Politechnika Warszawska

WYDZIAŁ ELEKTRONIKI
I TECHNIK INFORMACYJNYCH

Instytut Automatyki i Informatyki Stosowanej

Praca dyplomowa inżynierska

na kierunku Informatyka
w specjalności Systemy informacyjno - decyzyjne

Zestaw modułów analitycznych do środowiska Fiware

Jakub Jan Kochanowski
Numer albumu 277311

promotor
dr inż. Mariusz Kamola

WARSZAWA 2019

STRESZCZENIE

W pracy przedstawiono zestaw modułów do środowiska Fiware przeznaczonych do analizy danych. Opracowano wzorzec, według którego miała odbywać się implementacja. Zaprezentowano dostępne technologie związane z tą tematyką oraz opisano możliwości samego systemu. Zawarto również hipotetyczne scenariusze wykorzystania dostarczonych komponentów. Na koniec dokonano oceny środowiska docelowego Wirecloud.

Słowa kluczowe: Wirecloud, analiza danych, JavaScript, aplikacja webowa

---

SET OF ANALYTICAL MODULES DEDICATED FOR FIWARE ENVIRONMENT

The thesis describes a set of components dedicated for data analysis implemented in Fiware technology. Generic Enabler of Application Mashup API – Wirecloud is in the spotlight. The definition of the output-input communication data format is proposed. A walkthrough of similar solutions is conducted. The reasoning behind chosen modules is derived from hypothetical scenarios portrayed in the paper. At last, evaluation of the target system is performed.

Keywords: Wirecloud, data analysis, JavaScript, web application

…….....
miejscowość i data

……..
imię i nazwisko studenta

……..
numer albumu

…….……
kierunek studiów

OŚWIADCZENIE

Świadomy/-a odpowiedzialności karnej za składanie fałszywych zeznań oświadczam, że niniejsza praca dyplomowa została napisana przeze mnie samodzielnie, pod opieką kierującego pracą dyplomową.

Jednocześnie oświadczam, że:

- niniejsza praca dyplomowa nie narusza praw autorskich w rozumieniu ustawy z dnia 4 lutego 1994 roku o prawie autorskim i prawach pokrewnych (Dz.U. z 2006 r. Nr 90, poz. 631 z późn. zm.) oraz dóbr osobistych chronionych prawem cywilnym,
- niniejsza praca dyplomowa nie zawiera danych i informacji, które uzyskałem/-am w sposób niedozwolony,
- niniejsza praca dyplomowa nie była wcześniej podstawą żadnej innej urzędowej procedury związanej z nadawaniem dyplomów lub tytułów zawodowych,
- wszystkie informacje umieszczone w niniejszej pracy, uzyskane ze źródeł pisanych i elektronicznych, zostały udokumentowane w wykazie literatury odpowiednimi odnośnikami,
- znam regulacje prawne Politechniki Warszawskiej w sprawie zarządzania prawami autorskimi i prawami pokrewnymi, prawami własności przemysłowej oraz zasadami komercjalizacji.

Oświadczam, że treść pracy dyplomowej w wersji drukowanej, treść pracy dyplomowej zawartej na nośniku elektronicznym (płycie kompaktowej) oraz treść pracy dyplomowej w module APD systemu USOS są identyczne.

.....
czytelny podpis studenta

Spis treści

1. Wstęp

Obecnie prawidłowa obróbka danych stała się istotnym elementem sukcesu wielu szanowanych koncernów i instytucji. Mądra analiza otrzymywanych informacji decyduje o przedsięwzięciu właściwych kroków prowadzących do realizacji wyznaczonych sobie celów. Z powodu zysków jakie można osiągnąć poprzez prawidłowe zinterpretowanie danych, duże firmy stały się chętne do wydawania niemałych środków na narzędzia umożliwiające ich trafną eksplorację i wspomagające podejmowanie odpowiednich decyzji. Ogromne zainteresowanie spowodowało, że cena, jak i złożoność takiego oprogramowania rosły. Małe, często jednoosobowe biznesy nie posiadają funduszy na zakupienie takiego produktu, nie widzą plusów idących z analizy zbiorów danych lub próg wejścia związany z nauką takiego narzędzia jest dla nich za duży. Jednak, jak pisze Gregory Thompson na łamach witryny Data Science Central, w dzisiejszych czasach nie istnieje wymówka dla żadnej firmy, niezależnie od wielkości czy posiadanych środków, aby pomijać analizę danych w codziennych działaniach biznesowych [17]. Darmowego oraz łatwego w użyciu oprogramowania analitycznego jest dość niewiele, co było jedną z motywacji do stworzenia poniższej pracy dyplomowej.

1.1. Cel pracy dyplomowej

Celem projektu było stworzenie zestawu komponentów analitycznych charakteryzujących się modularnością działania. Środowisko implementacyjne *Fiware*, a dokładniej jedna z jego części – *Wirecloud,* zostało narzucone z góry. Ilość modułów i ich funkcjonalność była z założenia ograniczona. Zestaw programów został dobrany tak, aby istniała możliwość realizacji zadań w dwóch hipotetycznych scenariuszach użycia, korzystających z dwóch różnych zbiorów danych.

Kolejnym głównym celem pracy dyplomowej było zdefiniowanie uniwersalnego wzorca formatu danych wykorzystywanego do tworzenia tego typu oprogramowania. Standard ten miał umożliwiać jak największej liczbie modułów komunikację ze sobą. Definiowałby on zachowania kolejnych pozycji dodawanych do zbioru już istniejących komponentów.

1.2. Zawartość pracy

Na początku pracy zostały przedstawione istniejące na rynku podobne rozwiązania. Główny nacisk został położony na programach, które są darmowe oraz posiadają mechanizmy potokowego przetwarzania danych.

W rozdziale pt. „Środowisko implementacyjne” opisano platformę Wirecloud, na której stworzono moduły analityczne. W rozdziale tym omówiono również użycie tej platformy.

„Wzorzec formatu wejść/wyjść” zawiera opis koncepcyjny pracy nad stworzeniem uniwersalnego schematu wymiany danych pomiędzy elementami.

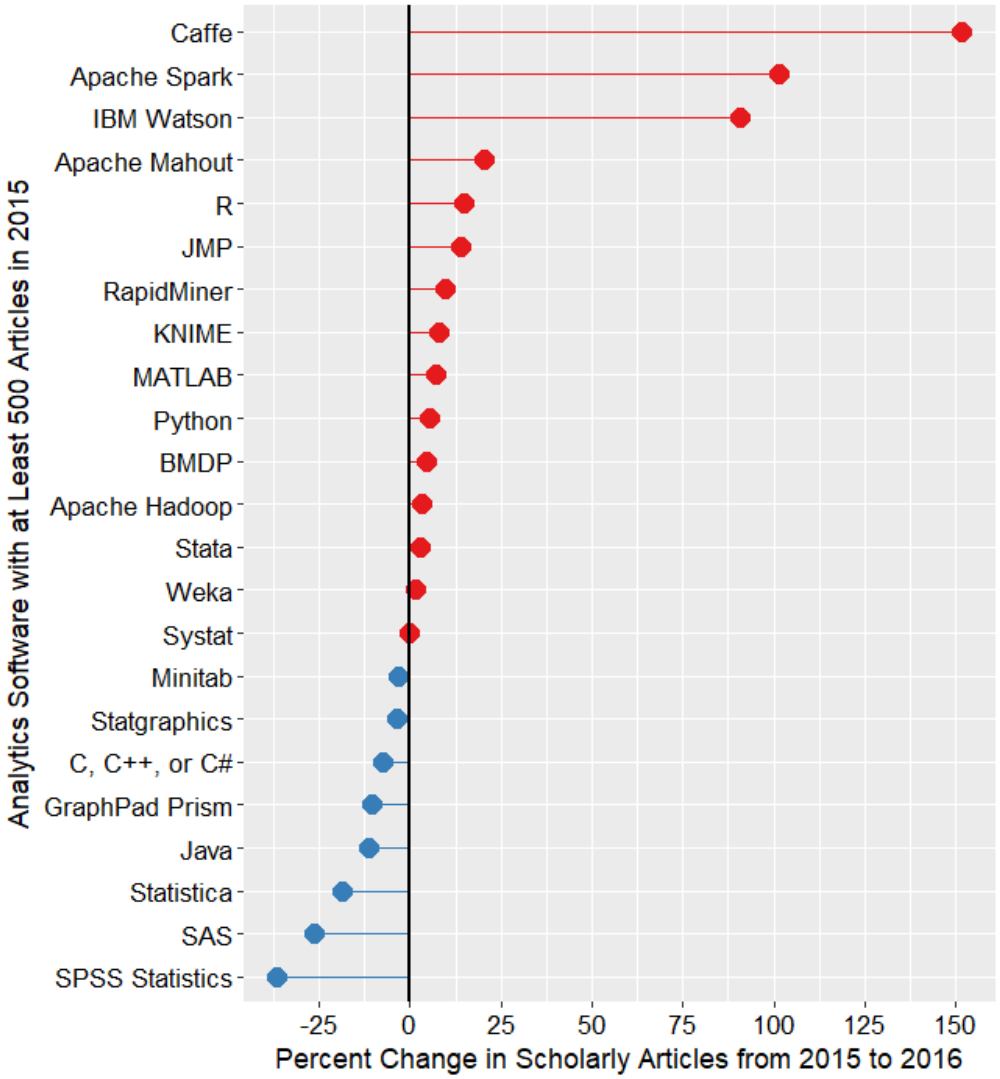
W rozdziale szóstym zaprezentowano zaimplementowane oprogramowanie. Zostało ono podzielone na podgrupy w zależności od charakteru ich działania.

W rozdziale siódmym zostały przedstawione hipotetyczne scenariusze, na podstawie których wybrano jakie funkcjonalności należy stworzyć w ramach dostarczanych modułów.

W rozdziałach „Możliwości rozwoju” oraz „Podsumowanie” autor przeprowadził analizę możliwych dróg rozwoju wykonanego projektu oraz dokonał podsumowania swojej pracy.

2. Istniejące rozwiązania

Rynek oprogramowania do analizy danych jest rynkiem rozwijającym się. Na rok 2015 najbardziej popularnym oprogramowaniem analitycznym było *R, SAS i SPSS Statistics*[1]. *SAS* oraz *SPSS Statisctics* są to rozwiązania płatne co wyklucza je z dalszej analizy, a środowisko *R* jest zbyt skomplikowane dla przeciętnego użytkownika. Dlatego w tym rozdziale wybrano pozycje, które najlepiej spełniają wymagania oprogramowania darmowego i łatwego w użyciu. Sa to programy *KNIME* oraz *RapidMiner*



