DANE PO WARSZAWSKU

Dokumentacja konfiguracyjna platformy

Jarosław Legierski
Henryk Rosa
Tomasz Janisiewicz

Dokumentacja wersji produkcyjnej platformy udostępniającej Open Data

Konfiguracja platformy

Flickr.com, jafsegal, CC BY

Projekt współfinansowany przez Narodowe Centrum Badań i Rozwoju w ramach Programu Innowacje Społeczne.

Spis treści

1 Cel projektu

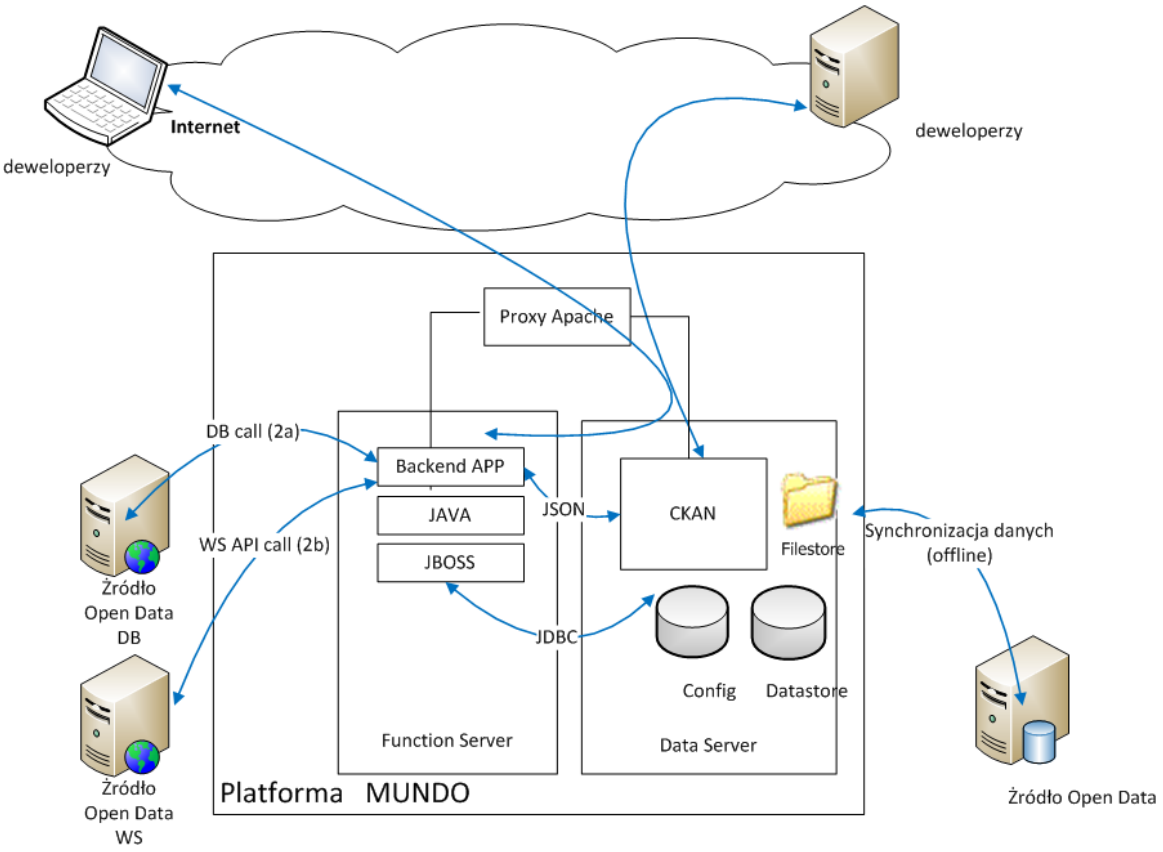
Celem projektu MUNDO [1] jest diagnoza możliwości ekspozycji oraz przygotowanie danych, które mogą zostać udostępnione przez Miasto Warszawa w modelu Open Data, a przede wszystkim budowa warstwy ekspozycji dla tych danych w formie platformy eksponującej API. Projekt MUNDO łączy w sobie opracowanie technologii do udostępnienia otwartych danych z metodologią społeczną dotyczącą włączania różnych grup społecznych w działania na rzecz poprawy życia w mieście. W ramach prac konsorcjum prowadzone były działania, które przygotowały techniczne zaplecze dla części społecznej projektu. Natomiast w części pilotażowej konsorcjum przeprowadziło konkurs na aplikacje internetowe dla Warszawy, by w ten sposób przekonać społeczeństwo o wadze otwartych danych (szczególnie publicznych) dla rozwoju miasta i ułatwienia funkcjonowania jego mieszkańców.

Innowacyjność projektu MUNDO polega na jego złożoności oraz na łączeniu ram technicznych z ramami metodologii partycypacji społecznej w życiu miasta. Wartością projektu jest również jego replikowalność w innych miastach Polski, gdyż zarówno oprogramowanie platformy do ekspozycji otwartych danych, jak i metodologia prowadzenia działań społecznych zostaną udostępnione do wykorzystania dla zainteresowanych jednostek z użyciem wolnych licencji (GPL, LGPL, FDL, CC).

2 Opis architektury platformy

W skład platformy wchodzą następujące komponenty:

- Proxy Server (opcjonalny) – warstwa bezpieczeństwa odpowiedzialna min za rozdział ruchu
- Data Server – oparty na systemie CKAN serwer będący: katalogiem danych, serwerem www i repozytorium danych plikowych i tabelarycznych
- Function Server – oparty na Java serwer, będący middleware dla wywołań API do źródeł danych opartych o Web Services, bazy danych itp. System ten ma zapewnić dostęp do danych dynamicznych, tj. danych, których źródłem są usługi sieciowe oraz bazy danych. Ten podsystem jest również odpowiedzialny za limitowanie wywołań, buforowanie danych, konwersje wywołań oraz konwersję formatu danych w celu uzyskania możliwie spójnego formatu API do wszystkich danych i funkcji UM Warszawa.



Rysunek 1 Elementy platformy

3 Dokumentacja konfiguracyjna platformy

W poniższym rozdziale zamieszczono procedurę konfiguracji elementów platformy MUNDO obejmującą opisy konfiguracji zainstalowanych już komponentów Data Server i Function Server

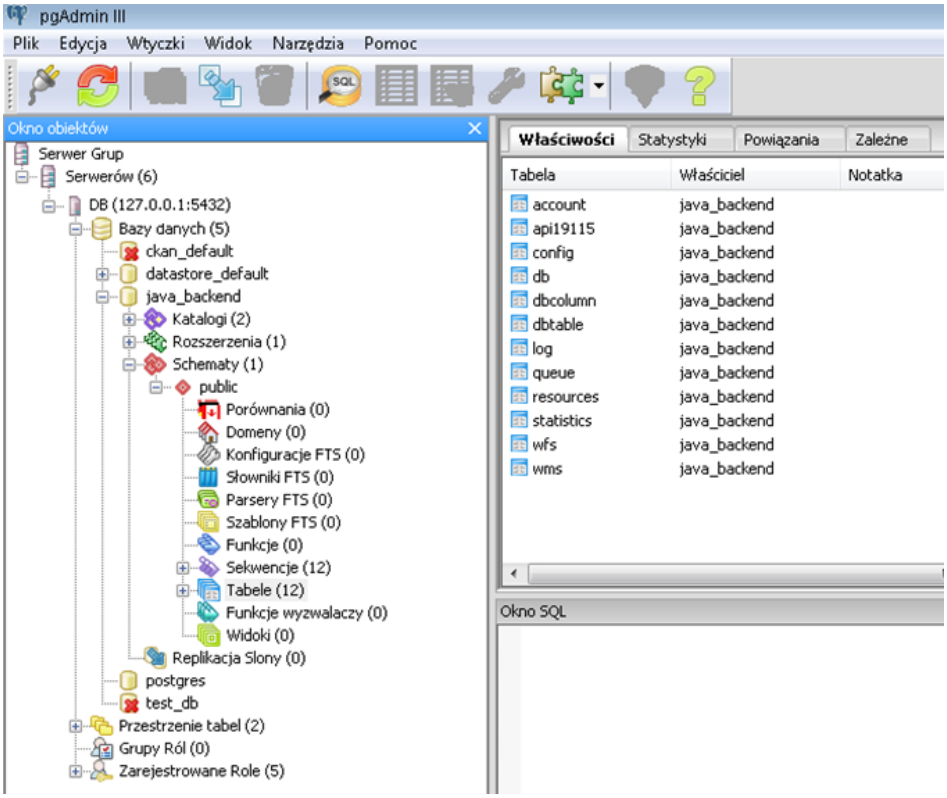
3.1 Konfiguracja podstawowa platformy

Po zainstalowaniu serwerów należy dokonać ich wstępnej konfiguracji. W niniejszym rozdziale przedstawiono kroki, jakie należy wykonać w celu wstępnego skonfigurowana elementów: Function Server i Data Server.

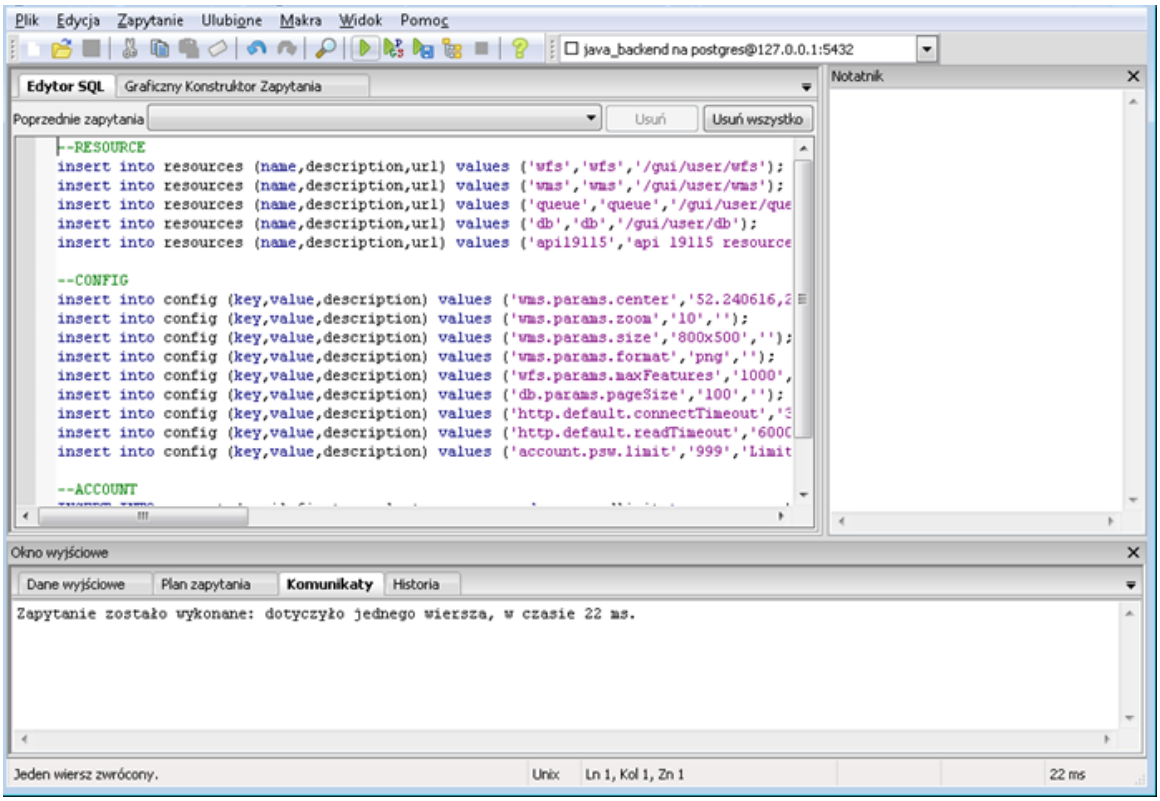
3.1.1 Konfiguracja Function Server

Konfiguracja inicjalizacyjna

W momencie instalacji pliku mundo-java-backend.war instalator tworzy odpowiednią strukturę bazy danych jednak jest ona pusta. W celu dodania wymaganych rekordów należy połączyć się z bazą danych np. za pomącą klienta PGAdmin i wykonać skrypt: init-config.sql



