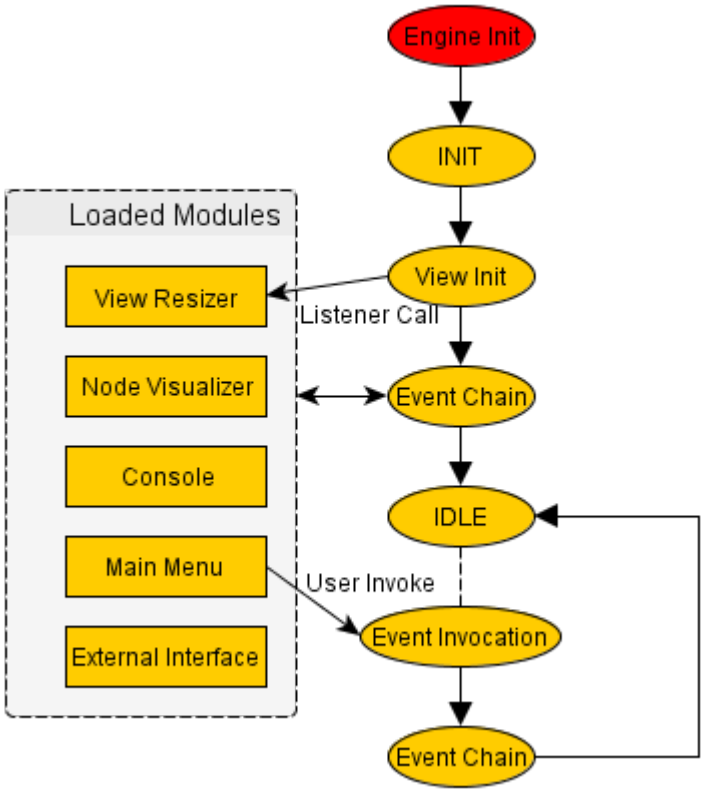
W ten sposób zbudowany zostaje szkielet, na bazie którego uruchamiane są moduły niekluczowe, których konfiguracja zależna jest od załadowanego pliku konfiguracyjnego.



Ryc. 3 - kolejność inicjalizacji javaScriptowych elementów SSDL_SIG

Po zakończeniu procesu uruchamiania silnika wykonywane są następujące operacje:

- inicjalizacja modułów graficznych
- wykonuje się sekwencja zdarzeń, ustawiających położenia obiektów na ekranie
- aplikacja przechodzi w stan bezczynności, dopóki nie zostanie wykonane żadne działanie ze strony użytkownika
- kiedy użytkownik wykona jakąś czynność, wywołane zostaną zdarzenia, a odpowiednie moduły na nie zareagują
- następnie aplikacja wróci do stanu oczekiwania.