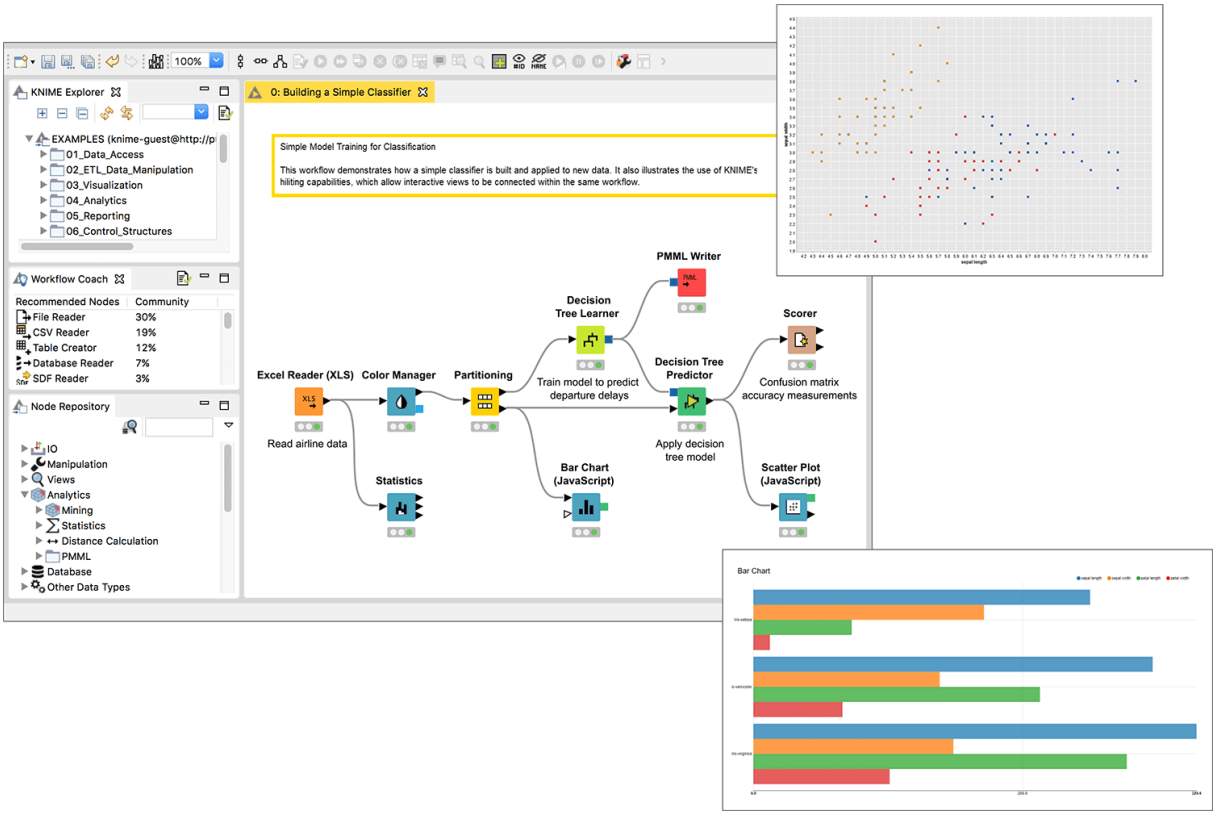
Rys. 1 Procentowa zmiana w ilości artykułów naukowych związanych z danym oprogramowaniem w latach 2015 i 2016.
Źródło: https://www.knime.com/knime-software/knime-analytics-platform

Oba rozwiązania zanotowały wzrost zainteresowania w latach 2015 i 2016 co pokazuje Rys. 1, a w 2018 roku firma Gartner, w swoim corocznym raporcie, umieściła zarówno *KNIME* jak i *RapidMiner* w kategorii „Liderzy” [15].

[1] Źródło informacji: https://r4stats.com/articles/popularity/

2.1. KNIME

KNIME (ang. Konstanz information Miner) jest to darmowe narzędzie o otwartym kodzie źródłowym spełniające paradygmat potokowego przetwarzania danych. Posługiwanie się programem polega na wykorzystywaniu modularnego środowiska i tworzeniu grafu, którego węzłami są odpowiednie moduły przeprowadzające konkretne działania na danych, a krawędziami strumienie przepływu danych (tak jak to przedstawia Rys. 2).



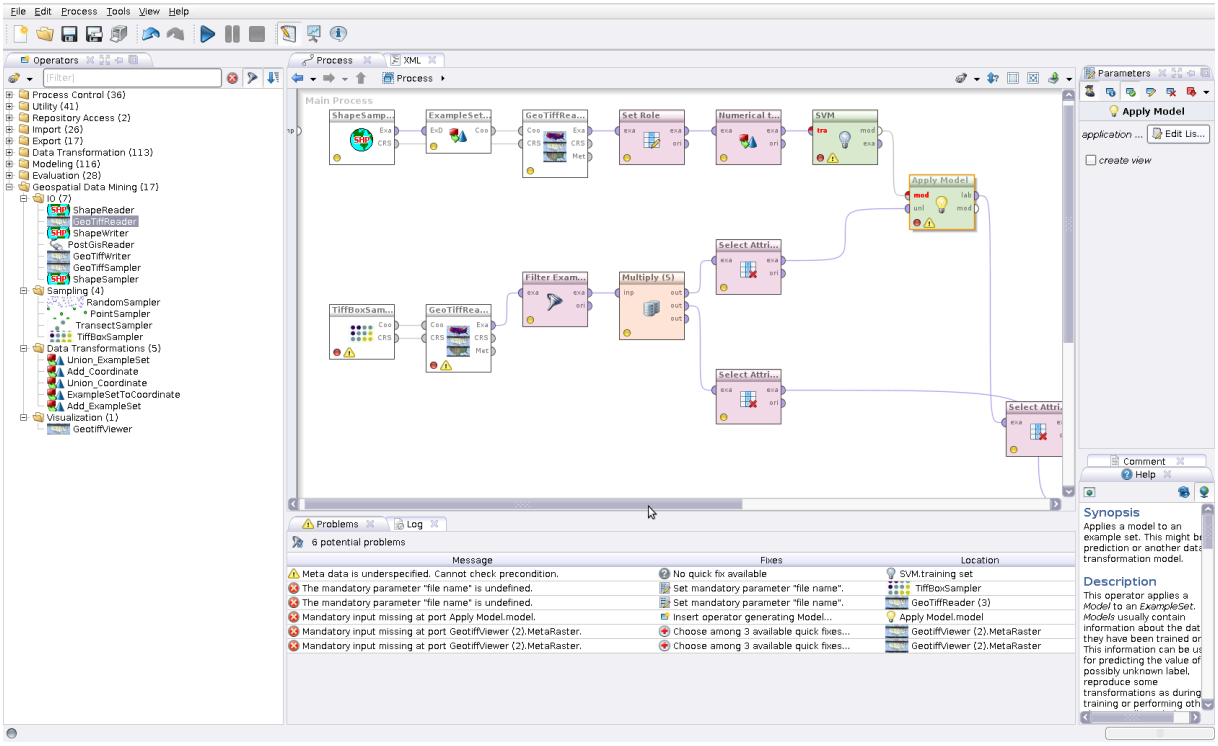
Rys. 2 Interfejs graficzny programu KNIME przedstawiający przykładowe zestawienie modułów. Źródło: https://www.knime.com/knime-software/knime-analytics-platform

Dodawanie kolejnych węzłów grafu polega na metodzie przeciągnij i upuść. Następnie użytkownik może połączyć wybrane węzły w logiczną całość. Architektura *KNIME* została zaprojektowana z myślą o trzech, głównych wymaganiach:

- Interaktywny interfejs umożliwiający tworzenie sprofilowanych aplikacji przy pomocy metody przeciągnij i upuść.
- Modularność, czyli niezależność formatów danych poszczególnych modułów. Możliwość użycia różnych węzłów w różnych kombinacjach.
- Łatwa rozszerzalność, czyli możliwość dodawania nowych modułów bez potrzeby trudnej procedury instalacji [13].

2.2. RapidMiner

RapidMiner jest to oprogramowanie, które posiada wersję darmową jak i wersję komercyjną. Środowisko to oferuje podobny interfejs graficzny co środowisko *KNIME*. Stworzenie autorskiego przepływu polega na wybraniu z menu odpowiednich operatorów i połączeniu ich ze sobą w graf reprezentujący potok przetwarzania danych. Rys. 3 przedstawia przykładowe zestawienie operatorów. Po lewej stronie interfejsu widać menu z listą dostępnych operatorów do wybrania. W centrum widnieje przestrzeń robocza, w której możemy wcześniej wybrane moduły zestawiać ze sobą.



Rys. 3 Interfejs programu RapidMiner z przykładem wykorzystania.
Źródło: http://geoimagermp.gforge.inria.fr/

Narzędzia *RapidMiner* oraz *KNIME* są podobne, jako, że oba oferują potokowy, łatwy w użyciu interfejs. Każde z nich udostępnia darmowe wersje o otwartym kodzie źródłowym [14].

Środowisko *RapidMiner* stale się rozwija i jego główną zaletą nad oprogramowaniem *KNIME* jest zrzeszona społeczność oferująca pomoc w przypadku jakichkolwiek problemów [12].

Programy wymienione wyżej działają jako aplikacje desktopowe. Opisane w tej pracy rozwiązanie zakłada stworzenie aplikacji webowej co odróżniałoby go od typowych rozwiązań biznesowych skupiających się na modularności działania. Takie podejście pozwala na korzystanie z oprogramowania bez potrzeby instalacji go na lokalnej przestrzeni dyskowej.

3. Środowisko implementacyjne

W rozdziale tym zostanie przedstawiona platforma Fiware jak i jedno z jej otwartych interfejsów programowania aplikacji - Wirecloud, na którym tytułowy zestaw modułów do analizy danych został stworzony. Zostaną również przedstawione zalety jak i wady tego oprogramowania z perspektywy tworzenia na nie narzędzi do analizy danych.

3.1. Fiware

Fiware jest to inicjatywa współfinansowana przez Unię Europejską, której założeniem jest stworzenie rozwiązań o otwartych źródłach skupiających się na Internecie Przyszłości. Platforma ta oferuje bogate biblioteki skupiające się na Internecie Rzeczy, analizie zawartości multimedialnych w czasie rzeczywistym oraz obróbce dużych zbiorów danych. Jej otwarte specyfikacje ułatwiają również tworzenie zaawansowanych interfejsów graficznych bazujących na rozwiązaniach przeglądarkowych. Platforma Fiware udostępnia swoisty katalog, w którym przedstawia implementacje każdego z API w jej ofercie. Programy znajdujące się w katalogu oferują otwarty kod źródłowy i są całkowicie darmowe. Założeniem twórców idei Fiware była modularność ich interfejsów programowania aplikacji tak, aby deweloperzy mogli wykorzystywać tylko te części, które są im potrzebne. W katalogu Fiware znajdują się także implementacje konkretnych API stworzone przez zewnętrzne firmy. Również te produkty posiadają otwarty kod źródłowy i są całkowicie darmowe.

3.2. Wirecloud

Wirecloud jest implementacją jednego z otwartych API oferowanych przez platformę Fiware. Aplikacja ta jest stworzona w oparciu o API o nazwie Application Mashup. Głównym założeniem tego interfejsu programowania aplikacji jest stworzenie środowiska umożliwiającego zestawianie z mniejszych modułów, dużych aplikacji webowych. Głównymi komponentami wykorzystywanymi przez system Wirecloud są:

- Widget: widgety w środowisku Wirecloud są to moduły, które posiadają swoją reprezentację graficzną i są wyświetlane w trybie Dashboard
- Operator: operatory w środowisku Wirecloud są modułami, które są jedynie dostępne z trybu łączenia przestrzeni roboczej. Nie posiadają swojej reprezentacji graficznej w trybie wyświetlania. Mają skupiać się na pozyskiwaniu, przetwarzaniu i dostosowywaniu danych tak, aby widgety mogły z nich korzystać.
- Zestawienia (ang. Mashup): zestawienia w środowisku Wirecloud są to gotowe struktury logiczne stworzone z połączonych już ze sobą modułów służące często jako szablony do tworzenia nowych zestawień.

Łatwość łączenia i korzystania z modułów umożliwia użytkownikom posiadającym minimalną wiedzę programistyczną do pracowania w tym środowisku bez większych przeszkód. Tworzenie modułów odbywa się przy użyciu standardowych technologii webowych tj. *JavaScript*, *HTML*, *CSS*, *PHP* itp. [1].

3.2.1. Uruchomienie Wirecloud

W tej pracy skorzystano z instancji Wirecloud uruchomionej na środowisku Fiware Lab[2]. Jest to środowisko udostępniane przez Fiware do uruchomiania oraz testowania technologii opartych o specyfikacje tej platformy. Istnieje również możliwość skonfigurowania instancji

[2] Link do środowiska Fiware Lab: https://mashup.lab.fiware.org/wirecloud/home

Wirecloud na lokalnej maszynie ale w tej pracy istotna była możliwość ciągłego dostępu do środowiska co wykluczało takie rozwiązanie bez posiadania odpowiedniego serwera. Mimo wszystko takowa instancja została zainstalowana na maszynie lokalnej autora, nie w celu bycia instancją docelową, a w celu sprawdzenia czy dokumentacja opisująca przebieg instalacji jest poprawna[3].