Rysunek 2 Tabele w bazie danych java_backend



Rysunek 3 Uruchomienie skryptu init-config.sql

Logujemy się do aplikacji Mundo Backend

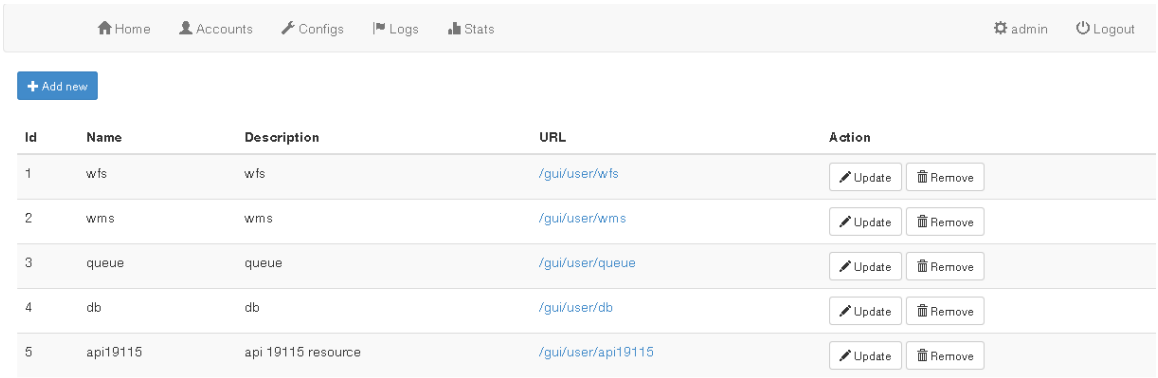
http://10.10.10.11:8080/cbr/mundo-java-backend/

user: admin

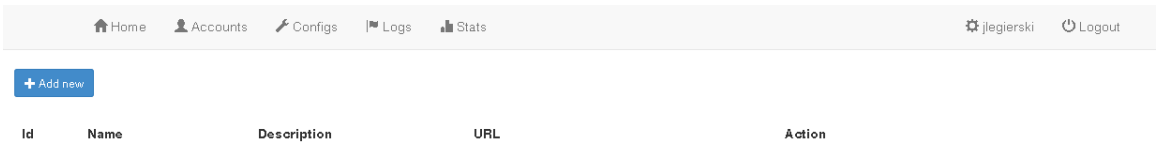
password: 123456

Rysunek 4 Okno logowania

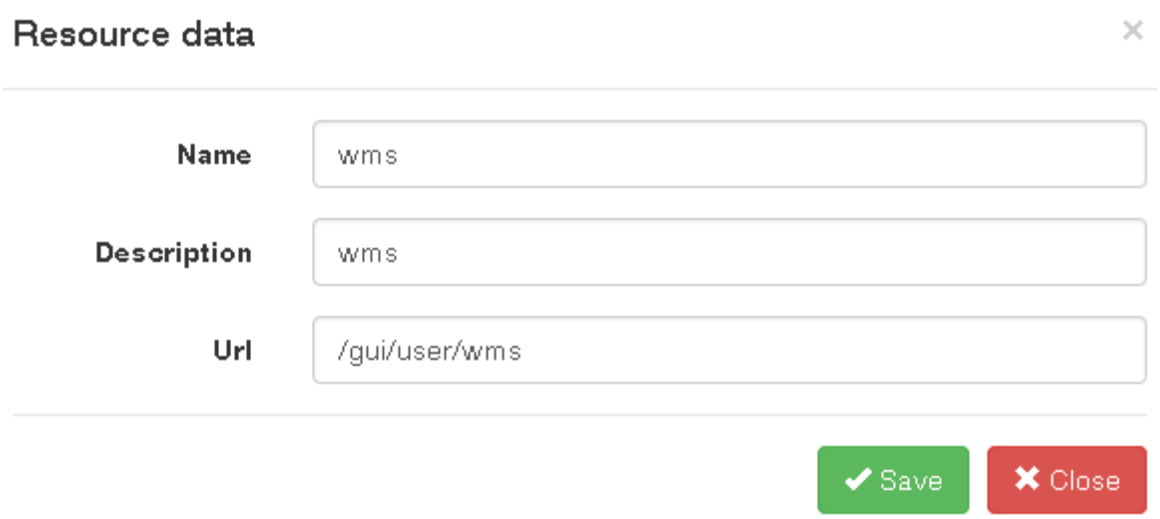
W zakładce home znajdują się predefiniowane typy zbiorów danych takie jak: wms, wfs, queue, db i api19115.



Rysunek 5 Predefiniowane typy zbiorów danych



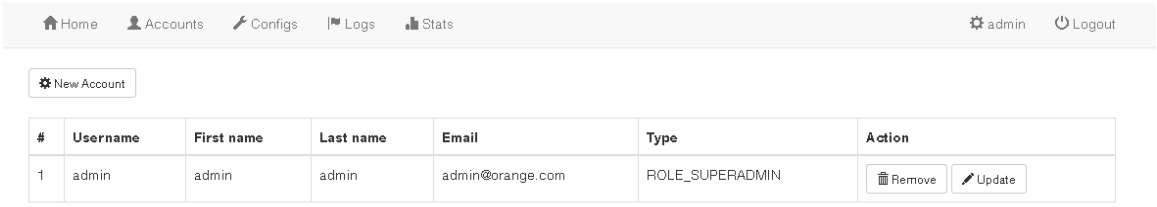
Rysunek 6 Dodawanie nowego typu danych



Rysunek 7 Dodawanie nowego typu danych

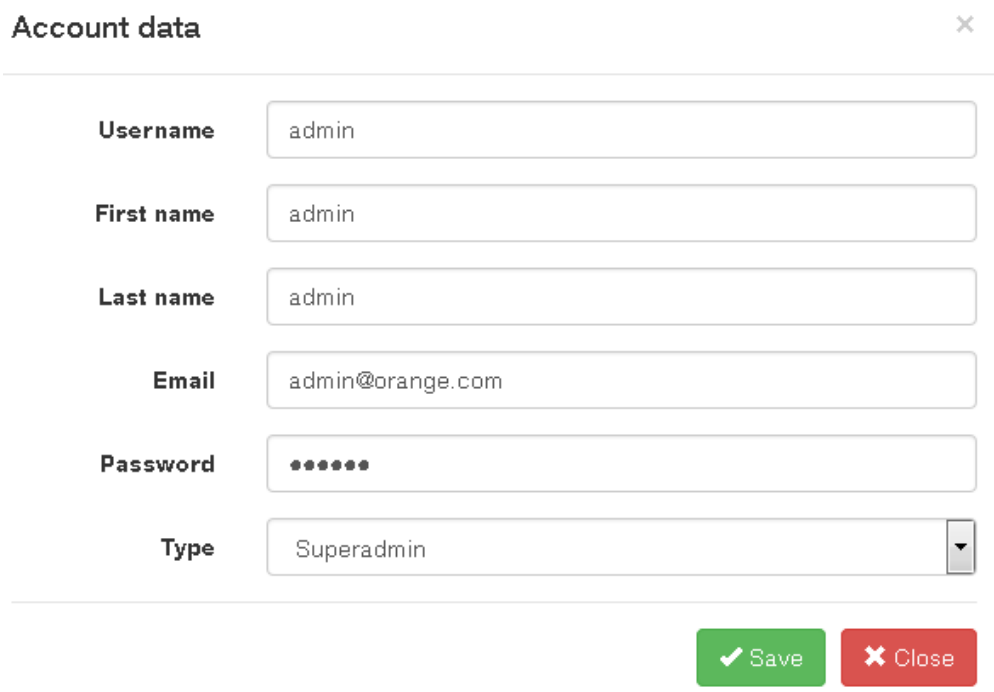
Konfiguracja kont użytkowników

Zakładka Accounts – ten element odpowiada za zarządzanie kontami użytkowników



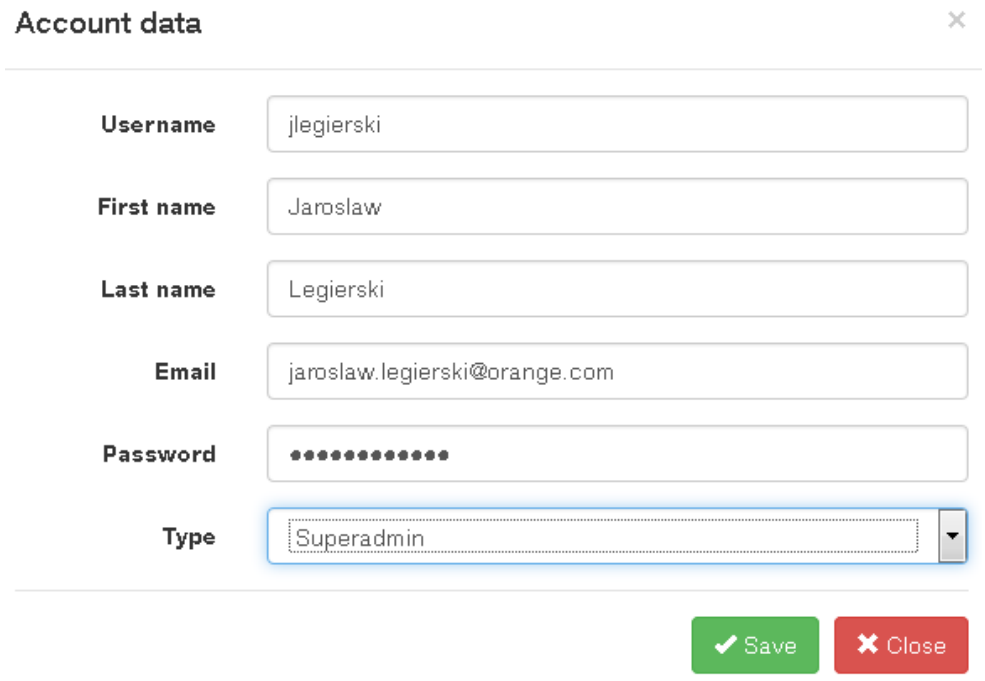
Rysunek 8 Konfiguracja kont użytkowników

W pierwszym kroku należy zmodyfikować domyślne hasło użytkownika admin korzystając z przycisku Update



Rysunek 9 Konfiguracja kont użytkowników

Poprzez przycisk New Account możemy utworzyć nowego użytkownika np:



Rysunek 10 Konfiguracja kont użytkowników

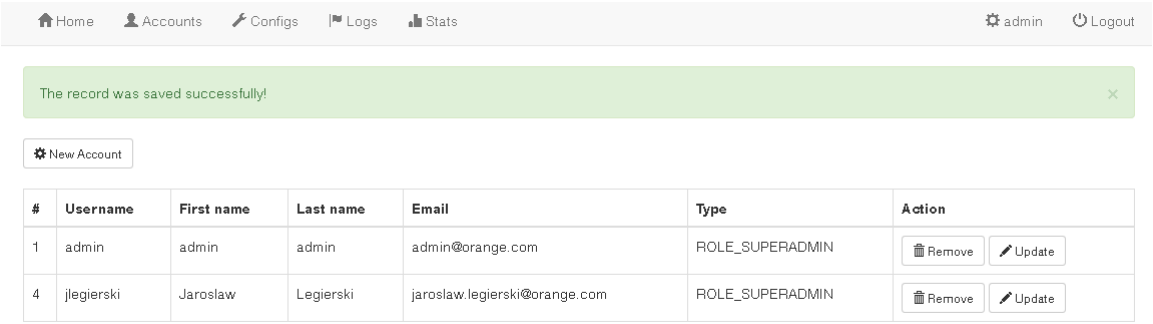
W systemie możemy tworzyć użytkowników o następujących uprawnieniach:

Superadmin – posiada dostęp do wszystkich elementów systemu.

Admin – dostęp do zakładek: Home, Logs, Stats.

User – dostęp do zakładki Home.

Rysunek 11 Wybór uprawnień użytkownika



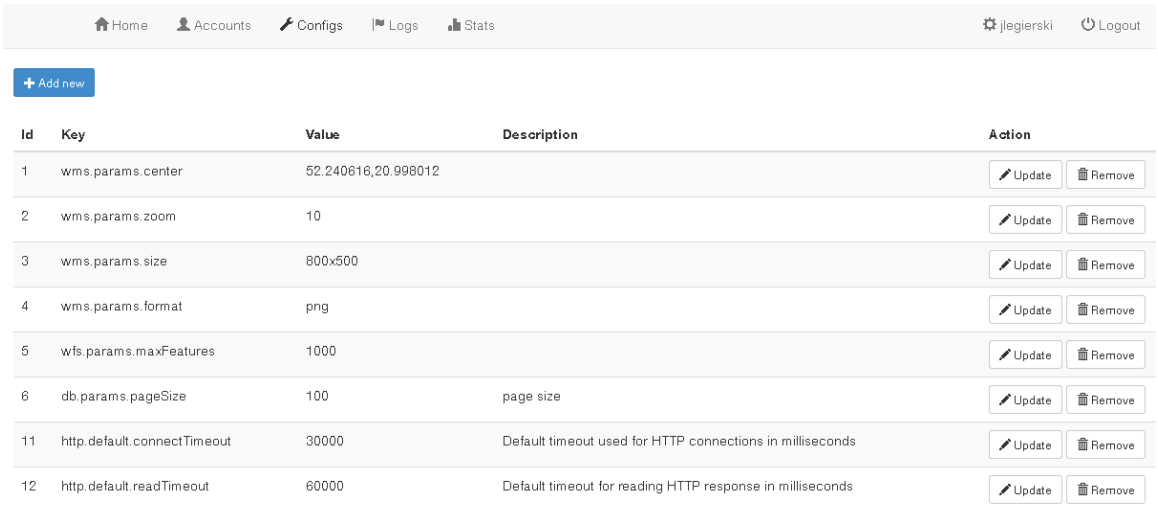
Rysunek 12 Przykładowe konta użytkowników

Parametry konfiguracyjne platformy

W zakładce Configs znajduje się lista parametrów konfiguracyjnych systemu:

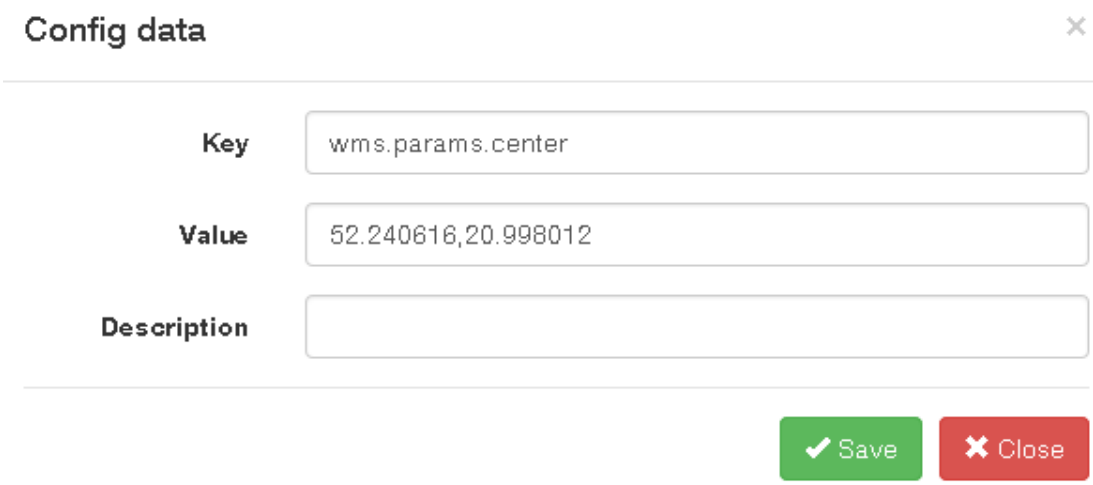
lp	parametr	Opis	Domyślna wartość
1	wms.params.center	Położenie centralnego punktu mapy dla danych typu wms	52.240616,20.998012
2	wms.params.zoom	Wartość parametru powiększenia dla danych typu wms	10
3	wms.params.size	Wartość parametru rozmiar obrazka dla danych typu	800x500

		wms	
4	wms.params.format	Wartość parametru format obrazka dla danych typu wms	png
5	wfs.params.maxFeatures	Maksymalna ilość danych pobieranych w jednym zapytaniu do systemu wfs	1000
6	db.params.pageSize	Maksymalna ilość rekordów zwracanych na jednej stronie dla danych typu db	100
7	http.default.connectTimeout	Domyślny limit czasu połączeń HTTP w milisekundach	30000
8	http.default.readTimeout	Domyślny limit czasu odczytu odpowiedzi HTTP w milisekundach	60000
9	account.psw.limit	Limit logowań. Używany do resetu hasła	999



Rysunek 13 Parametry konfiguracyjne systemu

Modyfikacji wartości danego parametru dokonujemy przy pomocy przycisku Update a usunięcia poprzez naciśnięcie przycisku Remove.