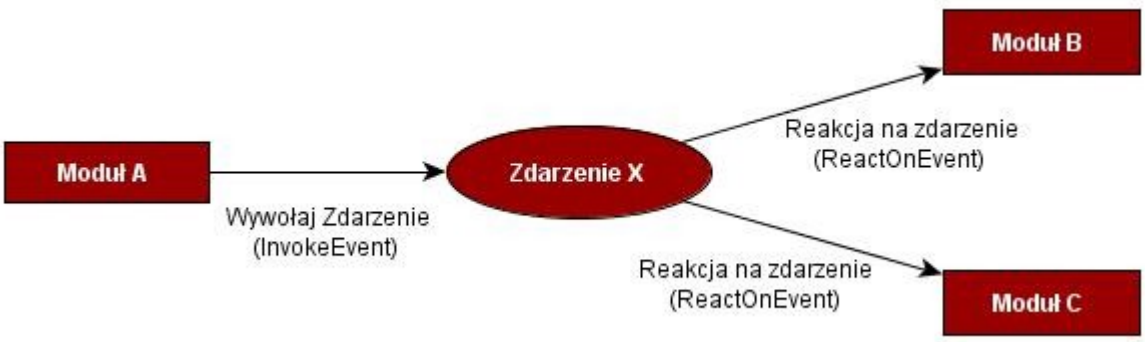


Ryc. 4 - Uruchamianie SSDL_SIG

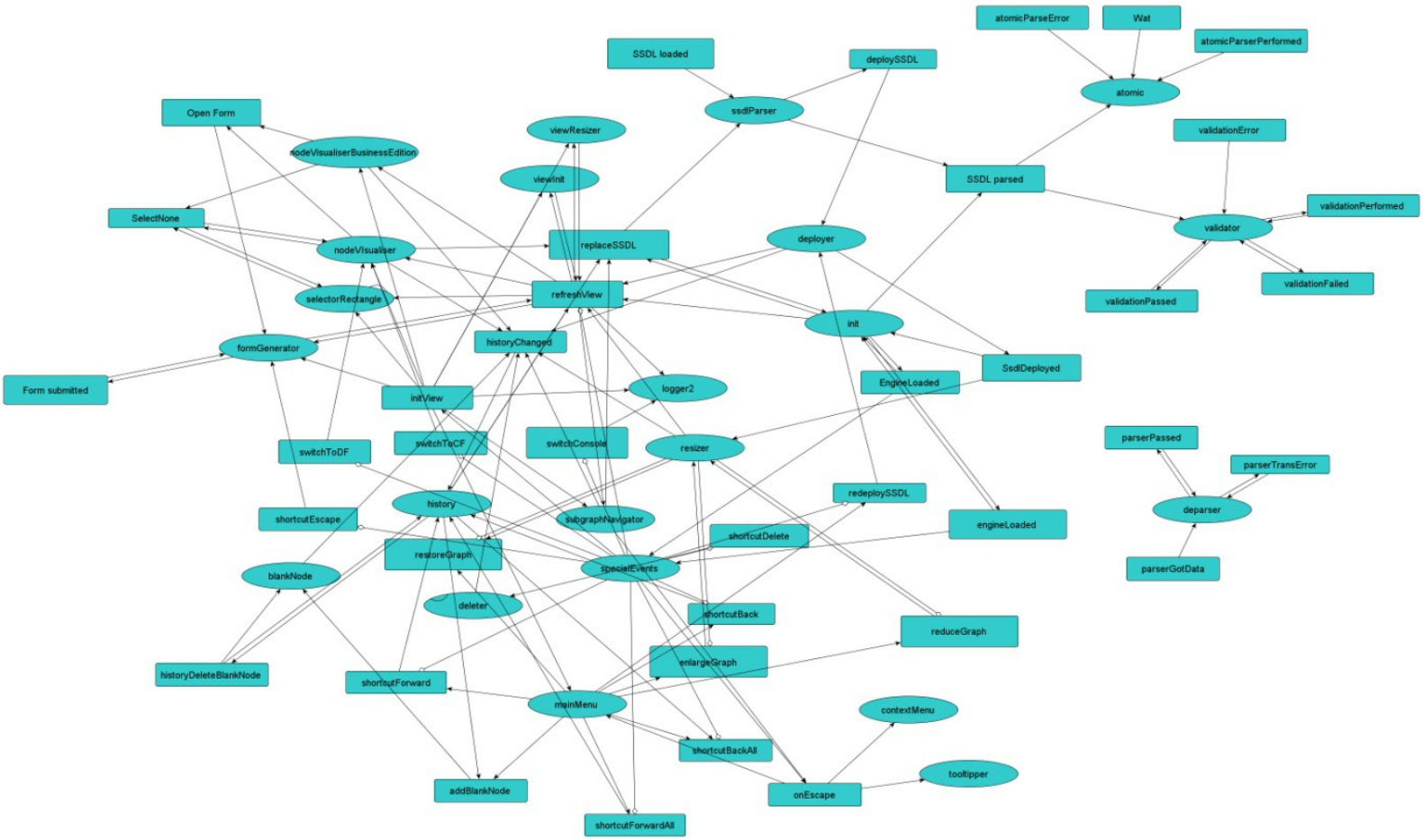
Zarządca zdarzeń

Współpraca pomiędzy modułami w SSDL_SIG została rozwiązana poprzez zarządcę zdarzeń (*Event Managera*).