3.2.2. Konta użytkownika

Konta dostępne w witrynie Fiware Lab dzielą się na dwa typy: próbne konto (ang. Trial Account) oraz konto społeczności (ang. Community Account). Dostęp do obydwu typów kont jest darmowy, jednak konta społeczności są przydzielane przez administratorów poszczególnych węzłów środowiska Fiware Lab [7]. W Polsce węzły te są w Poznaniu i Wrocławiu[4]. Uzyskanie konta społeczności umożliwia otrzymanie zasobów dyskowych i obliczeniowych, które znajdują się na wybranym węźle. Na potrzeby tej pracy dyplomowej została przeprowadzona próba otrzymania takowego konta jednak zakończyła się ona porażką. Wszystkie działania odbywają się na koncie próbnym.

3.2.3. Instalowanie modułów

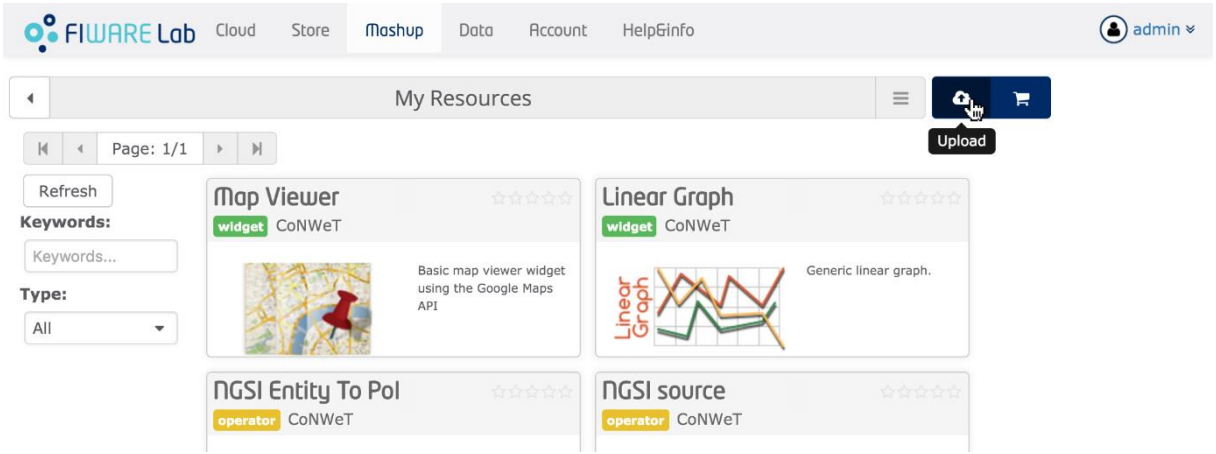
Autorzy oprogramowania postawili na łatwość umieszczania nowych modułów w środowisku. Aby to uczynić należy zalogować się na witrynie Fiware Lab oraz wybrać zakładkę Mashup tak jak na Rys. 4.

Rys. 4 Menu witryny Fiware Lab.
Źródło: Własne

Następnym krokiem jest przejście do zasobów przypisanych do konta i załadowanie na serwer pliku archiwum (.zip) zawierającego komponent. Moduły powinny posiadać odpowiednią strukturę wewnętrzną, która została opisana w rozdziale 3.2.4. W innym przypadku serwer poinformuje użytkownika o nieprawidłowej strukturze plików.

[3] Dokumentacja dostępna pod linkiem: https://wirecloud.readthedocs.io/en/stable/installation_guide/

[4] Mapa węzłów Fiware Lab dostępna pod adresem: http://infographic.lab.fiware.org/



Rys. 5 Ładowanie modułu do środowiska Wirecloud.
Źródło: Własne

3.2.4. Wewnętrzna struktura modułów

Moduły, które środowisko Wirecloud jest wstanie prawidłowo zinterpretować, dzielą się na operatory oraz widgety. Obydwa typy powinny posiadać odpowiednią strukturę wewnętrzną. Po pierwsze, do stworzenia tychże modułów powinny zostać wykorzystane technologie akceptowalne przez przeglądarki tj. *XHTML*, *JavaScript*, *Flash*, aplety itp. Po drugie w skład plików budujących dany moduł musi wchodzić plik konfiguracyjny *config.xml,* który zawiera opis wejść, wyjść oraz opcji komponentu. Powinien być on opisany według odpowiedniego schematu XML zdefiniowanego przez twórców Wirecloud (MACDL). W przypadku widgetów, plik ten powinien odsyłać do pliku *index.html*, a w przypadku operatorów do plików z kodem języka JavaScript. Jest tak ponieważ z założenia widgety są elementami ekranowymi, a operatory skupiają się na operowaniu na danych i nie posiadają reprezentacji graficznej.

Poniżej przedstawiona jest przykładowa struktura wewnętrzna modułu (Rys. 6).

```
.
+-- docs
|   |   ...
|   +-- index.md
+-- css
|   |   ...
|   +-- style.css
+-- images
|   |   ...
|   +-- catalogue.png
+-- js
|   |   ...
|   +-- main.js
+-- CHANGELOG.md
+-- config.xml
+-- index.html
+-- DESCRIPTION.md
```

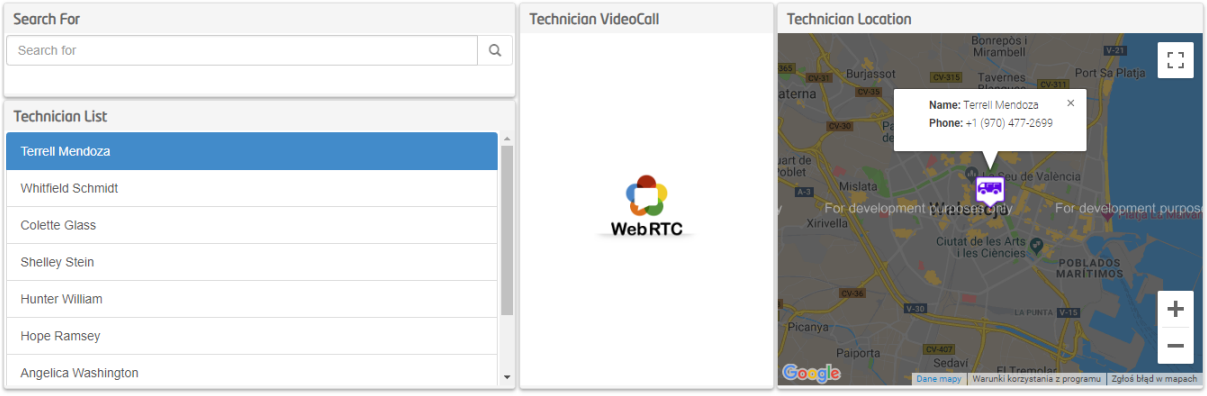
Rys. 6 Przykładowa struktura plików widgetu.
Źródło: https://wirecloud.readthedocs.io/en/latest/development/widget_and_operators/

3.2.5. Przestrzeń robocza (ang. workspace)

Przestrzenie robocze w środowisku Wirecloud oferują możliwość zestawiania ze sobą widgetów i operatorów w aplikację. Każda taka przestrzeń posiada dwa tryby edycji:

- Tryb wyświetlania (ang. Dashboard): jest to tryb, w którym reprezentowane są jedynie elementy ekranowe. Jest to główny tryb przestrzeni roboczej, w którym użytkownik spędzi większość czasu. Istnieje możliwość podziału tego widoku na zakładki, co ułatwia grupowanie widgetów w grupy tematyczne. Przykładowe zestawienie modułów w tym trybie przestawia Rys. 7.
- Tryb łączenia: jest to tryb, w którym użytkownik może stworzyć logiczną strukturę połączeń pomiędzy operatorami i widgetami tak, aby spełniała jego wymagania. Istnieje również możliwość edycji ustawień poszczególnych modułów. Do tego widoku dostęp posiada jedynie właściciel przestrzeni roboczej. Przykładowe zestawienie modułów w tym trybie przedstawia Rys. 8.

Wirecloud oferuje możliwość udostępniania stworzonych przestrzeni innym użytkownikom. W takim przypadku udostępniany jest jedynie widok Dashboard.

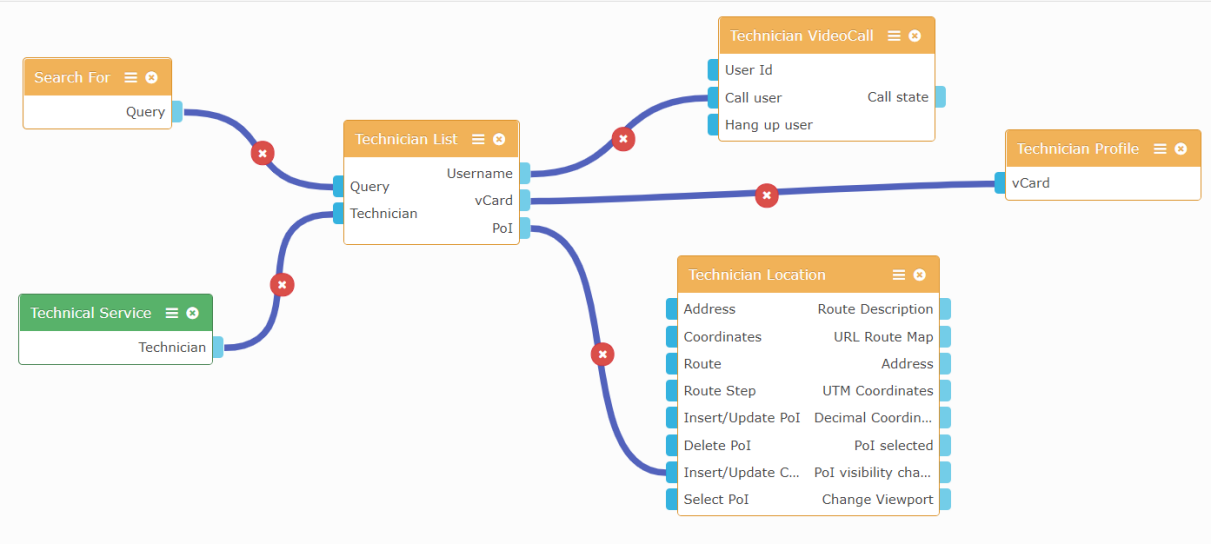


Rys. 7 Przykładowe zestawienie modułów w trybie wyświetlania.
Źródło: Własne

3.2.6. Zalety środowiska

Środowisko Wirecloud zostało stworzone z myślą o użytkownikach. Jego główną zaletą jest łatwość korzystania z przestrzeni roboczej. System łączenia jest bardzo intuicyjny i czytelny. Możliwe do wykorzystania moduły są przypisane do konta co pozwala na logowanie się do

środowiska z wielu różnych urządzeń i posiadaniu takich samych zasobów. Możliwość udostępniania swojej pracy innym użytkownikom jest również jedną z istotnych zalet.



Rys. 8 Przykładowe zestawienie modułów w trybie łączenia.
Źródło: Własne

3.2.7. Wady środowiska

W trakcie obcowania ze środowiskiem podczas tworzenia tej pracy dyplomowej zauważono dużo niedociągnięć i uchybień ze strony autorów oprogramowani. Innym powodem dlaczego platforma Wirecloud nie jest idealna jest malejące zainteresowanie ze strony samej organizacji Fiware. Głównymi problemami na jakie natknięto się podczas pisania pracy były:

- Brak społeczności: oprogramowanie Wirecloud jest na tyle nowe i na tyle mało popularne, że właściwie nie istnieje żadne wsparcie od strony społeczności. Twórcy tego środowiska proszą o zadawanie pytań na popularnej witrynie *StackOverflow* skupiającej się na problemach programistycznych przy użyciu tagu *„fiware-wirecloud”* [1]. W ciągu pięciu lat zostało dokonanych 145 zapytań z użyciem tego tagu z czego 52 pytania pozostały bez satysfakcjonującej odpowiedzi (35.9% pytań bez odpowiedzi)[5]. Te statystyki pokazują jak małą pomoc społeczności może otrzymać deweloper. Również firma Fiware, która prowadzi system oceniania poszczególnych aspektów implementacji ich API[6] oceniła parametry czasu reakcji i czasu naprawy (ang. Responsiveness) systemu Wirecloud na najgorszą w skali – ocenę „D”.
- Niekompletna dokumentacja: dokumentacja systemu Wirecloud jest dosyć czytelna jednak niekompletna. W celu przekazania informacji o tym systemie użytkownikom została stworzona grupa samouczków. Podczas opracowywania tej pracy dyplomowej zawartość samouczków nie przyniosła spodziewanych informacji. Wirecloud otrzymało ocenę „A” za kompletność dokumentacji oraz ocenę „A+” za jej czytelność (Rys. 9).