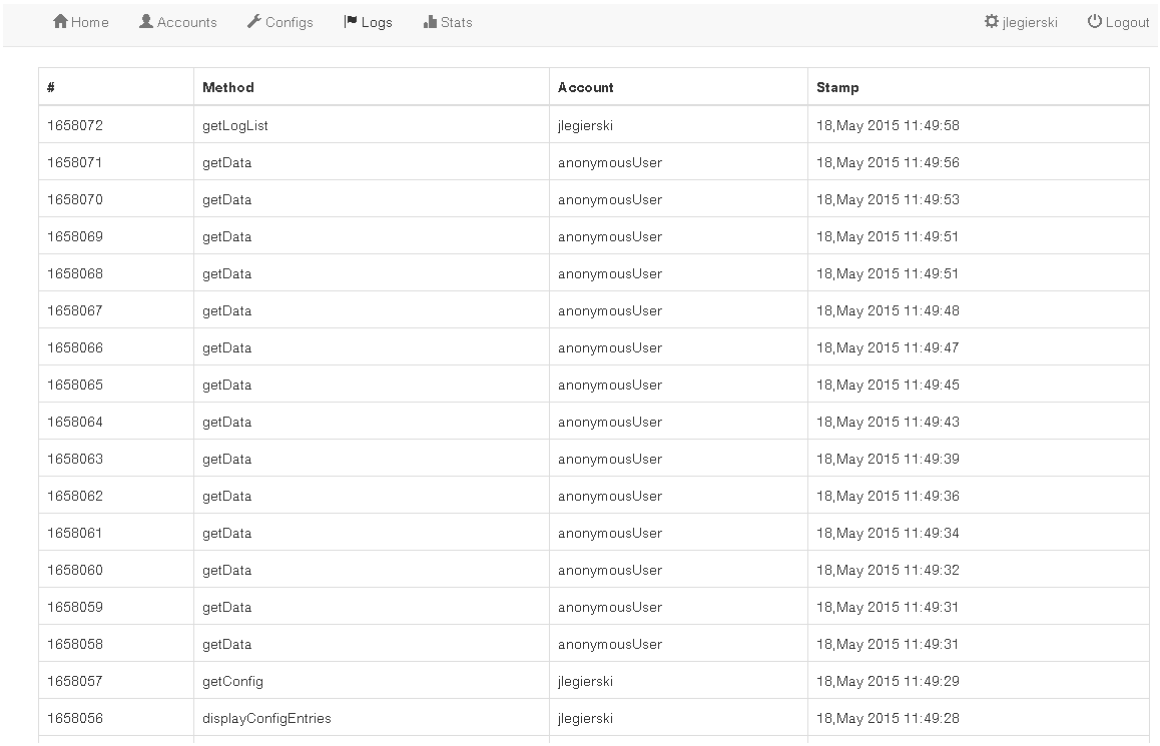


Rysunek 14 Parametry konfiguracyjne systemu - przykład

Zakładka Logi (Logs)

W zakładce Logs znajdują się informacje dotyczące aktywności użytkowników i administratorów systemu. Po wyświetleniu tej zakładki użytkownik Mundo Backend otrzymuje dostęp do informacji zawierających:

1) # – Id rekordu
2) Method – nazwie wywoływanej metody (funkcji)
3) Accounts – koncie użytkownika wywołującego w/w metodę (uwaga dla metod dostępnych dla użytkowników niezalogowanych wyświetlana nazwa użytkownika to anonymousUser)
4) Stamp – timestamp wykonania operacji

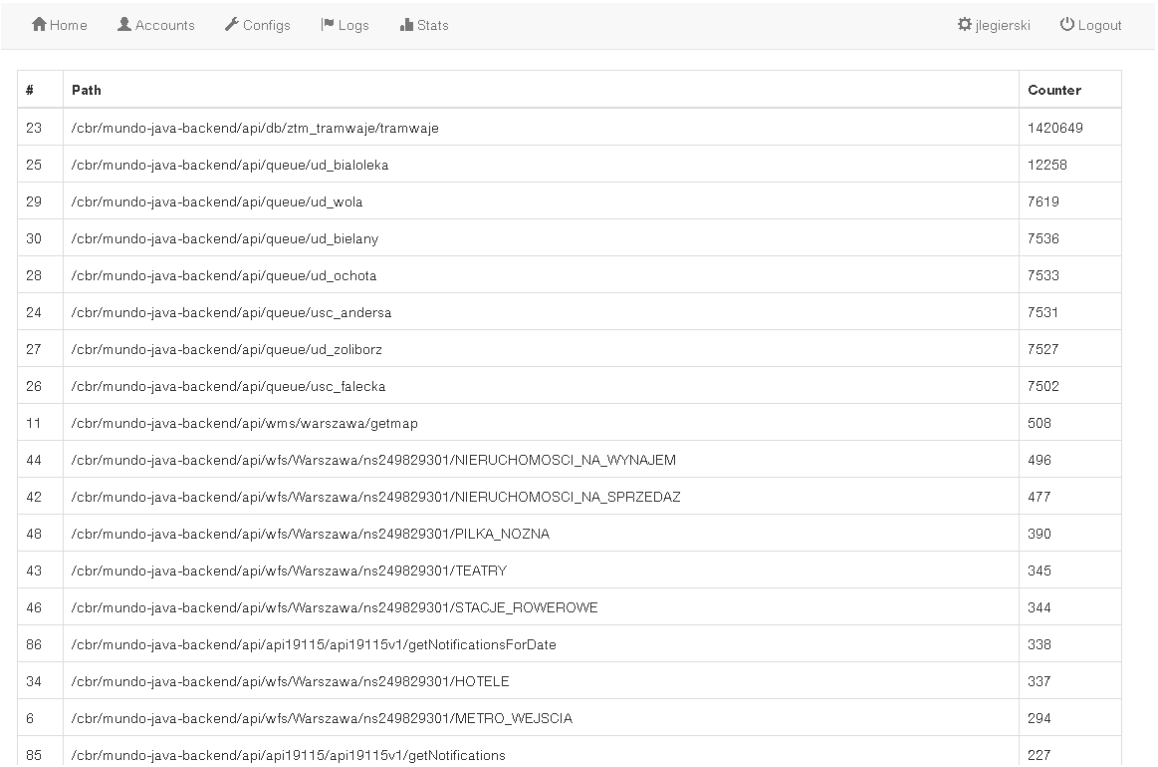


Rysunek 15 Logi systemu

Zakładka statystyki (Stats)

W zakładce Stats znajdują się informacje dotyczące statystyk użycia funkcji API eksponowanych przez system. Po wyświetleniu tej zakładki użytkownik Mundo Backend otrzymuje dostęp do informacji zawierających:

1) # – Id rekordu
2) Path – ścieżka wywoływanej funkcji (element url)
3) Counter – liczba wywołań w/w funkcji



#	Path	Counter
23	/cbr/mundo-java-backend/api/db/ztm_tramwaje/tramwaje	1420649
25	/cbr/mundo-java-backend/api/queue/ud_bialoleka	12258
29	/cbr/mundo-java-backend/api/queue/ud_wola	7619
30	/cbr/mundo-java-backend/api/queue/ud_bielany	7536
28	/cbr/mundo-java-backend/api/queue/ud_ochota	7533
24	/cbr/mundo-java-backend/api/queue/usc_andersa	7531
27	/cbr/mundo-java-backend/api/queue/ud_zoliborz	7527
26	/cbr/mundo-java-backend/api/queue/usc_falecka	7502
11	/cbr/mundo-java-backend/api/wms/warszawa/getmap	508
44	/cbr/mundo-java-backend/api/wfs/Warszawa/ns249829301/NIERUCHOMOSCI_NA_WYNAJEM	496
42	/cbr/mundo-java-backend/api/wfs/Warszawa/ns249829301/NIERUCHOMOSCI_NA_SPRZEDAZ	477
48	/cbr/mundo-java-backend/api/wfs/Warszawa/ns249829301/PILKA_NOZNA	390
43	/cbr/mundo-java-backend/api/wfs/Warszawa/ns249829301/TEATRY	345
46	/cbr/mundo-java-backend/api/wfs/Warszawa/ns249829301/STACJE_ROWEROWE	344
86	/cbr/mundo-java-backend/api/api19115/api19115v1/getNotificationsForDate	338
34	/cbr/mundo-java-backend/api/wfs/Warszawa/ns249829301/HOTELE	337
6	/cbr/mundo-java-backend/api/wfs/Warszawa/ns249829301/METRO_WEJSCIA	294
85	/cbr/mundo-java-backend/api/api19115/api19115v1/getNotifications	227

Rysunek 16 Statystyki z pracy systemu

1. Konfiguracja Data Server (CKAN)

Po uruchomieniu platformy CKAN należy dokonać jej konfiguracji zgodnie z dokumentacją Open Knowledge Foundation http://docs.ckan.org/en/latest/ poprzez założenie:

- kont administratorów i użytkowników,
- stworzenie organizacji i grup,
- przyporządkowanie użytkowników do określonych grup i organizacji

wg opisów zawartych w CKAN Sysadmin guide, oraz

- modyfikację wyglądu (opcjonalnie) wg opisów zawartych w CKAN Theming guide
- dołączenie do serwera e-mail (SMTP) wg opisów zawartych w CKAN Maintainer's guide

3.2 Konfiguracja zbiorów danych

1. Konfiguracja danych plikowych w systemie CKAN

Platforma wykorzystuje następujące rozszerzenia (extension) systemu CKAN które są wykorzystywane do składowania i ekspozycji plikowych danych statycznych:

Filestore – umożliwia załadowanie pliku do systemu CKAN.

DataPusher – dla plików posiadających strukturę tabelaryczną rozszerzenie to umożliwia wgranie pliku do bazy danych Datastore.

Datastore – przechowuje i eksponuje zawartość plików w bazie danych umożliwiając do nich dostęp poprzez Datastore API.

Pełną dokumentacje w/w elementów systemu zawiera dokumentacja CKAN:

http://docs.ckan.org/en/latest/

Przy czym cześć dotycząca wgrywania zbiorów danych jest dostępna w rozdziale User Guide:

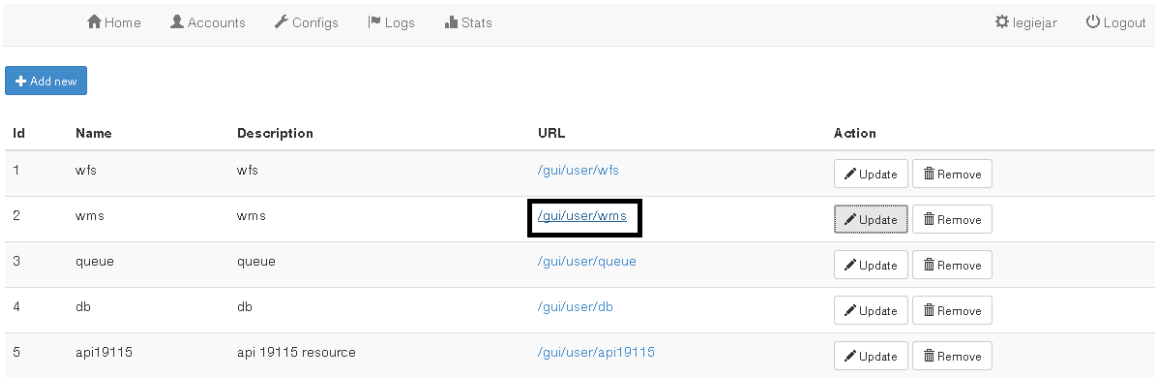
http://docs.ckan.org/en/latest/user-guide.html

2. Konfiguracja zbiorów typu WMS

Zbiory typu WMS to mapy rastrowe, które są przechowywane na serwerach Web Map System. Platforma MUNDO umożliwia udostępnienie tych zbiorów w prostszej dla developerów formie w porównaniu ze standardowym protokołem WMS.