Jest to zdecydowanie najważniejszy moduł aplikacji, napisany w oparciu o paradygmat programowania sterowanego zdarzeniami (*Event-Driven javaScript*). Moduł ten obsługuje natywne zdarzenia przeglądarki oraz wewnętrzne zdarzenia aplikacji. Ułatwia dodawanie nowych modułów bez ingerencji w już istniejące poprzez uproszczenie komunikacji między nimi.



Ryc. 5 - schemat działania zarządcy zdarzeń - moduł A wywołuje zdarzenie, na które nasłuchują moduły B i C



Ryc. 6 - schemat powiązań pomiędzy modułami, a zdarzeniami

Powyższy schemat ilustruje skalę aplikacji, jak również liczbę modułów i zdarzeń, które są między nimi przesyłane. Aplikacja o podobnej skali, napisana bez wykorzystania paradygmatu sterowania zdarzeniami, byłaby o wiele bardziej skomplikowana programistycznie i hermetyczna, podczas gdy SSDL_SIG w dalszym ciągu podatne jest na rozbudowę.

Ta skalowalność jest niezaprzeczalnie główną zaletą naszego rozwiązania, pozwalającą na nieskończony rozwój aplikacji zarówno przez jej oryginalnych twórców, jak i innych programistów. Jedynym wymaganiem jest znajomość javaScriptu, który stanowi podstawę zarówno modułów, jak i plików ustawień, które mają postać obiektów javaScript.

Spis zastosowanych modułów

W niniejszej sekcji dokumentu przedstawione i krótko opisane zostaną moduły, które umieściliśmy w SSDL_SIG. W celu poprawienia czytelności, zostały podzielone na kategorie w zależności od swojej roli w funkcjonowaniu aplikacji.

Wizualizacja – kanwa

Następujące moduły odpowiadają za obsługę kanwy, na której wyświetlany jest graf:

- **arrow**

  Moduł odpowiada za rysowanie krawędzi grafu w dwóch trybach widoku: przepływu kontroli i przepływu danych.

- **deployer**

  Moduł odpowiada za automatyczne rozmieszczanie wierzchołków na kanwie. W celu obliczenia jak najlepszych położeń wierzchołków w grafie wykorzystany jest algorytm genetyczny.

- **nodeVisualizer**

  Moduł jest kluczowym elementem widoku w aplikacji; odpowiada za rysowanie i przerysowywanie zarówno pojedynczych wierzchołków, jak i całego grafu, łączenie ze sobą wierzchołków, obsługę zdarzeń myszy.

- **resizer**

  Moduł umożliwia zmianę rozmiaru elementów grafu, zarówno oddalenie, jak i przybliżenie.

- **selectorRectangle**

  Moduł pozwala na zaznaczenie metodą „przeciągnij i upuść” wielu wierzchołków jednocześnie. W celu uzyskania maksymalnej intuicyjności obsługi, zasady zaznaczania, odznaczania i przesuwania wierzchołków są identyczne, jak w przypadku ikon na pulpicie systemu Windows.

Wizualizacja – GUI

Moduły, które odpowiadają za wyświetlanie i obsługę elementów udostępnionego użytkownikowi interfejsu:

- **contextMenu**

  Menu kontekstowe, które może być przymocowane do dowolnego elementu interfejsu.

- **formGenerator**

  Generator formularzy, który w oparciu o pliki konfiguracyjne na poczekaniu generuje wielostronicowe formularze zgodne z HTML5.

- **logger**

  Moduł odpowiada za tworzenie konsoli wyświetlającej komunikaty o

funkcjonowaniu aplikacji, podzielone na kategorie błędów (np. niepowodzenie połączenia z repozytorium), ostrzeżeń (np. o brakujących danych w grafie) i informacji (np. o pomyślnej walidacji pliku .xml).

- **mainMenu**

  Belka menu u góry, zawierająca dostępne użytkownikowi opcje.