[5] Dane pozyskane ze strony https://stackoverflow.com/tags/fiware-wirecloud/

[6] System oceny prowadzony przez organizację Fiware dostępny pod linkiem: https://www.fiware.org/wp-content/uploads/2016/10/QA_public_document.pdf

- Problemy techniczne: podczas tworzenia pracy dyplomowej natrafiono na kilka problemów technicznych:
    1. Ograniczona funkcjonalność wtyczki Wirecloud do Eclipse IDE: w dokumentacji środowiska Wirecloud można znaleźć instrukcję opisującą przeprowadzenie instalacji wtyczki. Według opisu, jedną z zalet instalacji wtyczki jest możliwość połączenia się z instancją systemu Wirecloud co umożliwia umieszczanie modułów w systemie bez potrzeby korzystania ze standardowej procedury ładowania modułów przez przeglądarkę [1]. Po odnalezieniu pliku z wtyczką udało się przeprowadzić instalację wedle instrukcji i zestawić połączenie z instancją Wirecloud umieszczoną na witrynie Fiware Lab. Niestety połączenie to po jakimś czasie się przerywało. Odbyła się nieudana próba skontaktowania się z jednym z twórców systemu drogą mailową w celu zidentyfikowania próbnemu.
    2. Błąd krytyczny przeglądarki: podczas testowania działania stworzonych, w trakcie przygotowywania pracy, komponentów zauważono, że przy próbie przesyłania zbyt dużych danych pomiędzy modułami system Wirecloud powodował błąd krytyczny przeglądarki. Zostało stworzone zapytanie z odpowiednim oznaczeniem na witrynie *StackOverflow* o możliwe przyczyny występowania problemu jednak nie udało się otrzymać odpowiedzi.

criteria	value	label
Documentation completeness	Good	qa A
Documentation soundness	Quite Good	qa A+
APIs Failure Rate	0.12 tests failed/executed	qa A++
Time to respond issues	< 15.74 days	qa D
Time to fix issues	< 24.78 days	qa D
Scalability	1,22 response time/thread number	qa A+
Performance	1,11 res/sec/user	qa A+
Stability	Low memory leaks detected	qa A++

Rys. 9 Ocena systemu Wirecloud przy użyciu systemu oceniania organizacji Fiware.
Źródło: https://catalogue-server.fiware.org/enablers/application-mashup-wirecloud/documentation

4. Wykorzystane technologie

W rozdziale tym zostaną przedstawione technologie, z których korzystano przy produkcji tytułowych modułów analitycznych. Pokazano tutaj zarówno elementy wspomagające programistę w pracy, wykorzystane języki programowania jak i niezbędne biblioteki użyte do stworzenia docelowego oprogramowania.

4.1. Formaty danych

W tym podrozdziale zostaną przedstawione sposoby opisu danych, z którymi obcowano podczas tworzenia poszczególnych elementów.

4.1.1. CSV

Format *CSV* (ang. Comma-Separated Values) jest wykorzystywany do wymiany i konwersji danych tabelarycznych. Jest to format danych przetrzymywanych w formie tekstowej. Mimo braku ogólnie przyjętego standardu większość implementacji tego formatu zakłada, że:

- Każdy wiersz oddzielony jest znakiem końca linii (ostatni wiersz nie musi spełniać tego wymagania).
- Każde pole w wierszu musi być rozdzielone przecinkiem (lub innym rodzajem separatora). Po ostatnim polu w wierszu nie może być separatora.
- Każdy wiersz musi posiadać tyle samo pól.
- Opcjonalnie może istnieć wiersz nagłówka w pierwszej linii pliku. Powinien on posiadać tyle samo pól co reszta wierszy. Zawiera on nazwy kolumn odnoszące się do odpowiednich pól w pliku [16].

4.1.2. JSON

JSON (ang. JavaScript Object Notation) jest formatem tekstowym do serializacji ustrukturyzowanych danych. Wywodzi się on z języka *JavaScript*. Głównymi założeniami przy projektowaniu tego opisu danych były: prostota, przenośność, reprezentacja w postaci tekstowej oraz bycie podzbiorem języka *JavaScript*. Format *JSON* jest w stanie opisać cztery podstawowe (łańcuchy znaków, liczby, wartości logiczne i znak Null) i dwa ustrukturyzowane (tablice i obiekty języka JavaScript) typy danych [2]. Poniżej przedstawiono przykładowy obiekt opisany notacją *JSON*.

```
1. {
2.     "imie":"Jakub",
3.     "nazwisko":"Kochanowski",
4.     "wiek":21
5. }
```

4.1.3. JSON Schema

JSON Schema jest próbą dostarczenia czytelnego i zrozumiałego dla człowieka języka schematów opisującego struktury *JSON*. Głównymi funkcjonalnościami oferowanymi przez język *JSON Schema* jest możliwość nałożenia ograniczeń na obiekty opisane formatem *JSON* oraz dostarczenie narzędzi do weryfikacji tych obiektów [6]. Poniżej przedstawiono przykładowy schemat opisany językiem *JSON Schema*, który pozytywnie weryfikuje strukturę przedstawioną w poprzednim podrozdziale.

```
{
    "&schema": "http://json-schema.org/draft-07/schema#",
    "type":"object",
    "required": [
        "imie",
        "nazwisko",
        "wiek"
    ],
    "properties": {
        "imie": {
        "type": "string"
        },
        "nazwisko": {
        "type": "string"
        },
        "wiek": {
        "type": "integer",
        "minimum": 0
        }
    }
}
```

Jak widać struktury opisane językiem *JSON Schema* są także obiektami typu *JSON*. To sprawia, że schematy można porównać do innego istniejącego wzorca. W przykładzie powyżej własność „`&schema`" mówi o tym, wedle jakiego standardu *JSON Schema* będzie sprawdzany schemat. Jak mówi oficjalna dokumentacja formatu *JSON Schema*, najnowszym wydaniem jest standard w wersji siódmej [11].

4.2. Technologie

Poniżej zostały przedstawione języki programowania wykorzystane w implementacji tego projektu.

4.2.1. JavaScript

JavaScript jest interpretowanym językiem programowania wysokiego poziomu z możliwościami wykorzystania rozwiązań obiektowych w kodzie źródłowym. Język ten, jak pisze Dawid Flanagan w swojej książce, jest obecny w znakomitej większości wszystkich współczesnych przeglądarek – to sprawia, że jest on najczęściej występującym językiem programowania w historii [8].

W tej pracy dyplomowej *JavaScript* będzie najczęściej stosowaną technologią ponieważ wymaga tego charakterystyka środowiska Wirecloud.

4.2.2. HTML

HTML (ang. HyperText Markup Language) jest językiem opisu danych wykorzystywanym w sieci WWW do publikowania informacji [5]. Definiuje on wygląd stron internetowych jak również ich zawartość.

4.2.3. CSS

CSS (ang. *Cascading Style Sheets*) jest językiem wykorzystywanym do opisu stylów wykorzystywanych na stronach *HTML*. Opisuje on miedzy innymi czcionki, kolory, tła i wiele innych obiektów [16].

4.2.4. XML

XML (ang. *Extensive Markup Language*) jest językiem znaczników opisującym klasę obiektów danych zwanych dokumentami XML. Jest on podzbiorem języka *SGML* (ang. *Standard Generalized Markup Language*). Dokumenty opisane przy użycia *XML* definiują zachowanie programów komputerowych, do których się odnoszą [3]. W tym projekcie język *XML* jest wykorzystywany w plikach `config.xml` opisujących główną strukturę modułów.

Języki *HTML*, *CSS* i *JavaScript* tworzą triadę technologii, które każdy deweloper rozwiązań Webowych powinien znać [8].

4.3. Biblioteki

Poniżej znajduje się opis bibliotek wykorzystanych w projekcie wraz z argumentacją ich wyboru.

4.3.1. Highcharts

Highcharts jest biblioteką napisaną w języku JavaScript do tworzenia interaktywnych wykresów[7]. Bazuję na technologii *SVG* (ang. *Scalable Vector Graphics*) co ułatwia skalowanie stworzonych obrazów. Wedle twórców biblioteka ta jest wykorzystywana przez ponad 80% największych, światowych firm w skład których wchodzą między innymi: Facebook, Microsoft czy Mastercard [9]. Właśnie z powodu jej popularności została wybrana do stworzenia tytułowego zestawu modułów do analizy danych.

4.3.2. Papaparse

Papaparse[8] jest biblioteką stworzoną z myślą o konwersji plików *CSV* do formatu *JSON*. Obydwa formaty zostały zdefiniowane wcześniej. Biblioteka ta została w całości napisana w języku *JavaScript* i jest często stosowana w rozwiązaniach webowych. Na platformie *jsPerf*[9], która skupia się na testowaniu różnego rodzaju narzędzi napisanych w języku JavaScript, właśnie *Papaparse* w wersji 4. wypadł najlepiej w porównaniu do czterech innych konwerterów. Między innymi szybkość i skuteczność zdecydowały o wybraniu tego narzędzia do wykorzystania w tej pracy dyplomowej.

Rezultatem konwersji pliku *CSV* jest obiekt *JSON* o poniższej strukturze [10]:

```
{
    data:   ,       // tablica przekonwertowanych wierszy
    errors: ,       // tablica z błędami
    meta:           // obiekt zawierający dodatkowe informacje
}
```

Struktura ta stała się inspiracją do zdefiniowania formatu wejść i wyjść modułów analitycznych stworzonych na środowisko Wirecloud. Został on opisany w rozdziale „Wzorzec formatu wejść/wyjść”.

[7] Oficjalna strona Highcharts: https://www.highcharts.com/

[8] Oficjalna strona konwertera Papaparse znajduje się pod adresem: https://www.papaparse.com/

[9] Platforma *jsperf* znajduje się pod linkiem: https://jsperf.com/

4.3.3. Everpolate

Biblioteka wykorzystywana do wyliczania współczynnika korelacji Pearsona oraz do przeprowadzenia operacji usuwania trendu z danej kolumny. Jest to proste narzędzie programistyczne udostępnione przez twórców do użytku publicznego. Przeznaczona jest do połączenia z językiem *JavaScript* dzięki czemu nadaje się do zastosowania w przeglądarkach webowych [4].

4.3.4. Ajv

Ajv (ang. Another JSON-Schema Validator) jest oprogramowaniem napisanym w języku JavaScript umożliwiającym walidacje struktur *JSON* na podstawie schematu opisanego przy użyciu standardu *JSON Schema*. Biblioteka ta jest możliwa do wykorzystania w środowiskach przeglądarek internetowych. Wspiera ona najnowsze wydanie standardu *JSON Schema* w wersji 7. Według najnowszych testów[10] Ajv jest drugim co do szybkości działania narzędziem przeznaczonym do walidacji struktur *JSON*. Pierwsze miejsce przypadło oprogramowaniu Djv (ang. Dynamic JSON-Schema Validator) jednak nie wspiera ono najnowszego standardu w wersji 7 co było powodem odrzucenia go przy wyborze tego typu oprogramowania. Prędkość działania jak i stałe wsparcie oprogramowania Ajv zadecydowało o jego zastosowaniu w tym projekcie.

4.4. Zintegrowane środowisko programistyczne (IDE)

W tej części dokumentu zostanie opisane wybrane środowisko programistyczne oraz wszelkie dodatki do niego dołączone.

4.4.1. Eclipse

Jako IDE zostało wybrane oprogramowanie *Eclipse* przeznaczone dla deweloperów rozwiązań webowych. Według ocen użytkowników[11], nie jest to najlepsze rozwiązanie do tego typu projektów jednak zdecydowano się z niego skorzystać z powodu istnienia dedykowanej wtyczki do środowiska Wirecloud. Dzięki temu programista uzyskuje dostęp do wzorców projektowych komponentów, importowania projektów operatorów lub widgetów oraz możliwości połączenia się z zewnętrzną instancją środowiska Wirecloud [1].