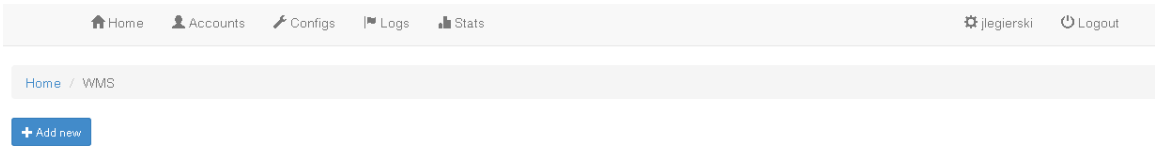
Konfiguracja po stronie Function Server

W celu skonfigurowania zbiorów danych związanych z mapami rastrowymi należy w zakładce Home kliknąć na URL elementu o nazwie wms



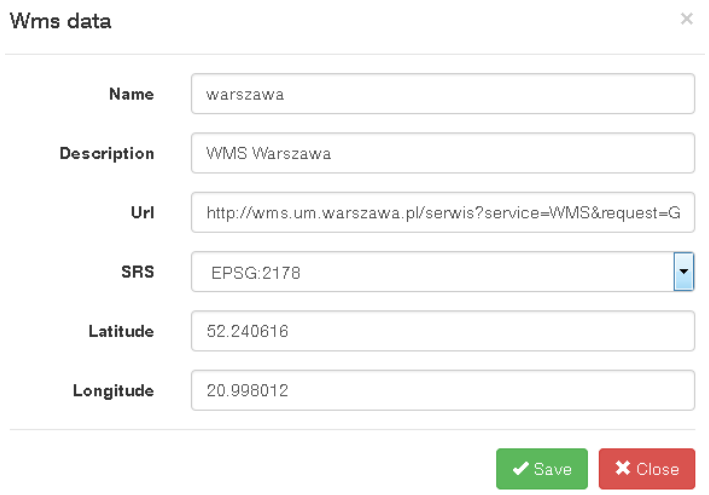
Rysunek 17 Konfiguracja zbioru typu WMS

A następnie kliknąć na przycisk Add new



Rysunek 18 Konfiguracja zbioru typu WMS

W kolejnym kroku należy podać: nazwę, opis i url dostępu do usługi WMS (GetCapabilities) np.:

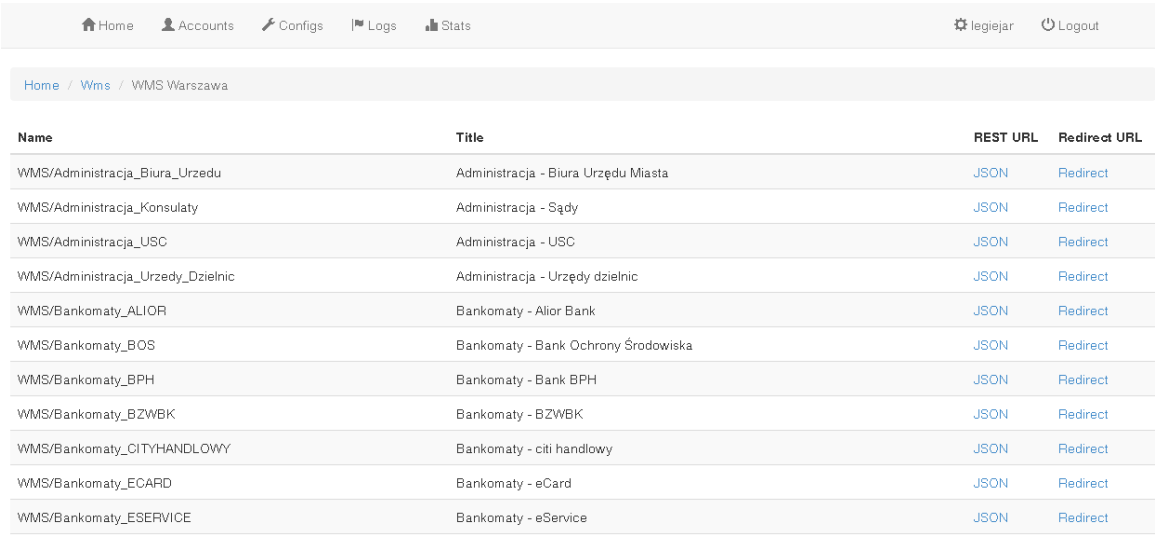


Rysunek 19 Konfiguracja zbioru typu WMS

Podając odpowiedni identyfikator układu współrzędnych (SRS) i ewentualnie współrzędne punktu centralnego mapy jeśli te są inne niż zawarte w zakładce Configs platformy.

Rysunek 20 Konfiguracja zbioru typu WMS

W przypadku poprawnej konfiguracji po kliknięciu na nazwę zdefiniowanego zasobu WMS (kolumna Name) otrzymamy listę wszystkich warstw, jakie posiada zdefiniowany serwer WMS.

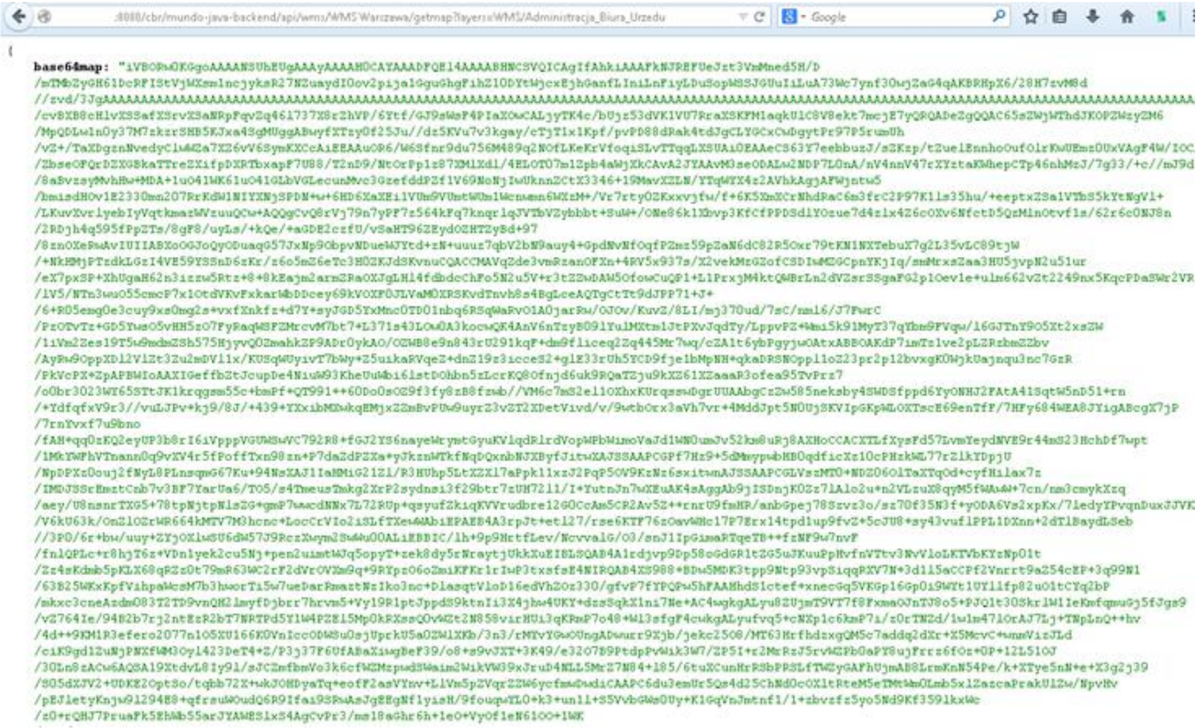


Rysunek 21 Zbiór typu WMS

Po kliknięciu na link JSON (kolumna REST URL) wywołujemy uproszczone API (Easy WMS API) – i w odpowiedzi otrzymujemy obrazek opakowany w JSON kodowany w base64 jako obiekt base64map.

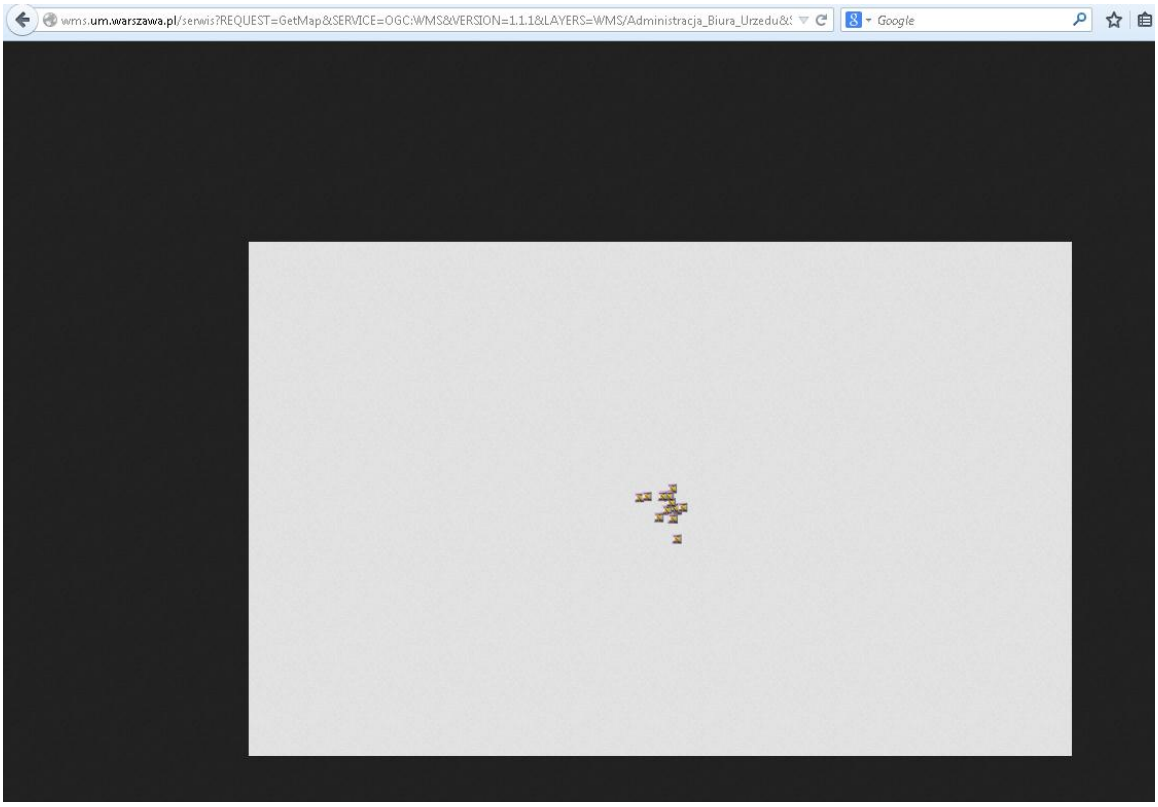
Np.:

http://10.10.10.11:8080/cbr/mundo-java-backend/api/wms/warszawa/getmap?layers=WMS/Administracja_Biura_Urzedu



Rysunek 22 Wywołanie API do zbioru typu WMS

Po kliknięciu na link Redirect (kolumna Redirect URL) platforma dokonuje konwersji naszego wywołania API i po dodaniu domyślnych parametrów do url, przekierowania na adres serwera WMS



Rysunek 23 Wywołanie API do zbioru typu WMS

Konfiguracja po stronie CKAN

Do konfiguracji zbioru danych typu WMS po stronie Data Server (CKAN) służy rozszerzenie ckanext-wmsstore.

Do dodania i modyfikacji zasobów (resource) po stronie CKAN służą zaimplementowana w w/w rozszerzeniu funkcja API wmsstore_create

1) Zbiór danych z wszystkimi warstwami

Utworzenie pojedynczego zbioru danych (dataset) zawierającego wszystkie warstwy w formie zasobów (resources)

http://adres_ip_ds/api/3/action/wmsstore_create?name=**nazwa_dataset**&wms_url=**http://adres_ip_fs:8080/cbr/mundo-java-backend/api/wms/nazwa_wms**&**capabilities=true**

gdzie:

adres_ip_ds – adres IP Data Server (CKAN)

name – nazwa tworzonego zbioru danych

adres_ip_fs - adres IP Function Server

capabilities=true – opcja umożliwiająca automatyczne utworzenie całego zbioru

wms_url – url dostępu do API w Function Server

np.:

http://10.10.10.10/api/3/action/wmsstore_create?name=warszawa&wms_url=http://10.10.10.11:8080/cbr/mundo-java-backend/api/wms/warszawa&capabilities=true

Zwracany jest identyfikator utworzonego zbioru danych.

{

*"help": "Adds a new Wmsstore.\n\n **Params:**\n :name [String]: Wmsstore name.\n :wms_url [String]: Wms url.\n :capabilities [Boolean]: If URL contains capabilities (Default false)\n :package_id [String] Existing package id.\n\n **Results:**\n\n :returns: The newly created data object.\n :rtype: dictionary\n ",*

"success": true,

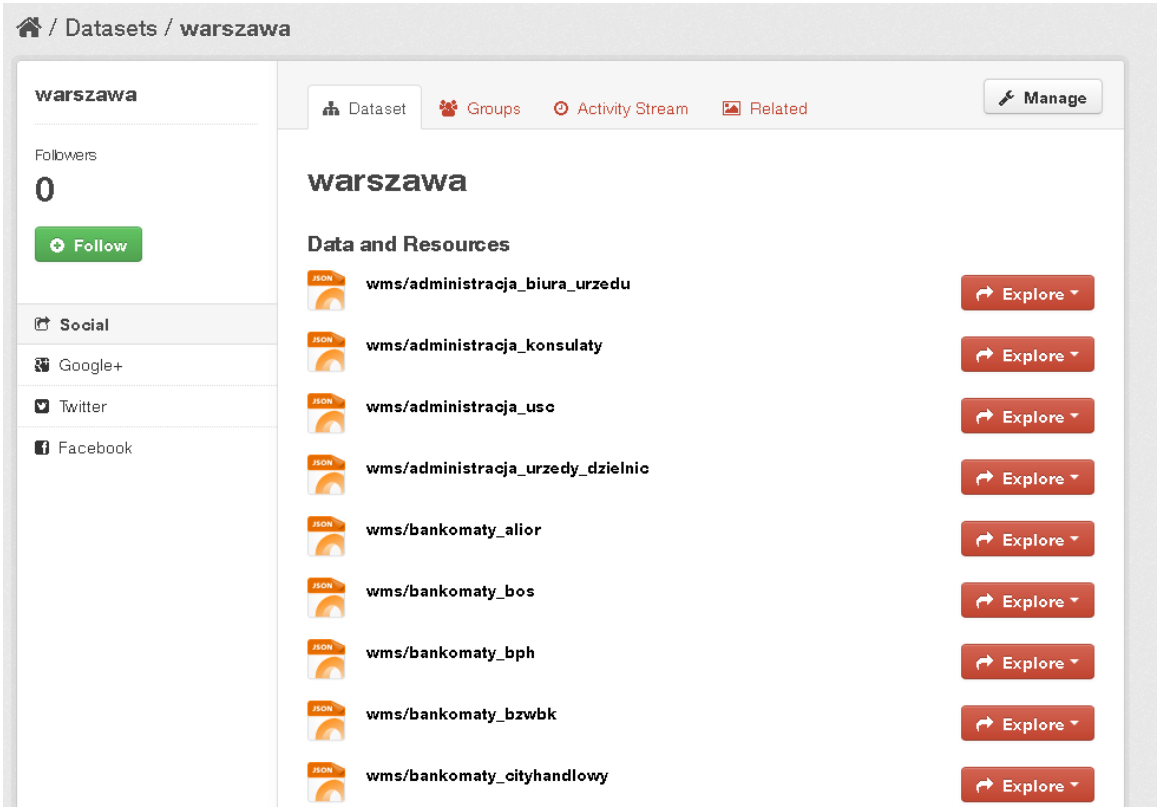
"result": {

"package_id": "828c9932-de4f-4b7b-83b5-d57eca58404d"

}

}

Oraz tworzona odpowiednia struktura danych w CKAN:



Rysunek 24 Zbiór typu WMS w Data Server (CKAN)