- **tooltipper**

  Moduł udostępnia API do dodawania dymków z podpowiedziami do dowolnych elementów zarówno kanwy, jak i interfejsu.

Dane

Moduły, które odpowiadają za operacje na modelu danych:

- **atomic**

  Moduł, który łączy się z repozytoriami poprzez protokół SOAP i pobiera informacje o dostępnych usługach atomowych, listy konceptów i nazw parametrów QoS.

- **blankNode**

  Moduł pozwala na dodawanie do kanwy pustych wierzchołków typów: funkcjonalność, mediator, usługa emulująca, oraz par: wierzchołek początkowy i końcowy, wierzchołki definiujące warunek.

- **deleter**

  Moduł obsługuje usuwanie wierzchołków z grafu. Ta pozornie banalna operacja wymaga szeregu czynności w związku z wzajemnymi powiązaniami pomiędzy elementami grafu.

- **deparser**

  Moduł, który zapisuje utworzony przez użytkownika graf do pliku .xml, wykorzystując język SSDL.

- **externalInterface**

  Moduł odpowiada za komunikację zewnętrzną poprzez AJAX.

- **init**

  Moduł uruchamiany podczas startu aplikacji i synchronizujący jej elementy.

- **validator**

  Moduł sprawdzający poprawność pliku .xml w oparciu o schemę.

- **ssdlParser**

  Moduł odpowiada za parsowanie pliku .xml i konwersję do javaScriptowego modelu danych.

Obsługa zdarzeń

Jak już zostało wspomniane, SSDL_SIG został napisany w oparciu o paradygmat programowania sterowania zdarzeniami, a co za tym idzie obsługa zdarzeń jest istotnym jego elementem.

- **eventManager**

  Najważniejszy moduł w SSDL_SIG, odpowiada za rejestrowanie i obsługę zdarzeń, które są sposobem komunikacji pomiędzy modułami. Tym samym eventManager odpowiada za synchronizowanie działań całej aplikacji.

- **viewInit**

  Moduł odpowiada za inicjalizację widoku przy pierwszym uruchomieniu aplikacji.

- **shortcut**

  Moduł pozwala na używanie przy korzystaniu z aplikacji skrótów klawiaturowych, np. Shift+C i Shift+D w celu przełączenia odpowiednio do trybu przepływu kontroli i przepływu danych.

- **specialEvents**

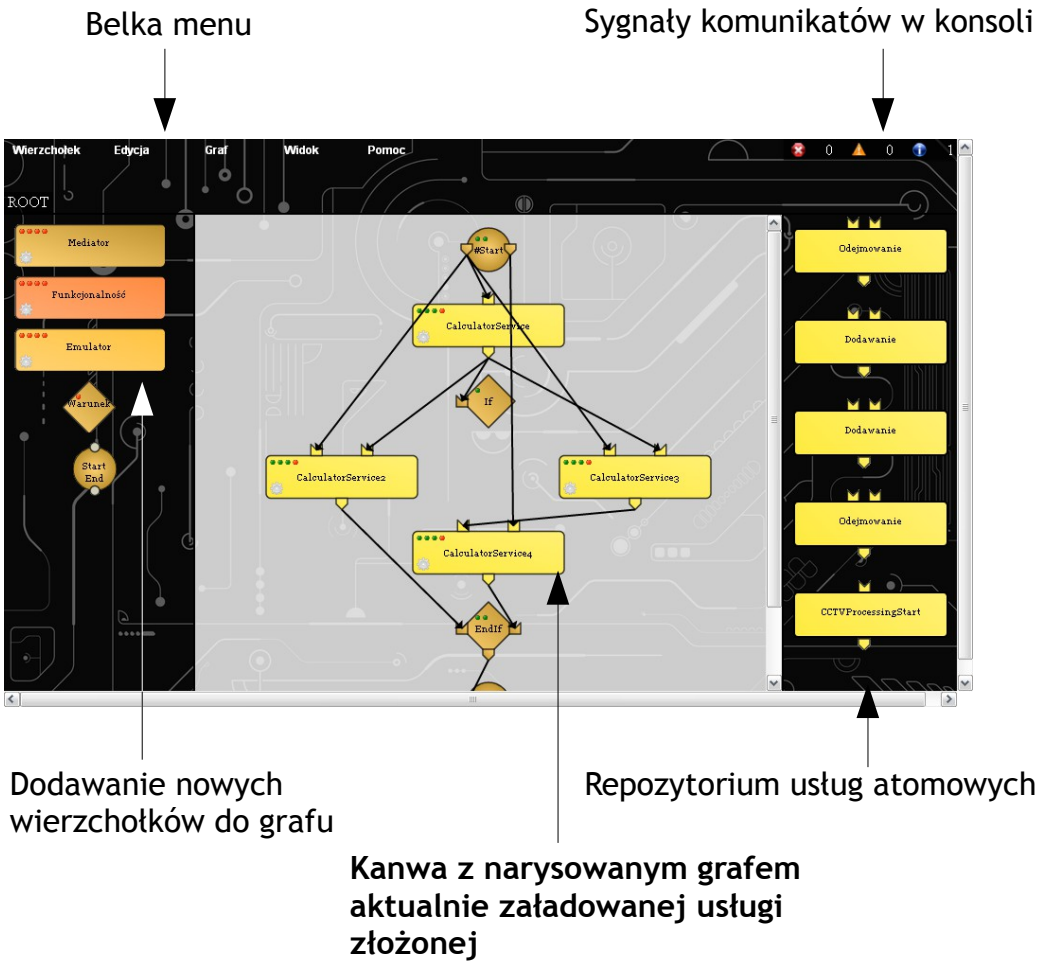
  Moduł obsługuje zdarzenia specjalne, które występują dokładnie jeden raz.