[10] Wyniki testów dostępne pod adresem: https://github.com/ebdrup/json-schema-benchmark

[11] Dane pozyskane z witryny: https://www.slant.co/topics/8150/~ides-for-web-development

5. Wzorzec formatu wejść/wyjść

Modularność komponentów była głównym założeniem projektu. Jednym z podstawowych problemów inżynieryjnych związanych z tym założeniem było zdefiniowanie zarówno prostego, jak i uniwersalnego standardu wejść i wyjść modułów tak, aby większość z nich była ze sobą kompatybilna.

Po przeprowadzeniu poznawczej analizy problemu, zauważono potrzebę reprezentowania zarówno danych tabelarycznych jak i metadanych opisujących ich strukturę w formie obiektów w formacie *JSON*. Decyzja taka była wypadkową wielu czynników:

- Główne funkcjonalności modułów są opisane językiem *JavaScript*, który posiada odpowiednie wsparcie dla struktur *JSON*.
- Architektura środowiska Wirecloud wymaga, aby komunikaty przesyłane pomiędzy modułami były w formie tekstowej. Takie wymaganie spełnia format *JSON*.
- Format *JSON* jest popularny w rozwiązaniach webowych.

Przesyłanymi strukturami będą obiekty języka *JavaScript* posiadające dwie główne własności: własność `data` oraz własność `meta`. Poniżej przedstawiono przykładowy obiekt poruszający się pomiędzy modułami.

```
{
    data : [{
    imie:"Jakub",
    nazwisko:"Kochanowski",
    wiek:21},{
    imie:"Anna",
    nazwisko:"Galanty",
    wiek:21
        }
    ],
    meta : {
    fields:["imie","nazwisko","wiek"]
    }
}
```

Jak widać na powyższym przykładzie, własność `data` jest tablicą obiektów, a własność `meta` jest obiektem zawierającym tablice nazw kolumn. W przypadku, gdy dane tabelaryczne nie posiadają nagłówka, własność `data` jest tablicą tablic reprezentujących poszczególne wiersze, a własność `meta` jest pusta. Taki przypadek przedstawiono poniżej.

```
{
    data : [
        ["Jakub","Kochanowski",21],
        ["Anna","Galanty",21]
    ],
    meta : {
    }
}
```

Inspiracją do zdefiniowania takiego standardu komunikatów była struktura wyjściowa konwertera *PapaParse* opisanego we wcześniejszej części pracy.

W przypadku, gdy obiekt posiada informacje o wierszu nagłówka można zarzucić temu rozwiązaniu rozrzutność jeśli chodzi o zasoby sprzętowe. Informacja o nazwach kolumn przechowywana jest zarówno w obiekcie opisującym meta dane jak i każdym obiekcie reprezentującym poszczególne wiersze. Rozwiązanie, które samo się nasuwa na myśl jest proste: pozostawić dane o nagłówkach w części z metadanymi, a tabelę obiektów zamienić na tabelę tabel tak jak jest to w przypadku opcji bez nagłówka. Głównymi powodami dlaczego postanowiono jednak pozostać przy redundancji danych były:

- Z powodu modularności rozwiązania dane będę przemieszczały się pomiędzy różnymi otoczeniami. Wychodząc z modułu będą obsługiwane przez środowisko Wirecloud, aby następnie wejść do innego modułu. Przesyłanie obiektów może prowadzić do wzrostu ryzyka błędu w danych. Ich redundancja natomiast minimalizuje to ryzyko i pozwala wykryć błędy w informacji na podstawie porównania nadmiarowości ze sobą i stwierdzeniu czy są jednakowe.
- Język *JavaScript* jest zdefiniowany tak, że tablice są rozszerzeniem obiektów. W uproszczeniu, stworzona tablica jest tak naprawdę obiektem, który posiada uporządkowane własności. Zdefiniowanie kolejności pól nie jest wymagane w przypadku, gdy znane są nazwy kolumn.

Przedstawiony format został zastosowany przy budowie wszystkich modułów analitycznych.

6. Stworzone moduły

W tym rozdziale zostanie przedstawiony stworzony zestaw modułów przeznaczonych do analizy danych. Zestaw ten został podzielony na dwie podgrupy. Jedna przedstawia stworzone operatory, a druga widgety.

Poprzez często wykorzystywany w tym rozdziale zwrot „standardowa struktura” lub „standardowy obiekt” autor ma na myśli obiekt w formacie danych wejścia/wyjścia opisanym w rozdziale piątym.

Wszystkie moduły wykorzystują bibliotekę *Ajv* do walidacji danych wejściowych w oparciu o schematy *JSON Schema*. Struktury te są opisane w rozdziale Dodatki.

6.1. Operatory

Komponenty typy operator skupiają się na manipulacji danymi oraz ich obróbce. Niektóre moduły z tej podgrupy są wyłącznie skierowane na zmiane formatu danych. Należą do nich: „Wybór kolumny”, „Łączenie danych” oraz „Formatowanie regex”.

Istnieją również operatory, których celem jest dokonywanie obliczeń na podstawie otrzymanych danych. Przeprowadzają one potrzebne kalkulacje i, na wyjściu, udostępniają rezultaty. Takim zadaniem zajmują się następujące, zaimplementowane moduły: „Korelacja”, „Usunięcie trendu” oraz „Histogram”.

6.1.1. Operator „Wybór kolumny”

Operator ten, ma za zadanie wyłuskanie odpowiedniego pola z otrzymanego obiektu i przekazanie na wyjście struktury z jedną kolumną. Wyjściowe dane wciąż utrzymują standard komunikatów pomiędzy modułami.

Aby element ten działał poprawnie, w jego preferencjach należy podać nazwę kolumny, którą chce się przekazać na wyjście.

6.1.2. Operator „Scalanie obiektów”

Zadaniem tego modułu jest połączenie dwóch obiektów w jeden. Założeniem udanego łączenia jest identyczna ilość wierszy i istnienie nagłówka w obydwu obiektach wejściowych.

Operator ten posiada dwa wejścia oraz jedno wyjście. Wejścia nie są tożsame. W przypadku, gdy tabele posiadają kolumnę o tej samej nazwie, dane z pierwszego wejścia będą miały priorytet nad danymi z wejścia drugiego. W przypadku tego modułu użytkownik nie musi definiować żadnych preferencji.

6.1.3. Operator „Formatowanie regex”

Operator „Formatowanie regex” jest kolejnym komponentem skupiającym się na modyfikacji danych. Jego zadaniem jest przekształcenie wartości w wybranej kolumnie według zadanego przez użytkownika wyrażenia regularnego.

Moduł ten posiada jedno wejście oraz jedno wyjście. Na wejściu przyjmuje on obiekt do przekształcenia. W przypadku pozytywnej weryfikacji wejścia następuje wysłanie zmodyfikowanej struktury na wyjście. Do poprawnego działania tego operatora preferencje użytkownika muszą definiować nazwę kolumny oraz wyrażenie regularnego.

6.1.4. Operator „Korelacja”

Pierwszym z grupy komponentów odpowiedzialnych za dokonywanie obliczeń jest operator korelacji. Jego zadaniem jest obliczenie współczynnika korelacji Pearsona na podstawie dwóch kolumn. Korzysta on z biblioteki *everpolate* w celu obliczenia szukanej wartości.

Komponent ten posiada jedno wejście oraz jedno wyjście. Na wejściu operator ten powinien otrzymać obiekt w zdefiniowanym standardzie. W przypadku, gdy wejściowe dane posiadają odpowiednie kolumny oraz są poprawne składniowo, na wyjście zostają wysłane informacje o nazwach kolumn oraz wyliczonym współczynniku spakowane w standardową strukturę. Przykład wyjścia tego operatora znajduje się poniżej.

```
{
    data : [
        {firstColumn:"P",secondColumn:"D",correlationCoefficient:0.15}
    ],
    meta : {
    fields: ["firstColumn","secondColumn","correlationCoefficient"]
    }
}
```

6.1.5. Operator „Usunięcie trendu”

Operator ten może być zaliczany do modułów liczących. Jego zadaniem jest usunięcie trendu liniowego z wybranej przez użytkownika kolumny. W celu obliczenia regresji liniowej metodą najmniejszych kwadratów korzysta on z biblioteki *everpolate*.

Moduł ten posiada jedno wejście oraz dwa wyjścia. W celu poprawnego działania użytkownik musi zdefiniować nazwę kolumny, z której należy usunąć trend. Opcjonalną preferencją jest również zdefiniowanie nazwy pola zawierającego przebieg czasowy, według którego pojawiają się wartości. W przypadku braku informacji o momentach wystąpienia próbek moduł zakłada ich równomierne rozmieszczenie na osi czasu. Przy braku wystąpienia błędów, na jedno z wyjść zostaje wysłany obiekt wejściowy z usuniętym trendem na wybranej kolumnie, a na drugie struktura opisująca linie najlepszego dopasowania przedstawiona poniżej.

```
{
    data: [{
            slope: 3,   // współczynnik a
            intercept: 2 // współczynnik b
        }
    ],
    meta: {fields: ["slope","intercept"]}
}
```

6.1.6. Operator „Histogram”

Zadaniem ostatniego z elementów jest stworzenie rozkładu z próby danej kolumny. Można go zaliczyć do grupy operatorów liczących.

Posiada on jedno wejście i jedno wyjście. Od użytkownika wymagane jest podanie nazwy kolumny, która powinna być poddana analizie. W przypadku, gdy obiekt wejściowy nie posiada wiersza nagłówka, zamiast nazwy można podać numer danego pola. Jeżeli moduł nie napotkał na błędy, na wyjście wysyłana jest standardowa struktura zawierająca informację o rozkładzie empirycznym danej tabeli. Poniżej został przedstawiony przykład takiej struktury. Jak widać spełnia on zdefiniowany wcześniej format komunikatów.

```
1. {
2.     data : [
3.     {value:"Jakub",quantity:4},
4.     {value:"Jan",quantity:2},
5.     {value:"Adam",quantity:6},
6.     {value:"Ewa",quantity:3}
7.     ],
8.     meta : {
9.     fields: ["value","quantity"]
10.    }
11. }
```

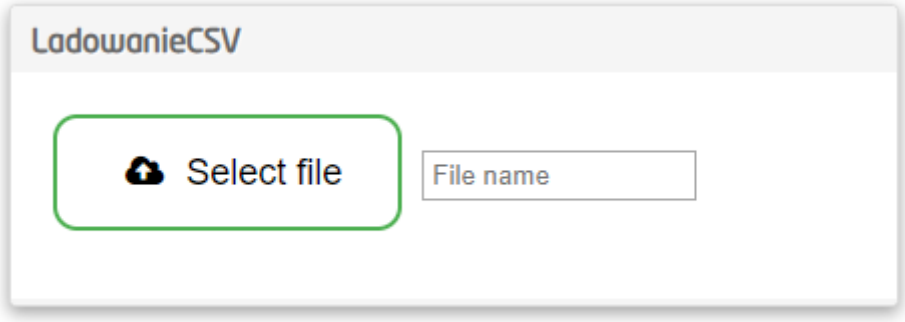
6.2. Widgety

W trybie deski rozdzielczej środowiska Wirecloud widgety, w przeciwieństwie do operatorów, posiadają swoją formę graficzną. Ich głównym celem jest prezentacja informacji użytkownikowi oraz interakcja z nim. Podczas tworzenia tej pracy dyplomowej zaimplementowano trzy moduły tego rodzaju.

6.2.1. Widget „Ładowanie CSV”

Moduł ładujący plik *CSV* i konwertujący go do standardowej struktury jest głównym źródłem danych dostarczonych przez użytkownika. Mimo posiadania charakterystyki operatora, został zaimplementowany jako widget w celu umożliwienia interaktywnego wyboru pliku z systemu operacyjnego. Do konwersji wykorzystuje on bibliotekę *PapaParse*.