2) Tworzenie zbioru danych z wybranymi warstwami

Utworzenie pojedynczego zbioru danych (dataset) zawierającego wybrane warstwy w formie zasobów (resources) wymaga wywołania url:

http://adres_ip_ds/api/3/action/wmsstore_create?name=**nazwa_zbioru**&wms_url=**http://adres_ip_fs:8080/cbr/mundo-java-backend/api/wms/nazwa_wms?layers=nazwa_warstwy**

gdzie:

adres_ip_ds – adres IP Data Server (CKAN),

name – nazwa tworzonego zbioru danych,

adres_ip_fs – adres IP Function Server ,

layers – nazwa warstwy,

wms_url – url dostępu do API w Function Server.

np:

http://10.10.10.10/api/3/action/wmsstore_create?name=mapyadministracja&wms_url=http://10.10.10.11:8080/cbr/mundo-java-backend/api/wms/warszawa/getmap?layers=WMS/Administracja_Biura_Urzedu

tworzy Dataset o nazwie mapyadministracja z jednym resource WMS/Administracja_Biura_Urzedu i zwraca jego id:

{

*"help": "Adds a new Wmsstore.\n\n **Params:**\n :name [String]: Wmsstore name.\n :wms_url [String]: Wms url.\n :capabilities [Boolean]: If URL contains capabilities (Default false)\n :package_id [String] Existing package id.\n\n **Results:**\n\n :returns: The newly created data object.\n :rtype: dictionary\n ",*

"success": true,

"result": {

"package_id": "111245f5-4863-412d-92aa-307536850af5"

}

}

3) Dodanie zasobu do istniejącego zbioru danych

Dodanie zasobu do istniejącego zbioru danych wymaga wywołania url w postaci

http://adres_ip_ds/api/3/action/wmsstore_create?name=**nazwa_zbioru**&wms_url=**http://adres_ip_fs:8080/cbr/mundo-java-backend/api/wms/nawa_wms/getmap?**layers=**nazwa_warstwy**&package_id=**id_zbioru_danych**

adres_ip_ds – adres IP Data Server (CKAN)

name – nazwa tworzonego zasobu

adres_ip_fs – adres IP Function Server

layers – nazwa warstwy

package_id – identyfikator zbioru_danych

wms_url – url dostępu do API w Function Server

np.:

Dodanie resource do istniejącego datasetu o package_id 111245f5-4863-412d-92aa-307536850af5

http://10.10.10.10/api/3/action/wmsstore_create?name=administracja_usc&wms_url=http://10.10.10.11:8080/cbr/mundo-java-backend/api/wms/warszawa/getmap?layers=WMS/Administracja_USC&package_id=111245f5-4863-412d-92aa-307536850af5

Zwraca odpowiedź:

{

*"help": "Adds a new Wmsstore.\n\n **Params:**\n :name [String]: Wmsstore name.\n :wms_url [String]: Wms url.\n :capabilities [Boolean]: If URL contains capabilities (Default false)\n :package_id [String] Existing package id.\n\n **Results:**\n\n :returns: The newly created data object.\n :rtype: dictionary\n ",*

"success": true,

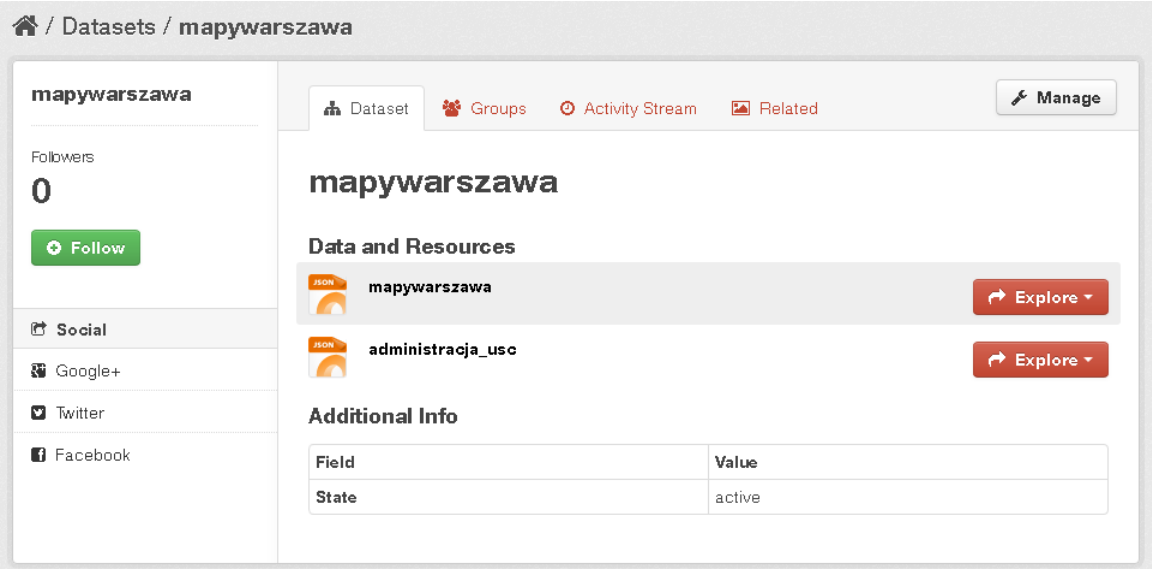
"result": {

"package_id": "111245f5-4863-412d-92aa-307536850af5"

}

}

Oraz dodaje zasób w CKAN:



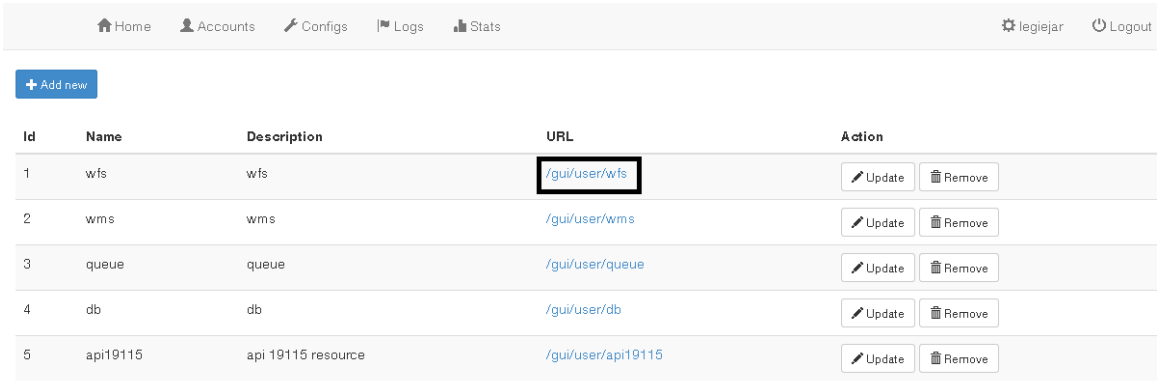
Rysunek 25 Zbiór typu WMS w Data Server (CKAN)

3. Konfiguracja elementu typu WFS

Zbiory typu WFS to mapy wektorowe, które są przechowywane na serwerach Web Feature Service. Platforma MUNDO umożliwia udostępnienie tych zbiorów programistom w prostszej składniowo dla developerów formie w porównaniu ze standardowym protokołem WFS.

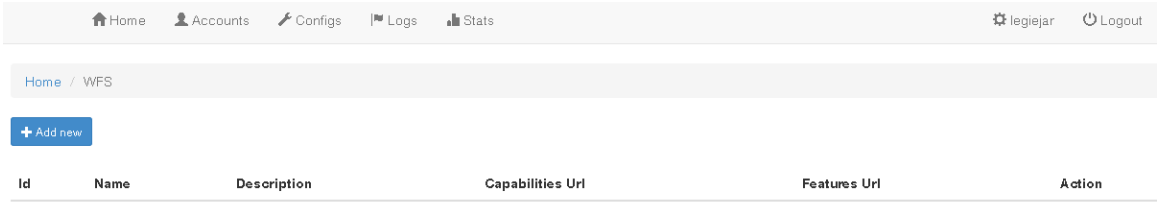
Konfiguracja po stronie Function Server

W celu skonfigurowania zbiorów danych związanych z mapami rastrowymi należy w zakładce Home kliknąć na URL elementu o nazwie wfs



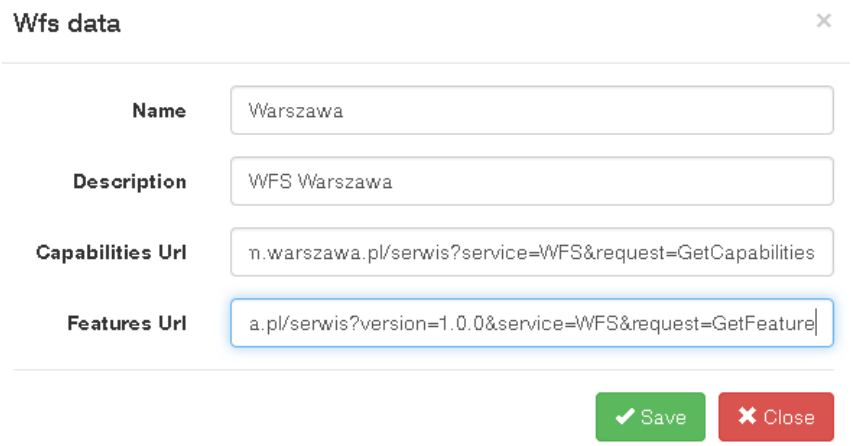
Rysunek 26 Konfiguracja zbioru typu WFS

A następnie kliknąć na przycisk Add new

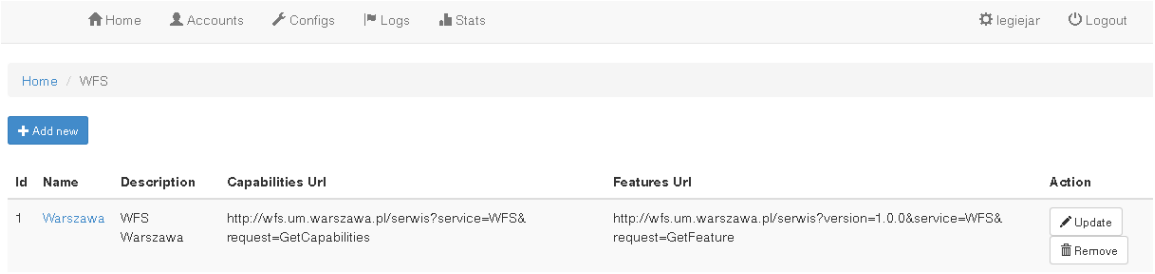


Rysunek 27 Konfiguracja zbioru typu WFS

W kolejnym kroku należy podać nazwę , opis i urle dostępu do usługi WFS (GetCapabilities GetFeature) np.:

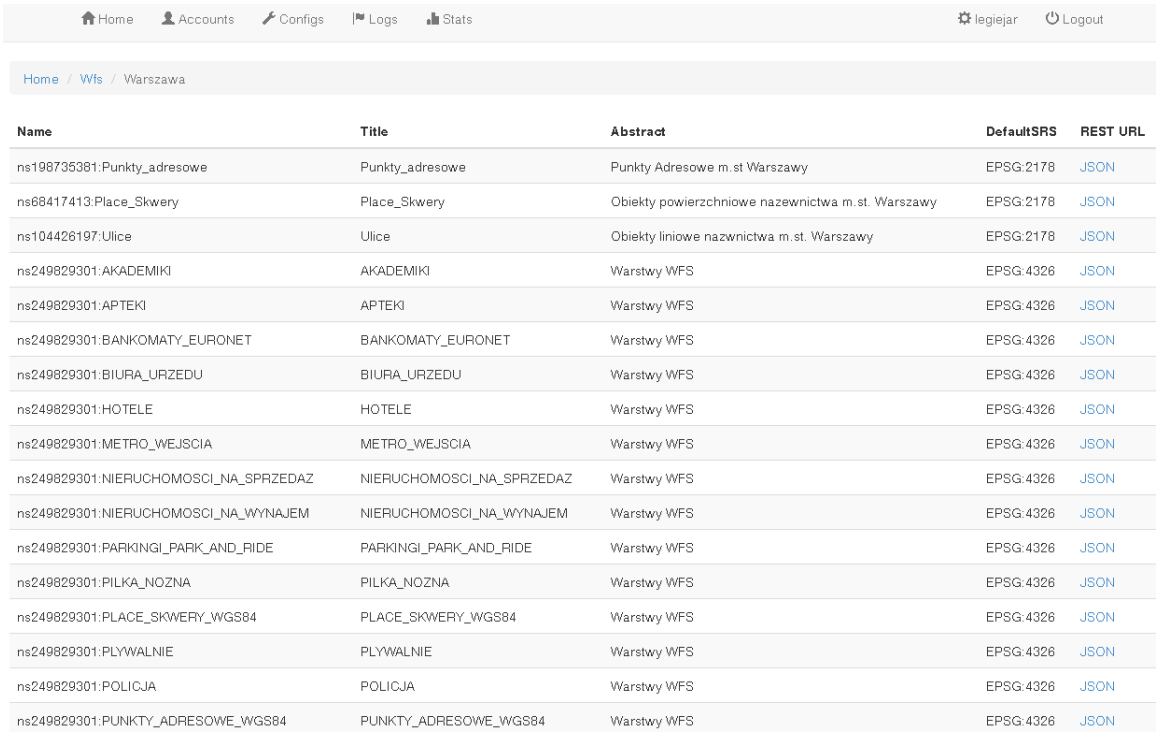


Rysunek 28 Konfiguracja zbioru typu WFS



Rysunek 29 Konfiguracja zbioru typu WFS

W przypadku poprawnej konfiguracji po kliknięciu na nazwę zdefiniowanego zasobu WFS (kolumna Name) otrzymamy listę wszystkich warstw jakie zawiera dany serwer WFS.