Przegląd aplikacji

W niniejszej sekcji zaprezentowane zostanie ekran główny SSDL_SIG oraz kluczowe elementy interfejsu. W trybie edycji usług złożonych SSDL_SIG oferuje podgląd usług w dwóch widokach: przepływu danych (*DataFlow*) oraz przepływu kontroli (*Control Flow*), ilustrującym kolejność wykonywania elementów usługi złożonej.

Ekran główny

W trybie edycji usług złożonych, w trybie widoku przepływu danych, ekran główny aplikacji prezentuje się następująco:



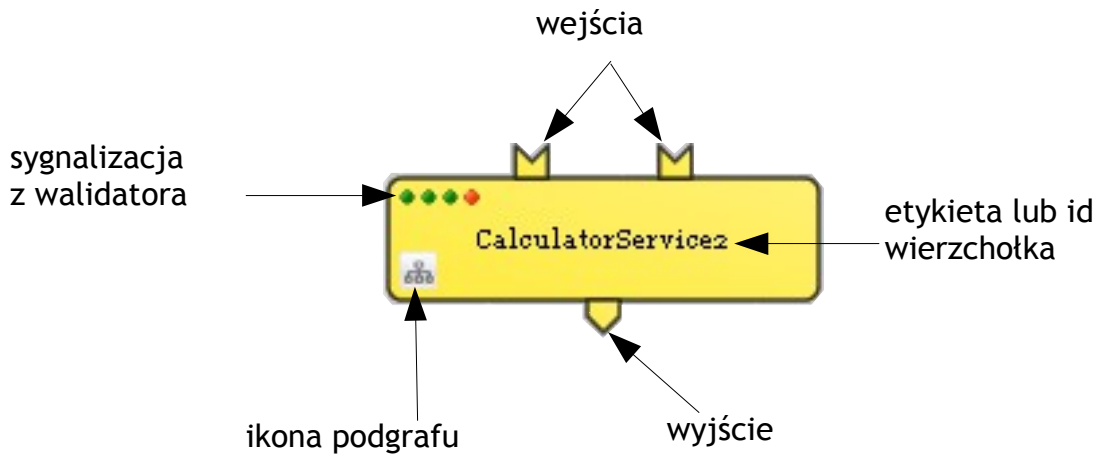
Ryc. 7 - elementy ekranu głównego SSDL_SIG

Po uruchomieniu SSDL_SIG użytkownik może załadować zapisaną w SSDL usługę złożoną w postaci pliku .xml, lub zacząć budować całkiem nową usługę. Po lewej stronie ma do swojej dyspozycji puste wierzchołki grafu, którym może dopisać pożądane parametry po dodaniu ich do kanwy, natomiast po prawej repozytorium usług atomowych, które są w momencie uruchomienia SSDL_SIG pobierane poprzez protokół SOAP.