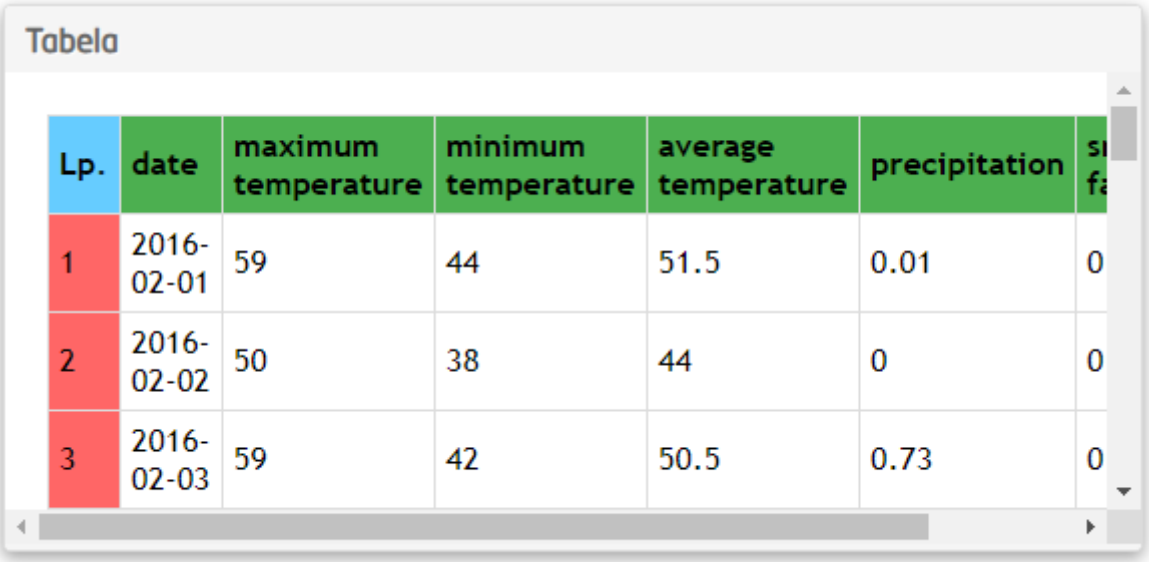
Komponent ten, z perspektywy środowiska Wirecloud, nie posiada wejścia. Pobiera on dane poprzez interakcję z użytkownikiem. Po wybraniu pliku, moduł dokonuje konwersji. W przypadku niezaobserwowania błędów wysyła na wyjście standardowy obiekt reprezentujący załadowane dane. W opcjach widgetu istnieje możliwość określenia, czy dany plik CSV posiada wiersz nagłówka oraz z jakiego separatora korzysta. Rys. 10 przedstawia wygląd komponentu.



Rys. 10 Reprezentacja graficzna widgetu ładującego plik CSV.
Źródło: Własne

6.2.2. Widget „Tabela”

Komponent ten ma za zadanie wyświetlić użytkownikowi tabelę na podstawie zgodnego ze standardem obiektu wejściowego. Nie posiada on wyjścia ani możliwości zdefiniowania preferencji. Pełni funkcję „ekranu” wyświetlającego dane w środowisku Wirecloud. Rys. 11 przedstawia wygląd tego modułu. Został on zaimplementowany przy użyciu czystego języka JavaScript. Po otrzymaniu obiektu wejściowego o odpowiedniej strukturze, analizuje go i wyświetla jego zawartość.

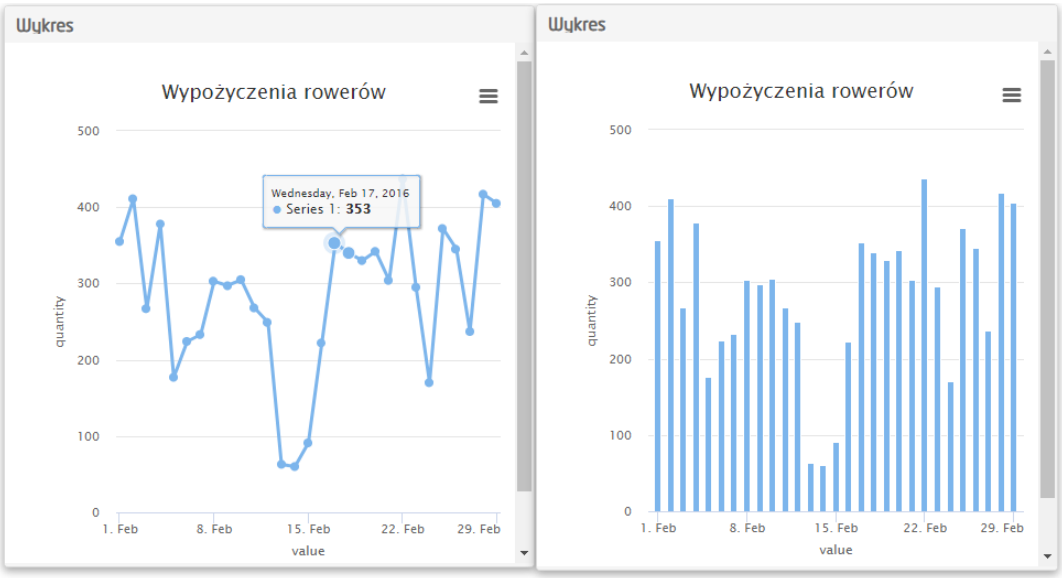


Rys. 11 Wygląd modułu tabeli z przykładową zawartością.
Źródło: Własne

6.2.3. Widget „Wykres”

Komponent tworzący wykres jest głównym, obok widgetu rysującego tabelę, modułem ekranowym wyświetlanym użytkownikowi. W celu rysowania estetycznych kształtów wykorzystuje on bibliotekę Highcharts.

W tym przypadku udostępniane są dwa wejścia reprezentujące oś X i oś Y wykresu. Każde z nich przyjmuje obiekt standardowy zredukowany przy pomocy operatora „Wybór kolumny” do jednego pola. W preferencjach użytkownik może zdefiniować rodzaj wykresu (słupkowy, liniowy, punktowy), tytuł wykresu oraz typ danych na osi X (numeryczne, kategoria, czas). Istnieje limit ilościowy punktów możliwych do narysowania równy 10000.



Rys. 12 Reprezentacja graficzna dwóch przykładowych modułów typu wykres.
Źródło: Własne

7. Scenariusze

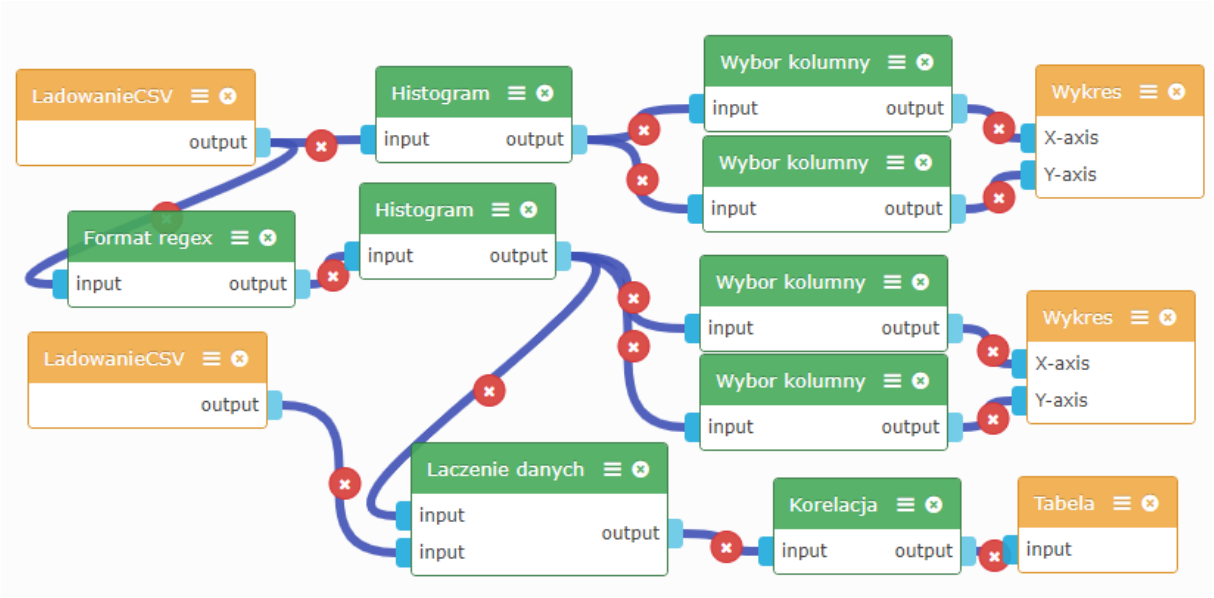
W tym rozdziale zostały opisane hipotetyczne przypadki wykorzystania wymienionych wcześniej modułów. Scenariusze zostały stworzone na bazie rzeczywistych plików. Głównym źródłem zbiorów danych była witryna *kaggle.com* udostępniająca swoje zasoby do użytku publicznego. Celem stworzenia przykładowych sytuacji było pokazanie modularnego charakteru analizy danych oraz możliwości wykorzystania wcześniej stworzonych komponentów w różnych przypadkach użycia.

7.1. Analiza wypożyczeń rowerów w Nowym Yorku

Głównym aktorem tego scenariusza jest mała firma zajmująca się wynajmem rowerów miejskich w Nowym Yorku. W tym przypadku właściciele chcą się dowiedzieć, czy istnieje korelacja pogody z ilością wypożyczeń danego dnia. Chcieli by również mieć możliwość wyświetlenia rozkładu empirycznego dziennej ilości wynajmów oraz uzyskania informacji o najbardziej popularnych stacjach.

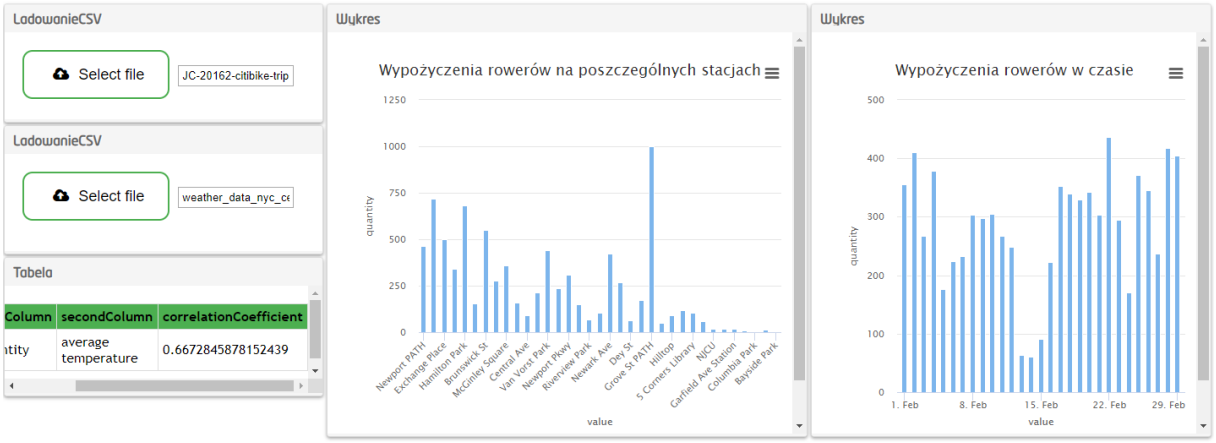
Dane potrzebne do zrealizowania tego scenariusza zostały pobrane z witryny *kaggle.com.* W ich skład wchodzą dane o pogodzie w lutym 2016 w Nowym Yorku oraz dane o wypożyczeniach rowerów miejskich w tym mieście.

W tej sytuacji zostały wykorzystane następujące typy komponentów: „Ładowanie CSV”, „Wykres”, „Tabela”, „Wybór kolumny”, „Łączenie danych”, „Histogram”, „Formatowanie regex” oraz „Korelacja”.



Rys. 13 Zestawienie modułów realizujące pierwszy scenariusz
Źródło: Własne

Na Rys. 13 zostało zaprezentowane zestawienie realizujące zamierzone cele hipotetycznego przypadku użycia. Wykorzystano dwa moduły ładujące dane ponieważ informacje o pogodzie są w innym pliku CSV niż informacje o wynajmie rowerów. Reprezentacja graficzna wyświetlana użytkownikowi końcowemu została przedstawiona na Rys. 14.