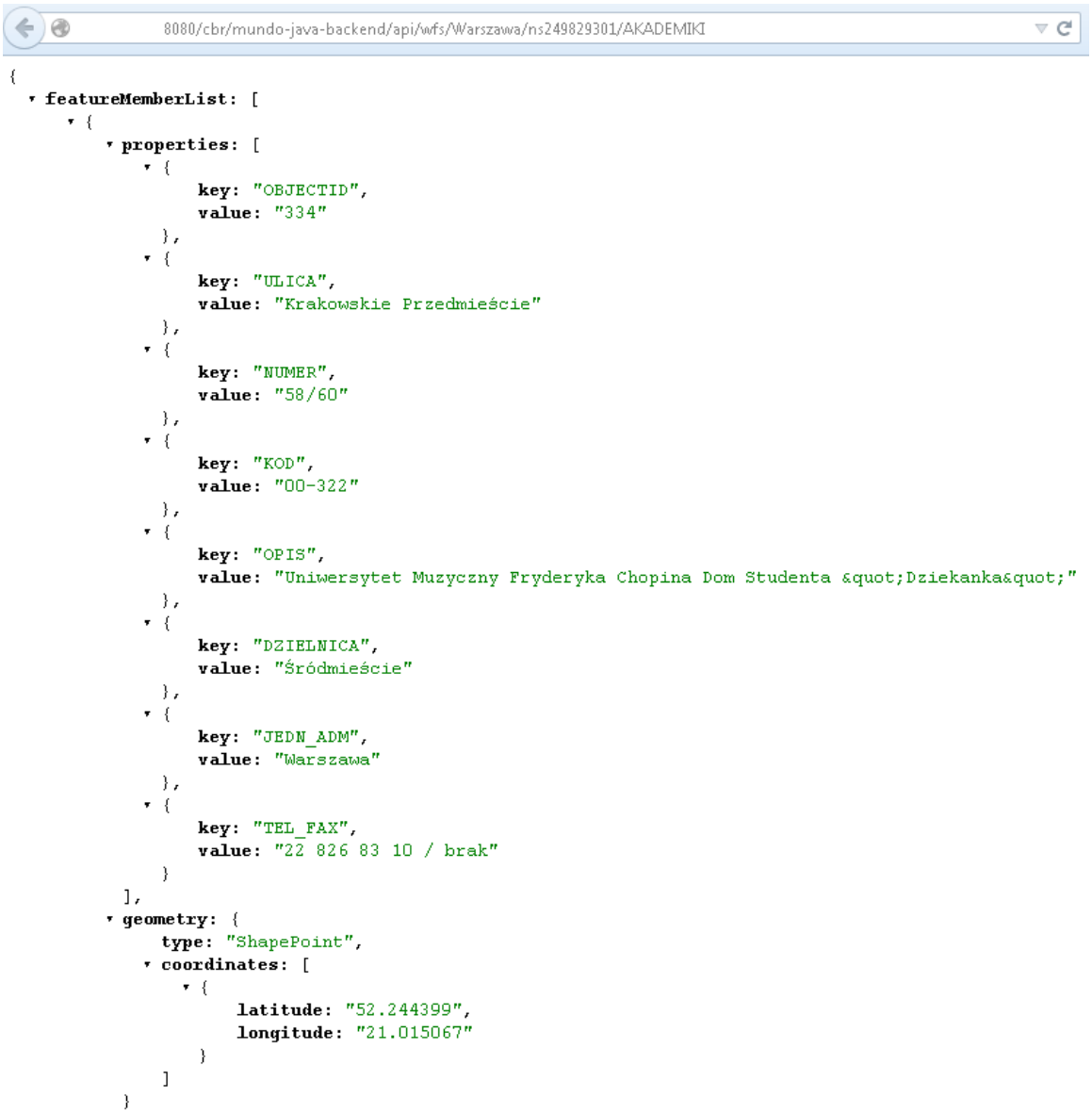
Name	Title	Abstract	DefaultSRS	REST URL
ns198735381:Punkty_adresowe	Punkty_adresowe	Punkty Adresowe m.st Warszawy	EPSG:2178	JSON
ns68417413:Place_Skwery	Place_Skwery	Obiekty powierzchniowe nazewnictwa m.st. Warszawy	EPSG:2178	JSON
ns104426197:Ulice	Ulice	Obiekty liniowe nazwnictwa m.st. Warszawy	EPSG:2178	JSON
ns249829301:AKADEMIKI	AKADEMIKI	Warstwy WFS	EPSG:4326	JSON
ns249829301:APTEKI	APTEKI	Warstwy WFS	EPSG:4326	JSON
ns249829301:BANKOMATY_EURONET	BANKOMATY_EURONET	Warstwy WFS	EPSG:4326	JSON
ns249829301:BIURA_URZEDU	BIURA_URZEDU	Warstwy WFS	EPSG:4326	JSON
ns249829301:HOTELE	HOTELE	Warstwy WFS	EPSG:4326	JSON
ns249829301:METRO_WEJSCIA	METRO_WEJSCIA	Warstwy WFS	EPSG:4326	JSON
ns249829301:NIERUCHOMOSCI_NA_SPRZEDAZ	NIERUCHOMOSCI_NA_SPRZEDAZ	Warstwy WFS	EPSG:4326	JSON
ns249829301:NIERUCHOMOSCI_NA_WYNAJEM	NIERUCHOMOSCI_NA_WYNAJEM	Warstwy WFS	EPSG:4326	JSON
ns249829301:PARKINGI_PARK_AND_RIDE	PARKINGI_PARK_AND_RIDE	Warstwy WFS	EPSG:4326	JSON
ns249829301:PILKA_NOZNA	PILKA_NOZNA	Warstwy WFS	EPSG:4326	JSON
ns249829301:PLACE_SKWERY_WGS84	PLACE_SKWERY_WGS84	Warstwy WFS	EPSG:4326	JSON
ns249829301:PLYWALNIE	PLYWALNIE	Warstwy WFS	EPSG:4326	JSON
ns249829301:POLICJA	POLICJA	Warstwy WFS	EPSG:4326	JSON
ns249829301:PUNKTY_ADRESOWE_WGS84	PUNKTY_ADRESOWE_WGS84	Warstwy WFS	EPSG:4326	JSON

Rysunek 30 Zbiór danych typu WFS

Po kliknięciu na link JSON (kolumna REST URL) wywołujemy uproszczone API (Easy WFS API) zwracające mapę wektorową.

Np.:

http://10.10.10.11:8080/cbr/mundo-java-backend/api/wfs/Warszawa/ns249829301/AKADEMIKI



Rysunek 31 Zbiór danych typu WFS – wywołanie API

Konfiguracja po stronie CKAN

Do konfiguracji zbioru danych typu WFS po stronie Data Server (CKAN) służy rozszerzenie ckanext-wfsstore.

Do dodania i modyfikacji zbiorów danych (data sets) i zasobów (resources) po stronie CKAN służą zaimplementowana w w/w rozszerzeniu funkcja API wfsstore_create.

Zbiór danych z wszystkimi warstwami

Utworzenie pojedynczego zbioru danych (dataset) zawierającego wszystkie warstwy w formie zasobów (resources):

http://adres_ip_ds/api/3/action/wfsstore_create?name=**nazwa_dataset**&wfs_url=**http://adres_ip_fs:8080/cbr/mundo-java-backend/api/wfs/nazwa_wfs**&**capabilities=true**

gdzie:

adres_ip_ds – adres IP Data Server (CKAN),

name – nazwa tworzonego zbioru danych,

adres_ip_fs - adres IP Function Server,

capabilities=true – opcja umożliwiająca automatyczne utworzenie całego zbioru,

wfs_url – url dostępu do API w Function Server.

np.:

http://10.10.10.10/api/3/action/wfsstore_create?name=wfswarszawa&wfs_url=http://10.10.10.11:8080/cbr/mundo-java-backend/api/wfs/Warszawa&capabilities=true

zwraca id utworzonego zbioru danych

{

*"help": "Adds a new Wfsstore.\n\n **Params:**\n :name [String]: Wfsstore name.\n :wfs_url [String]: Wfs url.\n :capabilities [Boolean]: If URL contains capabilities (Default false)\n :package_id [String] Existing package id.\n\n **Results:**\n\n :returns: The newly created data object.\n :rtype: dictionary\n ",*

"success": true,

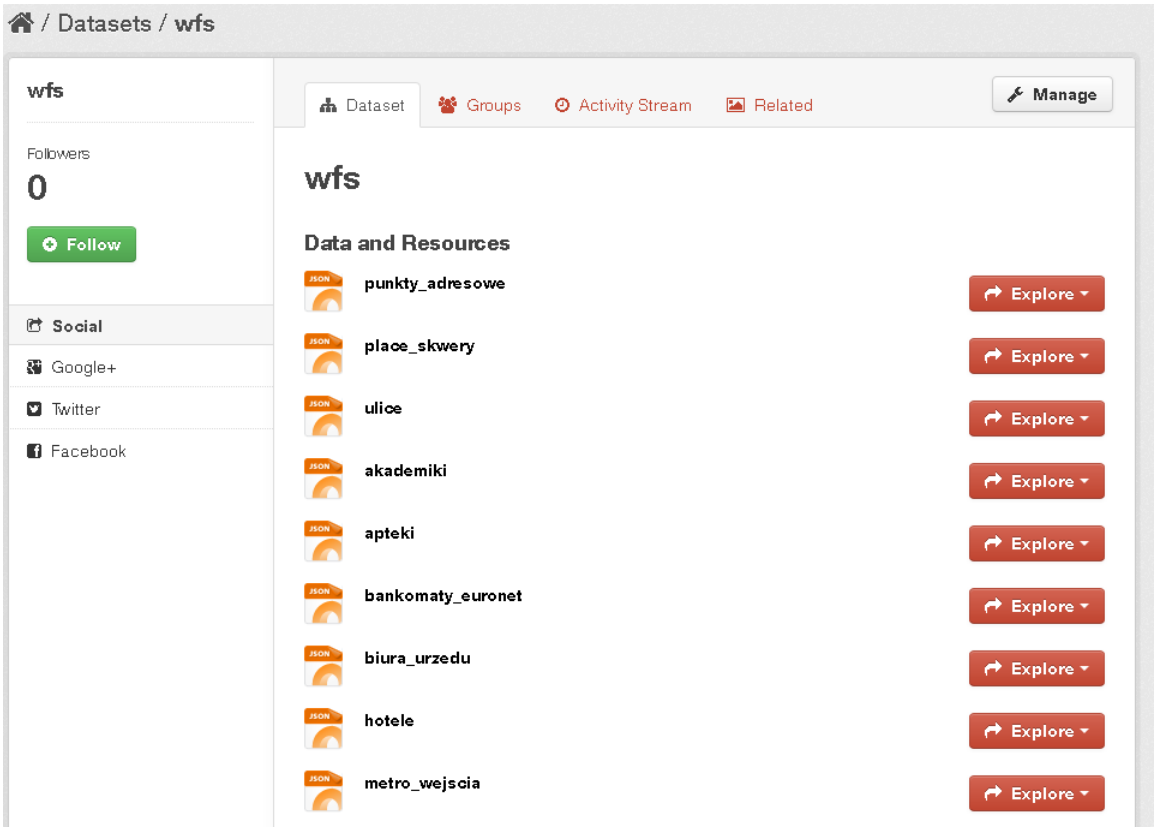
"result": {

"package_id": "ea6e800b-7ce8-4a75-88d3-7e80bb4e677d"

}

}

oraz tworzy zbiór po stronie CKAN.



Rysunek 32 Konfiguracja zbioru typu WFS po stronie Data Server (CKAN)

Tworzenie zbioru danych z wybranymi warstwami

Utworzenie pojedynczego zbioru danych (dataset) zawierającego wybrane warstwy w formie zasobów (resources)

http://adres_ip_ds/api/3/action/wmsstore_create?name=**nazwa_dataset**&wfs_url=**http://adres_ip_fs:8080/cbr/mundo-java-backend/api/wfs/nazwa_wfs/nazwa_warstwy**

gdzie:

adres_ip_ds – adres IP Data Server (CKAN)

name – nazwa tworzonego zbioru danych

adres_ip_fs – adres IP Function Server

wfs_url – url dostępu do API w Function Server

np:

http://10.10.10.10/api/3/action/wfsstore_create?name=administracja&wfs_url=http://10.10.10.11:8080/cbr/mundo-java-backend/api/wfs/Warszawa/ns249829301/BIURA_URZEDU

Tworzy zbiór danych nazwie administracja z jednym resource ns249829301/BIURA_URZEDU i zwraca jego id

{

"help": "Adds a new Wfsstore.\n\n **Params:**\n :name [String]: Wfsstore name.\n :wfs_url [String]: Wfs url.\n :capabilities [Boolean]: If URL contains capabilities (Default false)\n :package_id [String] Existing package id.\n\n **Results:**\n\n :returns: The newly created data object.\n :rtype: dictionary\n ",

"success": true,

"result": {

"package_id": "fa84d033-1dc7-453a-acc6-c0239c6a600c"

}

}

Dodanie zasobu do istniejącego datasetu

W celu dodania zasobu do istniejącego datasetu należy wywołać url w formie:

http://adres_ip_ds/api/3/action/wfsstore_create?name=**nazwa_zasobu**&wfs_url=**http://adres_ip_fs:8080/cbr/mundo-java-backend/api/wfs/nazwa_wfs/nazwa_warstwy**&package_id=**id_zbioru_danych**

gdzie:

adres_ip_ds – adres IP Data Server (CKAN),

name – nazwa tworzonego zasobu,

adres_ip_fs – adres IP Function Server,

layers – nazwa warstwy,

package_id – identyfikator zbioru_danych,

wfs_url – url dostępu do API w Function Server.

np.:

Dodanie resource do istniejącego datasetu o package_id fa84d033-1dc7-453a-acc6-c0239c6a600c

http://10.10.10.10/api/3/action/wfsstore_create?name=policja&wfs_url=http://10.10.10.11:8080/cbr/mundo-java-backend/api/wfs/Warszawa/ns249829301/POLICJA&package_id=fa84d033-1dc7-453a-acc6-c0239c6a600c

zwraca odpowiedź:

{

*"help": "Adds a new Wfsstore.\n\n **Params:**\n :name [String]: Wfsstore name.\n :wfs_url [String]: Wfs url.\n :capabilities [Boolean]: If URL contains capabilities (Default false)\n :package_id [String] Existing package id.\n\n **Results:**\n\n :returns: The newly created data object.\n :rtype: dictionary\n ",*