W trybie edycji procesów biznesowych nieobecne jest repozytorium usług, graf układany jest z innych typów wierzchołków (aktywność ludzka, aktywność biznesowa, zdarzenie), a jedyny dostępny tryb widoku to przepływ kontroli.

Wierzchołek

Graficzna reprezentacja stanowiącej jeden z wierzchołków grafu usługi, w widoku przepływu danych, prezentuje się następująco:



Ryc. 8 - wygląd usługi w SSDL_SIG

Wejścia oraz wyjścia reprezentują interfejs udostępniany przez daną usługę webową. Wyświetlane są one jedynie w widoku przepływu danych, w widoku przepływu kontroli zastępowane są przez punkty, w których występują połączenia pomiędzy wierzchołkami.

Obecność podgrafu w usłudze, zasygnalizowana ikoną w lewym dolnym roku wierzchołka, oznacza, że mamy do czynienia z usługą złożoną. W celu ułatwienia użytkownikowi pracy, w lewym górnym rogu wyświetlane są cztery kontrolki, pełniące rolę walidatora. Oznaczają one, kolejno, obecność zdefiniowanych: wejść, wyjść, słów kluczowych i opisu niefunkcjonalnego.

Inne typy wierzchołków obsługiwane i wyświetlane przez SSDL_SIG to funkcjonalność, mediator, usługa emulująca, oraz wierzchołki kontrolne typów „#start”, „#end”, „#conditionStart”, „#conditionEnd”, które to typy są zgodne z typami zdefiniowanymi w języku SSDL. Różne typy wierzchołków sygnalizowane są różnymi kolorami zaokrąglonego prostokąta oznaczającego wierzchołek, a w przypadku wierzchołków kontrolnych, przybierają różne kształty – koła w przypadku „#start” i „#end”, rombu w przypadku „#conditionStart” i „#conditionEnd”.