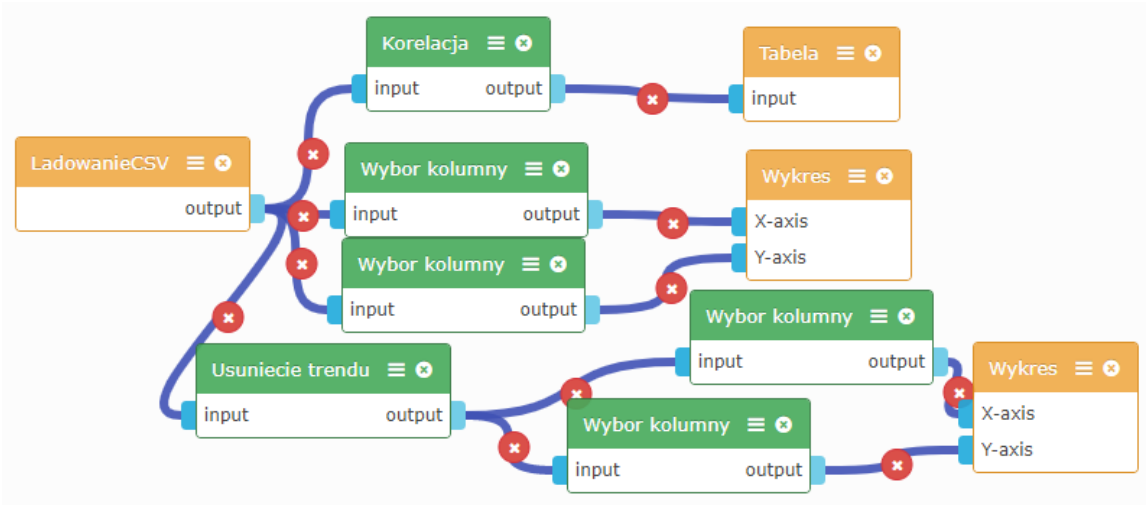
Rys. 14 Przedstawienie zrealizowanego scenariusza w trybie Dashboard.
Źródło: Własne

Jak widać, dzięki stworzonym modułom udało się osiągnąć cele właścicieli firmy. Okazało się, że w lutym 2016 najpopularniejszą stacją w ich ofercie była stacja o nazwie *„Grove St PATH”.* W tym miesiącu współczynnik korelacji pomiędzy średnią temperaturą, a ilością pobranych rowerów danego dnia wyniósł około 0.667 co może wskazywać na dość dużą zależność pomiędzy tymi zjawiskami.

7.2. Analiza cen mieszkań w dzielnicy Bemowo

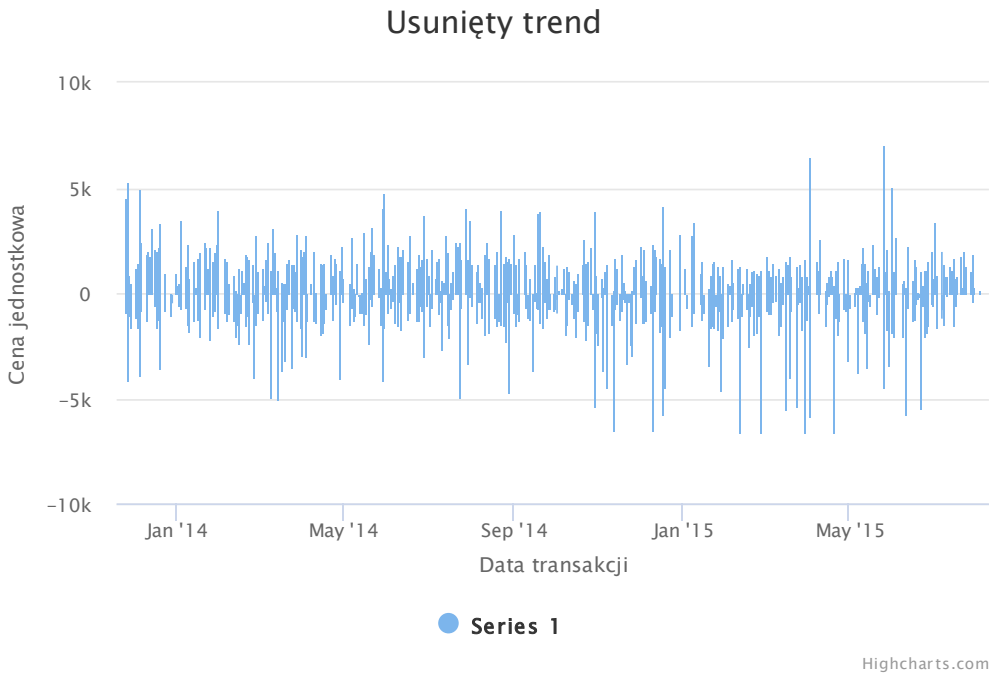
W tym scenariuszu zamierzamy przeanalizować dane ze sprzedaży lokali w warszawskiej dzielnicy Bemowo. Celem tego scenariusza jest zdefiniowanie współczynnika korelacji Pearsona pomiędzy wielkością mieszkania, a ceną za metr kwadratowy. Hipotetycznemu użytkownikowi zależy również na usunięciu trendu z wykresu zależności ceny jednostkowej mieszkania od daty wykonania transakcji.

Dane o sprzedaży lokalów w dzielnicy Bemowo zostały udostępnione ze źródeł Pana dr. Inż. Mariusza Kamoli na wyłączny użytek w pracy dyplomowej. Istnieje możliwość zakupu tego rodzaju danych. Zawierają one informacje ze sprzedaży lokali mieszkalnych z od listopada 2013 roku do sierpnia 2015 roku. Kolumny, które są wykorzystywane w tym scenariuszu to między innymi: Data transakcji, Powierzchnia użytkowa, oraz cena jednostkowa.



Rys. 15 Zestawienie modułów realizujące założenia drugiego scenariusza.
Źródło: Własne

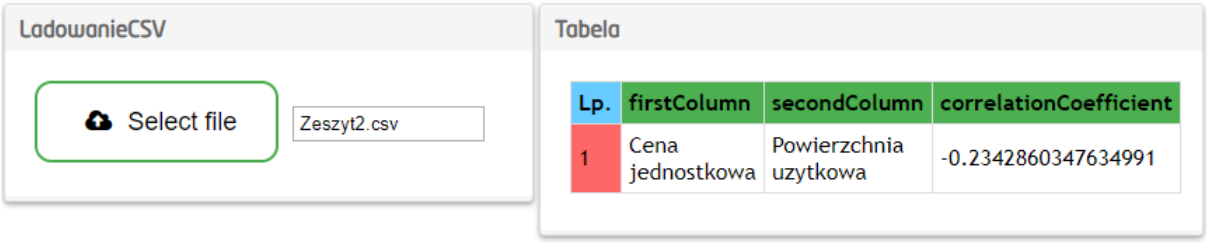
Na Rys. 15 przedstawiono połączone moduły w środowisku Wirecloud. Widać, w zestawieniu ze scenariuszem pierwszym, że duża część komponentów została wykorzystana powtórnie. To pokazuje jak powtarzalne są niektóre składowe typowych sytuacji o charakterze analitycznym.



Rys. 16 Wykres z usuniętym trendem przedstawiający ceny za metr kwadratowy lokalów w dzielnicy Bemowo.
Źródło: Własne

Z analizy dokonanej dzięki zaimplementowanym wcześniej operatorom i widgetom, udało się uzyskać współczynnik korelacji Pearsona pomiędzy ceną jednostkową mieszkania, a jego wielkością. Okazało się, że współczynnik ten równy jest -0.234 co może prowadzić do wysnucia teorii, że im większe było mieszkanie na Bemowie, tym mniejsza była jego cena za metr kwadratowy.

Wykres reprezentujący dane zawierające trend liniowy nie został przedstawiony z powodów estetycznych pracy. Jest on dostępny w rozdziale „Dodatki”.



Rys. 17 Część reprezentacji graficznej zrealizowanego scenariusza nr. 2.
Źródło: Własne

8. Możliwości rozwoju

Modularny charakter stworzonego oprogramowania daje ogromne szanse na rozwój. Ciągłe powiększanie katalogu dostępnych modułów bez potrzeby zmiany starego oprogramowania daje możliwość rozszerzania tego środowiska w nieskończoność. Istnieje wiele zagadnień analitycznych, których zaimplementowane w tej pracy elementy nie poruszają.

Inną drogą ulepszenia zaprezentowanego rozwiązania jest skupienie się na poprawie niektórych aspektów systemu Wirecloud. Lepsze wsparcie techniczne oraz poprawa mechanizmu wspomagania użytkownika w odpowiednim łączeniu elementów są pierwszymi, które przychodzą na myśl. Istotnym krokiem w przód byłaby również korekcja błędów znajdujących się w programie Wirecloud.

Możliwe jest również przeniesienie obowiązku wykonywania skomplikowanych obliczeń z lokalnej maszyny użytkownika na zewnętrzny serwer. Takie rozwiązanie posiada ogromne plusy jakimi są przyspieszenie lub nawet umożliwienie wykonywanych działań matematycznych. Odstąpienie od języka JavaScript umożliwiłoby również zrównoleglenie części zadań. Rozwiązanie to posiada jednak swoje wady. Potrzeba wysyłania danych poza lokalne środowisko zmniejszałaby ich bezpieczeństwo. Rozwiązanie to wymagałoby również stworzenia odpowiedniej infrastruktury po stronie maszyny serwerowej co skomplikowałoby cały system.

W tej chwili ograniczenia modułów sprawiają, że pliki na których użytkownik może operować są wielkości kilku MB. Spowodowane jest to możliwościami renderowania przeglądarki oraz ograniczeniami systemu Wirecloud. Usunięcie tej bariery znacznie przyczyniłoby się do rozszerzenia funkcjonalności rozwiązania.

Kolejną drogą jest zrezygnowanie z rozwiązania organizacji Fiware i przejście inną platformę. Równało by się to z przeprogramowaniem istniejących już komponentów. Mogłaby istnieć również konieczność dostosowania standardu opisu wejść i wyjść modułów.

Zdefiniowany format wejść/wyjść umożliwia łatwą jego modyfikację w zakresie obejmowanych metadanych. Dostosowywanie go do potrzeb implementacyjnych kolejnych modułów nie stanowi problemu.

Wprowadzenie zaimplementowanych komponentów do sklepu środowiska Wirecloud byłoby możliwością dotarcia z tym rozwiązaniem do innych osób. Wymagałoby to jednak głębszej analizy potrzeb użytkowników oraz przyłożenia większej wagi do testowania samych modułów. Napisanie czytelnej dokumentacji stanowiłoby, w tym przypadku, również pewne wyzwanie.

9. Podsumowanie

W czasie tworzenia pracy dyplomowej udało się zrealizować założone na samym początku cele. Wykonano zestaw dziewięciu modułów analitycznych do z góry narzuconego środowiska Wirecloud. Udało się utworzyć standard komunikatów, który był motywem przewodnim tego projektu. W celu utrzymania kompatybilności rozwiązań, kolejne moduły powinny go stosować. Udało się zrealizować funkcjonalność do dwóch wcześniej zdefiniowanych sytuacji hipotetycznych, które miały reprezentować rzeczywiste potrzeby użytkowników.

Przed przystąpieniem do pracy autor był laikiem jeśli chodzi o technologie webowe takie jak JavaScript, HTML, CSS czy XML. Wykonanie wymienionych na poprzednich stronach działań pozwoliło uzyskać niemałe doświadczenie z tego zakresu wiedzy. Udało się również zaznajomić z takimi pojęciami jak JSON czy JSON Schema. Są to rozwiązania aktualnie szeroko wykorzystywane w rozwiązaniach webowych. Rozwijającym również było zaznajomienie się ze środowiskiem Wirecloud.

W celu osiągnięcia jak najwyższej funkcjonalności oraz optymalności rozwiązań końcowych do stworzenia modułów wykorzystano jedne z najlepszych narzędzi dostępnych na rynku tj. biblioteki *Highcharts* oraz *PapaParse*.