"success": true,

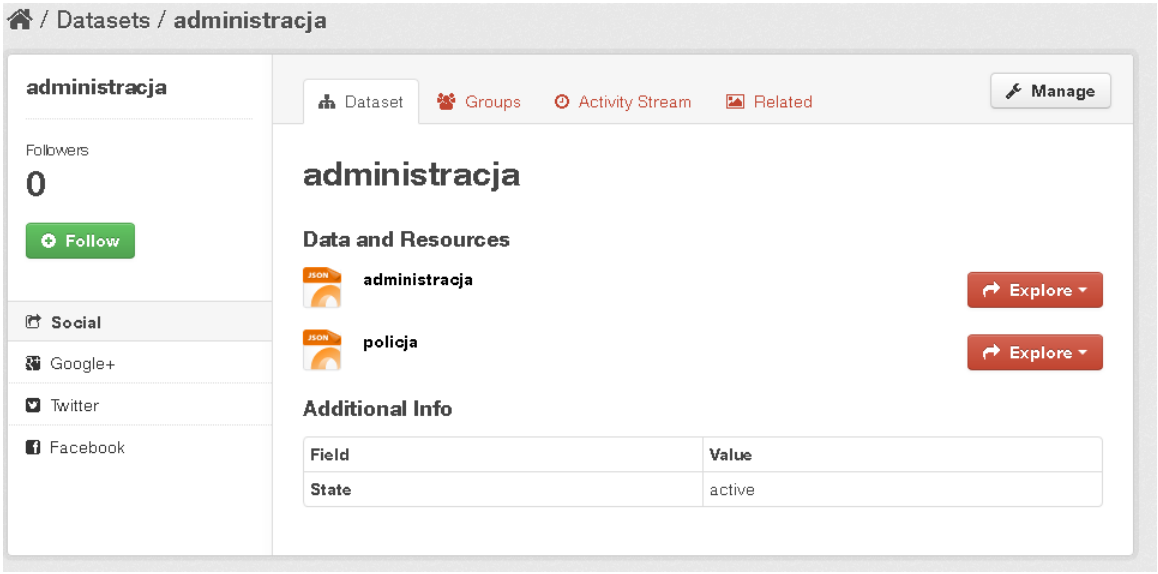
"result": {

"package_id": "fa84d033-1dc7-453a-acc6-c0239c6a600c"

}

}

Oraz modyfikuje zbiór po stronie Data Server (CKAN):



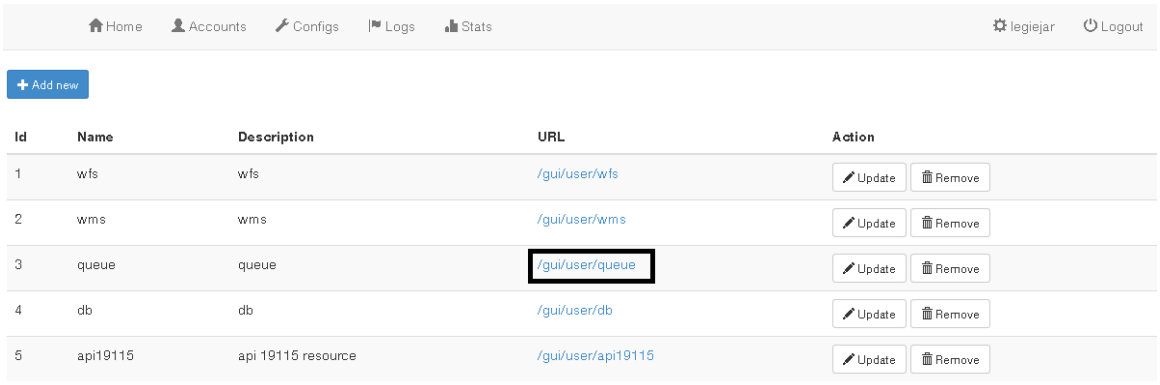
Rysunek 33 Konfiguracja zbioru typu WFS po stronie Data Server (CKAN)

4. Konfiguracja elementu typu Queue (Web Services)

Zbiory typu Queue, zawierają informacje z systemów kolejkowych, które są udostępniane jako usługi sieciowe (Web Services) w standardzie SOA/SOAP i transponowane przez platformę do modelu zasobowego ROA/REST.

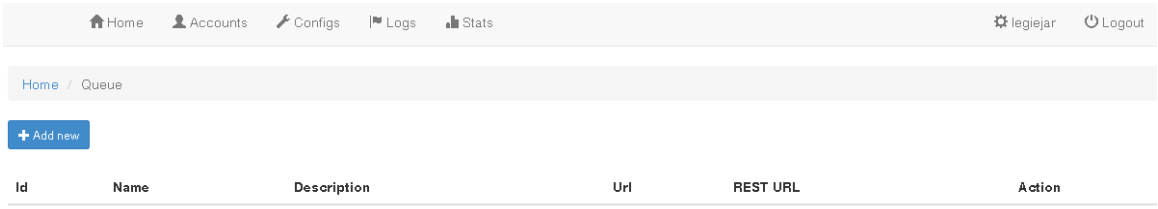
Konfiguracja po stronie Function Server

W celu skonfigurowania zbiorów danych związanych systemami kolejkowymi, należy w zakładce Home kliknąć na URL elementu o nazwie queue.



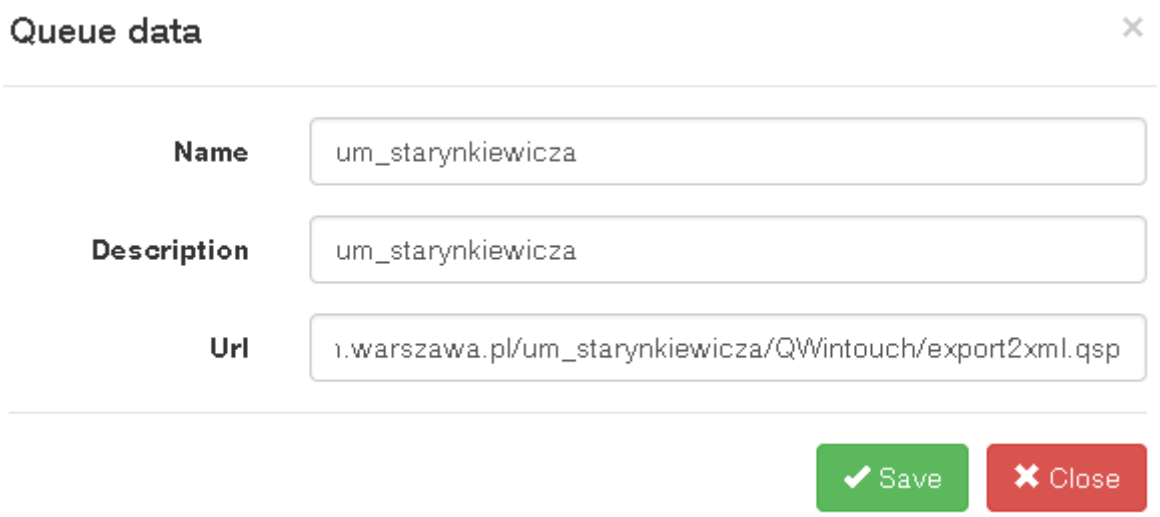
Rysunek 34 Konfiguracja zbioru typu Queue

A następnie kliknąć na przycisk Add new

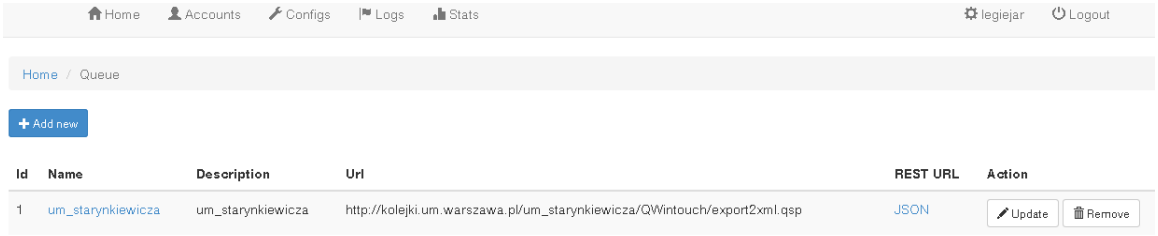


Rysunek 35 Konfiguracja zbioru typu Queue

W kolejnym kroku, należy podać nazwę, opis i url dostępu do usługi np.:

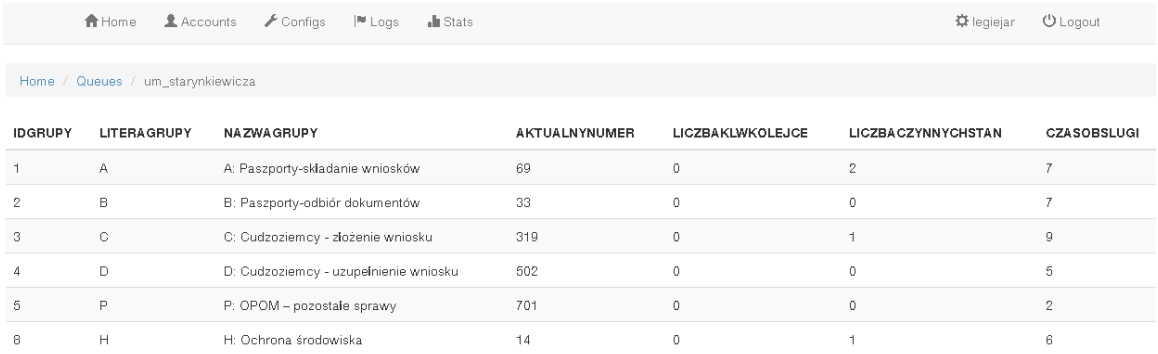


Rysunek 36 Konfiguracja zbioru typu Queue



Rysunek 37 Konfiguracja zbioru typu Queue

W przypadku poprawnej konfiguracji, po kliknięciu na nazwę zdefiniowanego zasobu queue (kolumna Name) otrzymamy wizualizację stanu kolejek w monitorowanym systemie.



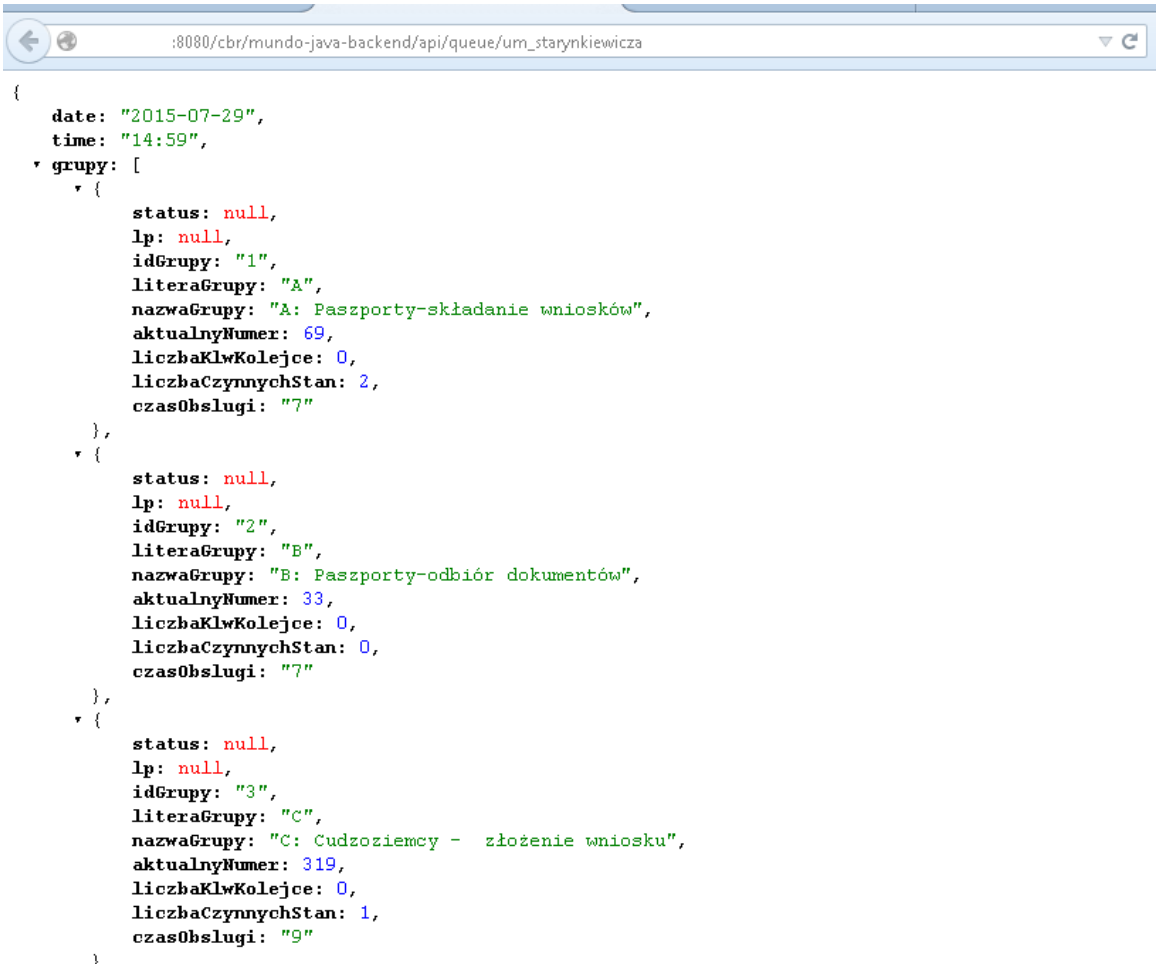
IDGRUPY	LITERAGRUPY	NAZWAGRUPY	AKTUALNYNUMER	LICZBAKLWKOLEJCE	LICZBACZYNNYCHSTAN	CZASOBSLUGI
1	A	A: Paszporty-składanie wniosków	69	0	2	7
2	B	B: Paszporty-odbiór dokumentów	33	0	0	7
3	C	C: Cudzoziemcy - złożenie wniosku	319	0	1	9
4	D	D: Cudzoziemcy - uzupełnienie wniosku	502	0	0	5
5	P	P: OPOM – pozostałe sprawy	701	0	0	2
8	H	H: Ochrona środowiska	14	0	1	6

Rysunek 38 Zbiór typu Queue - wizualizacja

Po kliknięciu na link JSON (kolumna REST URL), wywołujemy API zwracające stan kolejki.