Usługa emulująca

Funkcjonalność

Mediator

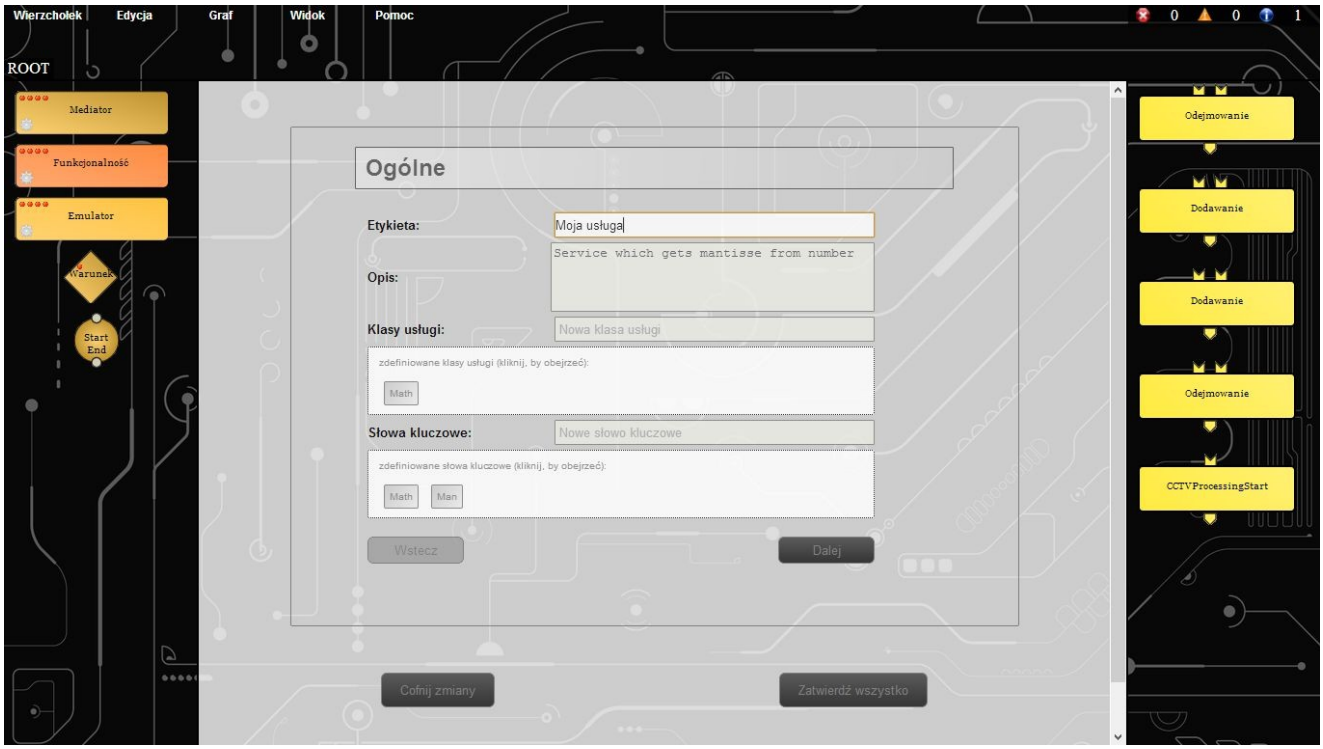
#conditionStart / End

Formularz edycji wierzchołka

Po wybraniu opcji „Edytuj wierzchołek” w oknie kanwy otwarty zostaje formularz, zawierający parametry tego wierzchołka. Mogą one mieć status „do edycji” lub „tylko do odczytu”, zależny od typu wierzchołka oraz od danego parametru.

Warto odnotować, że aplikacja SSDL_SIG nie zawiera predefiniowanych formularzy w HTML. Zamiast tego umożliwia ona napisanie pliku konfiguracyjnego, który definiuje dokładną postać formularza w zależności od typu wierzchołka, i na tej podstawie generuje w zależności od potrzeb kod HTML.

Formularze podzielone są na sekcje, których nazwy i zawartość również zależą od pliku konfiguracyjnego. Nawigacja odbywa się w oparciu o przyciski Dalej / Wstecz, na zasadzie kreatora.



Ryc. 9 - położenie formularza edycji wierzchołka w miejscu zajmowanym standardowo przez kanwę

Formularze pozwalają na dodanie wierzchołkom własnych etykiet, a w zależności od typu również:

- słów kluczowych i klas usługi
- wejść i wyjść
- parametrów/wymagań niefunkcjonalnych

W przypadku wierzchołków, w których dane parametry mają status „tylko do odczytu” formularz umożliwia szczegółowy podgląd np. wejść i wyjść danego wierzchołka.

Ryc. 10 – Główna sekcja formularza edycji wierzchołka typu Usługa.

Na powyższej ilustracji widoczna jest sekcja „Ogólne” formularza edycji / podglądu usługi. Warto zauważyć, że pola „opis”, „słowa kluczowe” i „klasy usługi” nie umożliwiają edycji zawartości, są one dostępne tylko do odczytu, natomiast „etykietę” usługi użytkownik może dowolną liczbę razy zmieniać.

Wykorzystane technologie

Aplikacja składa się z dwóch części:

- klienckiej, napisanej w javaScript
- serwerowej, napisanej w PHP

Dodatkowo wykorzystano:

- Ajax – płynne przekazywanie danych (PHP → javaScript)
- jQuery – ułatwiona modyfikacja elementów HTML
- jQueryUI - autouzupełnianie pól w formularzach
- Raphaël – javaScriptowa biblioteka SVG (rysowanie wierzchołków itd.)
- CSS3 (wygląd interfejsu)
- HTML5