W ramach testów sprawdzających poprawność działania modułów, oprócz opisanych wcześniej scenariuszy, przeprowadzono szereg przykładowych zestawień. Badano odporność na nieprawidłowe dane wejściowe, jak i na źle zdefiniowane preferencje. Podczas tych działań skupiono się również na systemie informowania użytkownika o występujących nieprawidłowościach w funkcjonowaniu poszczególnych elementów.

9.1. Wnioski

Podczas pracy ze środowiskiem Wirecloud zauważono dużo jego braków oraz niedociągnięć. Jeżeli to rozwiązanie miałoby być platformą docelową do rozwoju katalogu modułów analitycznych, powinno być ono dostosowane do takich funkcjonalności. Obecnie środowisko to nie jest preferowane do tego typu zadań. Z powodu swojej specyfikacji jest ono przeznaczone do integracji z innymi API organizacji Fiware.

Mimo wad środowiska zasadnym wydaje się sam pomysł składania aplikacji analitycznych z małych komponentów. Patrząc na rozwiązania takie jak *KNIME* czy *RapidMiner* można stwierdzić, że modularne podejście do analizy danych ma przyszłość. Paradygmat potokowego przetwarzania informacji świetnie się w nich sprawdza. Dzięki tego typu zastosowaniom istnieje możliwość dotarcia do użytkowników z minimalną wiedzą programistyczną.

Reasumując, pomysł zestawu modułów przeznaczonych do zadań czysto analitycznych powinien być rozwijany, jednak środowisko Wirecloud nie jest do tego przystosowane.

10. Bibliografia

1. **Arranz, Álvaro**. Documentation. Fiware Wirecloud. [Online] Wrzesień 13, 2018. [Cited: Styczeń 10, 2019.] https://wirecloud.readthedocs.io/en/latest/.

2. **Bray, T**. RFC 8259: The JavaScript Object Notation (JSON) Data Interchange Format. [Online] Grudzień 2017. [Cited: Styczeń 16, 2019.] https://www.rfc-editor.org/rfc/pdfrfc/rfc8259.txt.pdf.

3. **Tim Bray, Jean Paoli, C. M. Sperberg-McQueen, Eve Maler, François Yergeau, John Cowan**. W3C Recommendation: Extensible Markup Language (XML) 1.1 (Second Edition). [Online] Sierpień 16, 2006. [Cited: Styczeń 22, 2019.] http://www.w3pdf.com/W3cSpec/XML/2/REC-xml11-20060816.pdf.

4. **Chumichev, Boris**. Everpolate: JavaScript Numerical Interpolation library. [Online] 2017. [Cited: Styczeń 14, 2019.] http://borischumichev.github.io/everpolate/.

5. **Dave Raggett, Arnaud Le Hors, Ian Jacobs**. HTML 4.0 Specification - W3C recommendation. [Online] 1999. [Cited: Styczeń 14, 2019.] https://www.immagic.com/eLibrary/ARCHIVES/SUPRSDED/W3C/W980424S.pdf.

6. **Felipe Pezoa, Juan L. Reutter, Fernando Suarez, Martín Ugarte, Dogamoj Vrgoč**. Foundations of JSON Schema. In Proceedings of the 25th International Conference on World Wide Web (WWW '16). Republic and Canton of Geneva, Switzerland : International World Wide Web Conferences Steering Committee, 2016. 263-273. DOI: https://doi.org/10.1145/2872427.2883029.

7. **Fiware**. Account Management. wiki-Fiware. [Online] Styczeń 18, 2018. [Cited: Styczeń 18, 2019.] https://forge.fiware.org/plugins/mediawiki/wiki/fiware/index.php/Account_Management.

8. **Flanagan, David**. JavaScript: The Definitive Guide, 6th Edition. USA : O'Reilly Media, Maj 2011. ISBN 9781449308162.

9. **Highcharts**. Highcharts: EVERYBODY'S FAVORITE CHARTING LIBRARY. [Online] [Cited: Styczeń 14, 2019.] https://www.highcharts.com/blog/products/highcharts/.

10. **Holt, Matt**. PapaParse - documentation. [Online] [Cited: Styczeń 25, 2019.] https://www.papaparse.com/docs.

11. **json-schema-org**. Specyfication. Home of JSON Schema. [Online] Marzec 19, 2018. [Cited: Styczeń 14, 2019.] https://json-schema.org/specification.html.

12. **Meijs, Maartje.** KNIME vs. RapidMiner. Cmotions: Fact based decisions. [Online] Kwiecień 10, 2017. [Cited: Styczeń 10, 2019.] https://cmotions.nl/en/knime-vs-rapidminer/.

13. **Michael R. Berthold, Tobias Kötter, Nicolas Cebron, Thomas R. Gabriel, Thorsten Meinl, Kilian Thiel, Bernd Wiswedel, Peter Ohl,**. KNIME - the Konstanz information miner: version 2.0 and beyond, tom 11, zagadnienie 1, strony 26-31. Konstanz, Niemcy : ACM SIGKDD Explorations Newsletter, 2009. DOI: http://dx.doi.org/10.1145/1656274.1656280.

14. **Muenchen, Robert A**. The Popularity of Data Science Software. [Online] Czerwiec 6, 2017. [Cited: Styczeń 10, 2019.] http://r4stats.com/articles/popularity/.

15. **Piatetsky, Gregory**. Gainers and Losers in Gartner 2018 Magic Quadrant for Data Science and Machine Learning Platforms. KDnuggets. [Online] Luty 23, 2018. [Cited: Styczeń 14, 2019.] https://www.kdnuggets.com/2018/02/gartner-2018-mq-data-science-machine-learning-changes.html.

16. **Shafranovich, Y.** RFC 4180: Common Format and MIME Type for Comma-Separated Values (CSV) Files. [Online] Październik 2005. [Cited: Styczeń 14, 2019.] https://tools.ietf.org/html/rfc4180.

17. **Thompson, Gregory**. Self-Service Data Analytics for Smaller Companies (SME's). Data Science Central. [Online] Lipiec 30, 2017. [Cited: Grudzień 28, 2018.] https://www.datasciencecentral.com/profiles/blogs/self-service-data-analytics-for-smaller-companies-sme-s.

11. Spis rysunków

12. Dodatki

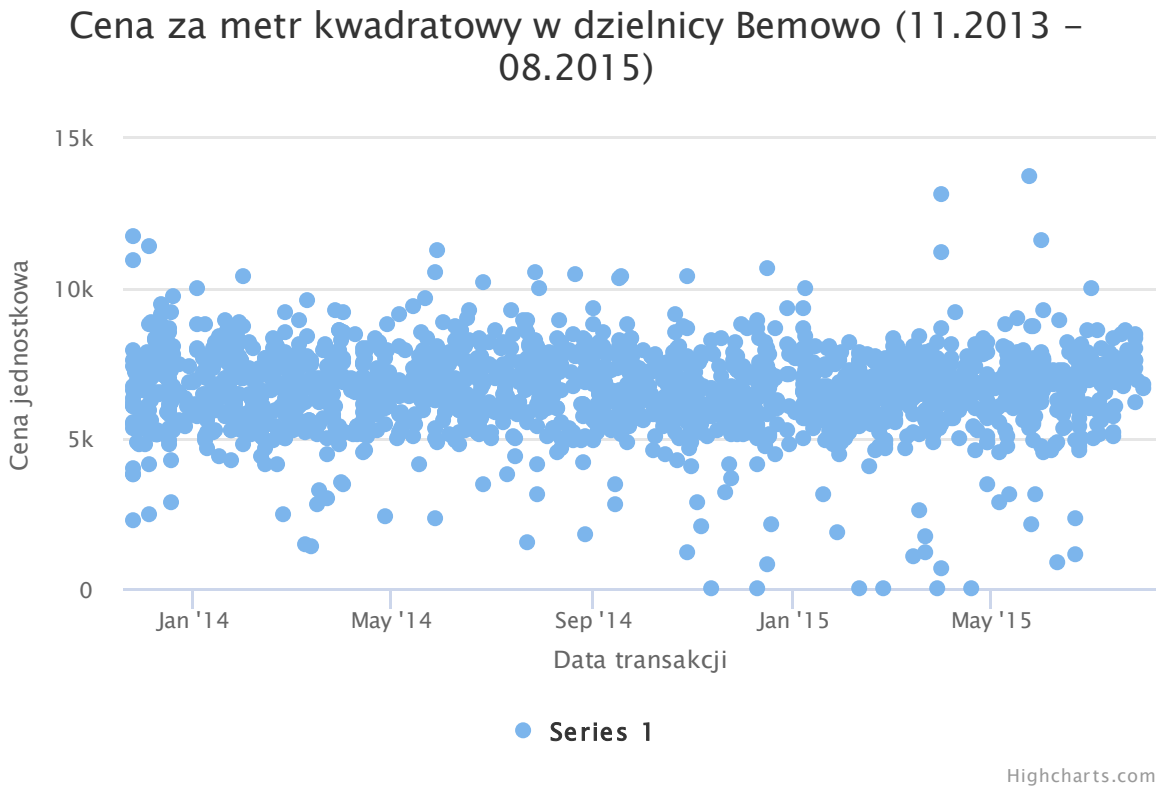
JSON Schema weryfikujący zarówno obiekty z nagłówkami, jak i bez nich.

```
{
  "$id": "http://example.com/root.json",
  "$schema": "http://json-schema.org/draft-07/schema#",
  "type": "object",
  "title": "The Root Schema",
  "required": [
    "data",
    "meta"
  ],
  "oneOf": [
    {
      "properties": {
        "data": {
          "$id": "#/properties/data",
          "type": "array",
          "title": "The Data Schema",
          "default": null,
          "minItems": 1,
          "items": {
            "$id": "#/properties/data/items",
            "type":"object",
            "title": "The Items Schema",
            "minProperties": 1
          }
        },
        "meta": {
          "$id": "#/properties/meta",
          "type": "object",
          "title": "The Meta Schema",
          "required": ["fields"],
          "properties": {
            "fields": {
              "$id": "#/properties/meta/properties/fields",
              "type": "array",
              "title": "The Fields Schema",
              "default": null,
              "minItems": 1,
              "items": {
                "$id": "#/properties/meta/properties/fields/items",
                "type": "string",
                "title": "The Items Schema",
                "default": "",
                "examples": [
                  "raz",
                  "dwa"
                ]
              }
            }
          }
        }
      }
    },{
      "properties": {
        "data": {
          "$id": "#/properties/data",
          "type": "array",
          "title": "The Data Schema",
          "default": null,
          "minItems": 1,
          "items": {
            "$id": "#/properties/data/items",
```

```
            "type":"array",
            "title": "The Items Schema",
            "minItems": 1
          }
        },
        "meta": {
          "$id": "#/properties/meta",
          "type": "object",
          "title": "The Meta Schema"
        }
      }
    }]}
```

JSON Schema weryfikujący obiekty zawierające pojedynczą kolumnę

```
{
  "$schema": "http://json-schema.org/draft-07/schema#",
  "$id": "http://example.com/root.json",
  "type": "object",
  "title": "The Root Schema",
  "required": [
    "data",
    "meta"
  ],
  "properties": {
    "data": {
      "$id": "#/properties/data",
      "type": "array",
      "title": "The Data Schema",
      "default": null,
      "minItems": 1,
      "items": {
        "$id": "#/properties/data/items",
        "type":"object",
        "title": "The Items Schema",
        "maxProperties": 1,
        "minProperties":1
      }
    },
    "meta": {
      "$id": "#/properties/meta",
      "type": "object",
      "title": "The Meta Schema",
      "required": [
        "fields"
      ],
      "properties": {
        "fields": {
          "$id": "#/properties/meta/properties/fields",
          "type": "array",
          "title": "The Fields Schema",
          "default": null,
          "minItems": 1,
          "maxItems": 1,
          "items": {
            "$id": "#/properties/meta/properties/fields/items",
            "type": "string",
            "title": "The Items Schema",
            "default": "",
            "examples": [
              "raz"
            ]
          }
        }
      } } } }
```



Rys. 18 Wykres zawierający trend