Np.:

http://10.10.10.11:8080/cbr/mundo-java-backend/api/queue/um_starynkiewicza



Rysunek 39 Zbiór typu Queue – wywołanie API

Konfiguracja po stronie CKAN

Do konfiguracji zbioru danych typu Queue po stronie Data Server (CKAN) służy rozszerzenie ckanext-wsstore.

Do dodania i modyfikacji zasobów (resource) po stronie CKAN służą zaimplementowana w w/w rozszerzeniu funkcja API wsstore_create.

Utworzenie zbioru danych z wybranymi zasobami

Utworzenie pojedynczego zbioru danych (dataset) zawierającego wybrane zasoby (kolejki) następuje poprzez wywołanie następującego url:

http://adres_ip_ds/api/3/action/wsstore_create?name=**nazwa_dataset**&ws_url=**http://adres_ip_fs:8080/cbr/mundo-java-backend/api/queue/nazwa_kolejki**

gdzie:

adres_ip_ds – adres IP Data Server (CKAN),

name – nazwa tworzonego zbioru danych,

adres_ip_fs – adres IP Function Server,

np:

http://10.10.10.10/api/3/action/wsstore_create?name=kolejki&ws_url=http://10.10.10.11:8080/cbr/mundo-java-backend/api/queue/um_starynkiewicza

tworzy dataset o nazwie kolejki z jednym resource queue/um_starynkiewicza i zwraca jego id

{

*"help": "Adds a new Wsstore.\n \n **Params:**\n :name [String]: Dbstore name.\n :ws_url [String]: Ws url.\n :package_id [String] Existing package id. \n\n **Results:**\n\n :returns: The newly created data object.\n :rtype: dictionary\n ",*

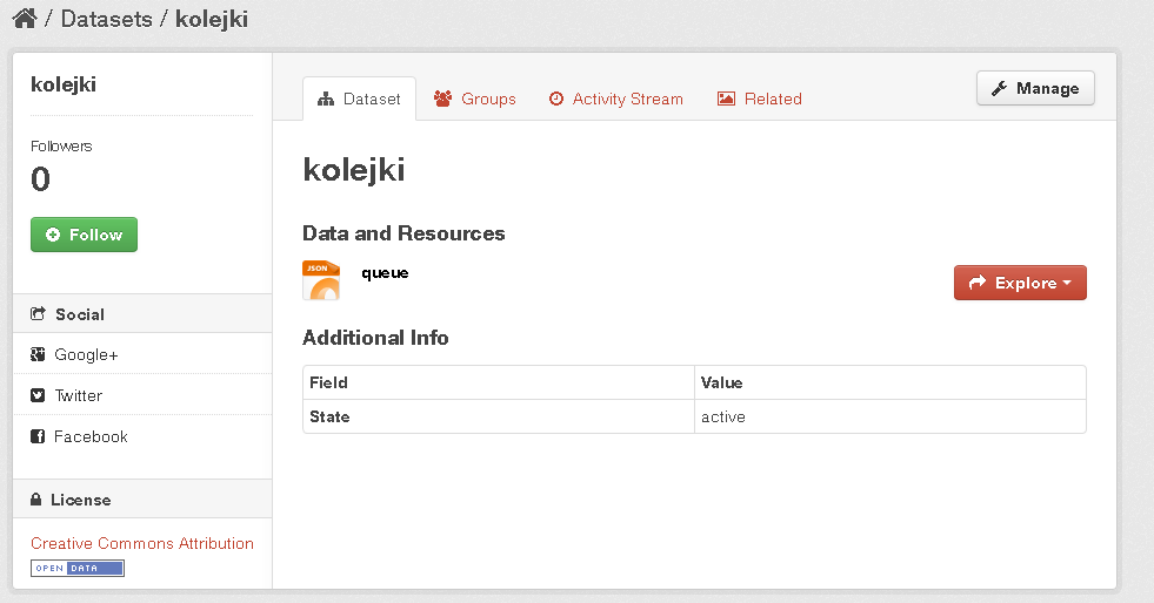
"success": true,

"result": {

"package_id": "72bc5b69-ca61-4955-8ef7-fe3be79f3750"

}

tworząc jednocześnie odpowiedni zbiór danych w CKAN.



Rysunek 40 Zbiór typu Queue – po stronie Data Server (CKAN)

Dodanie zasobu do istniejącego datasetu

W celu dodania zasobu do istniejącego datasetu konieczne jest wywołanie url w postaci:

http://adres_ip_ds/api/3/action/wsstore_create?name=**nazwa_zasobu**&ws_url=**http://adres_ip_fs:8080/cbr/mundo-java-backend/api/api/queue/nazwa_kolejki**&package_id=**id_zbioru_danych**

przy czym:

adres_ip_ds – adres IP Data Server (CKAN),

name – nazwa tworzonego zasobu,

adres_ip_fs – adres IP Function Server,

layers – nazwa warstwy,

package_id – identyfikator zbioru_danych,

ws_url – url dostępu do API w Function Server

np.:

Dodanie resource do istniejącego datasetu o package_id fa84d033-1dc7-453a-acc6-c0239c6a600c

http://10.10.10.10/api/3/action/wsstore_create?name=andersa&ws_url=http://10.10.10.11:8080/cbr/mundo-java-backend/api/queue/usc_andersa&package_id=72bc5b69-ca61-4955-8ef7-fe3be79f3750

skutkuje odpowiedzią:

{

*"help": "Adds a new Wsstore.\n \n **Params:**\n :name [String]: Dbstore name.\n :ws_url [String]: Ws url.\n :package_id [String] Existing package id. \n\n **Results:**\n\n :returns: The newly created data object.\n :rtype: dictionary\n ",*

"success": true,

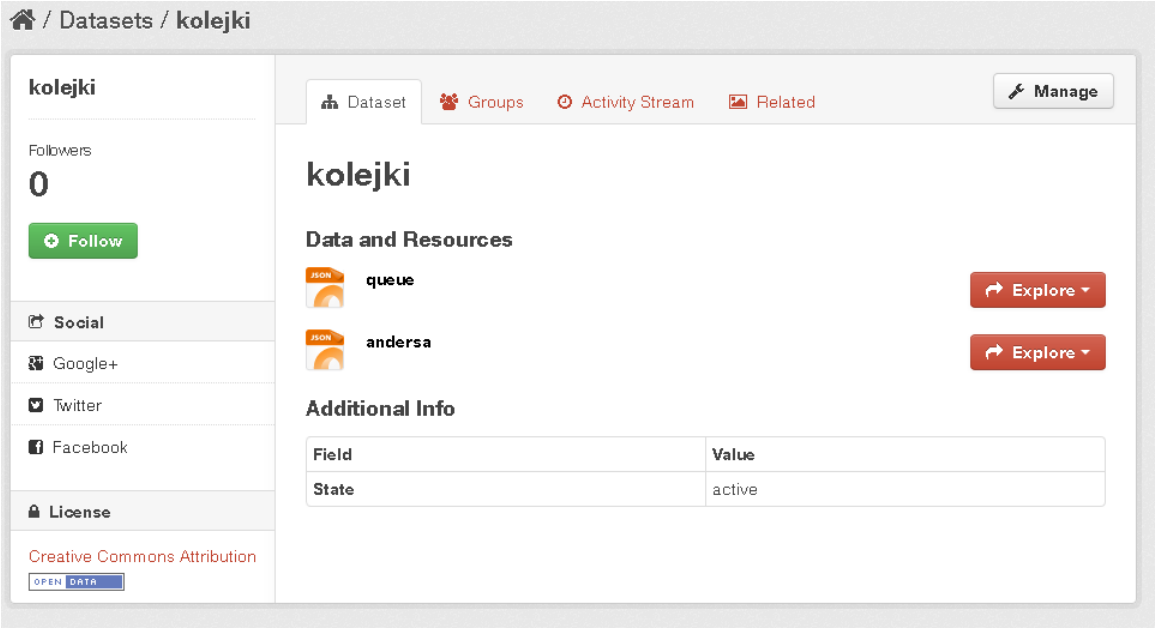
"result": {

"package_id": "72bc5b69-ca61-4955-8ef7-fe3be79f3750"

}

}

Oraz modyfikacją zbioru danych w CKAN



Rysunek 41 Zbiór typu Queue – po stronie Data Server (CKAN)

5. Konfiguracja elementu typu DB

Zbiory typu DB zawierają informacje z baz danych, które następnie są udostępniane przez platformę, jako usługi sieciowe (Web Services).

Konfiguracja po stronie Function Server

W pierwszym kroku, należy po stronie serwera JBoss zainstalować sterownik JDBC do danej bazy danych (o ile serwer nie posiada już załadowanego sterownika) i go skonfigurować.

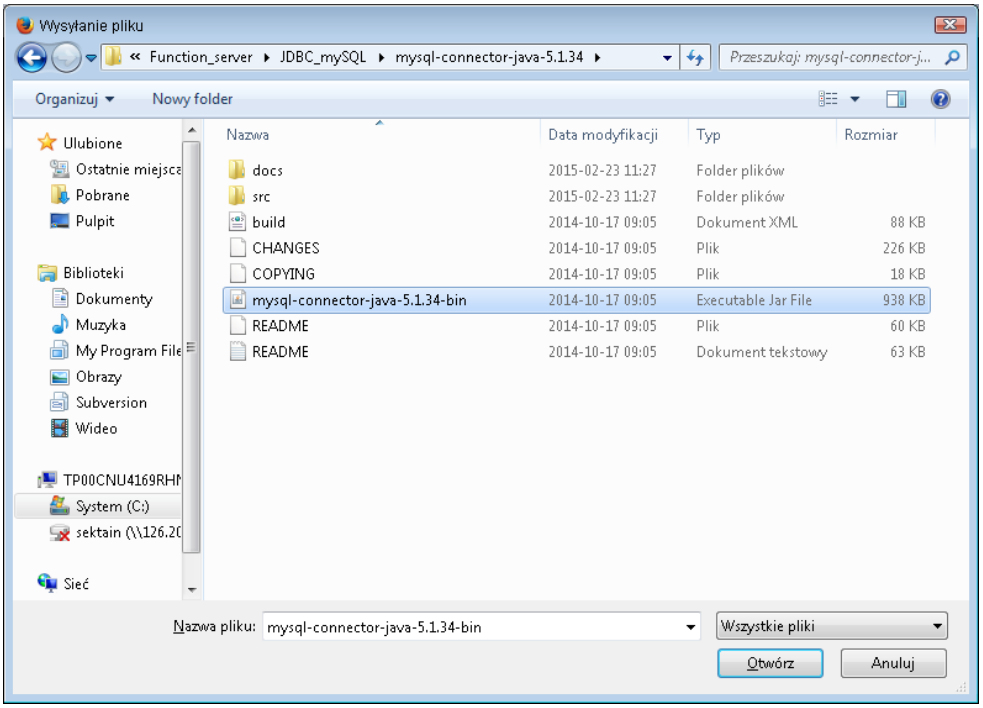
Konfiguracja po stronie JBoss

Poniżej podano przykład instalacji do bazy danych MySQL.

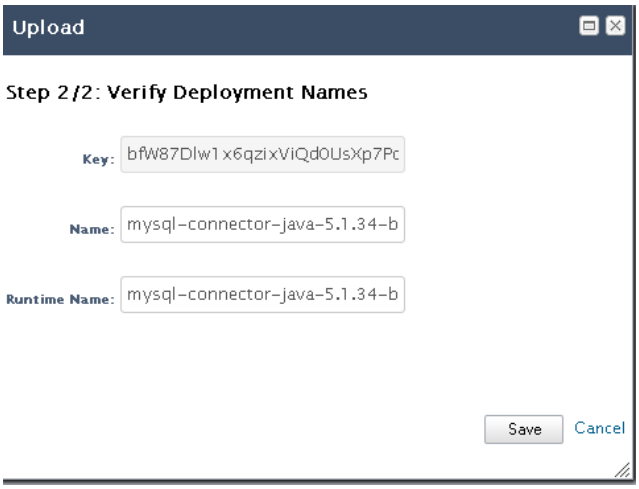
W zakładce Runtime wybieramy Deployments -> Manage Deployments I klikamy Add Content

Rysunek 42 Konfiguracja sterownika JDBC

Wskazujemy plik ze sterownikiem JDBC



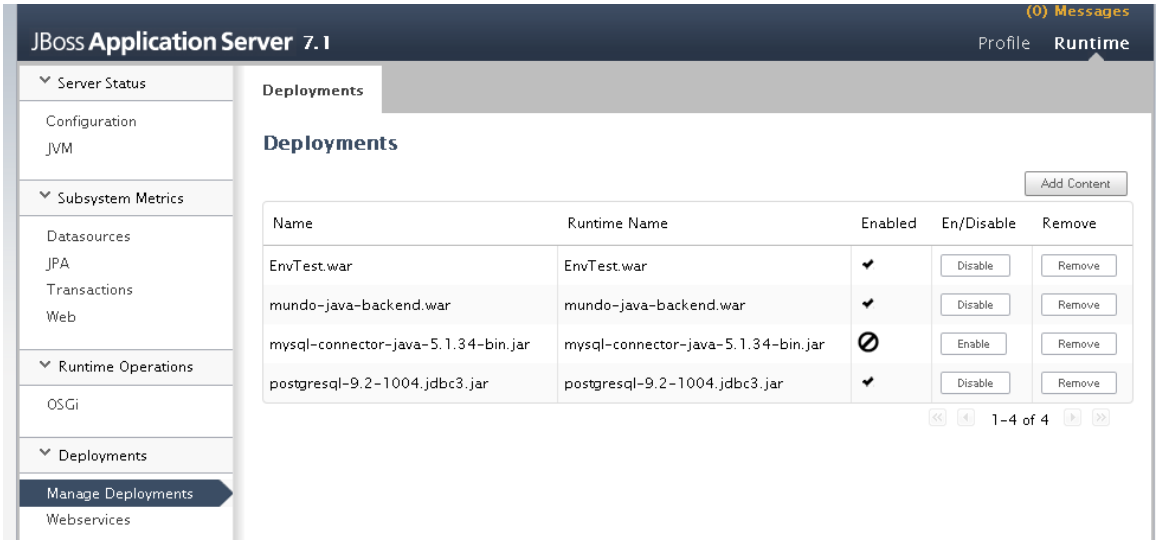
Rysunek 43 Konfiguracja sterownika JDBC



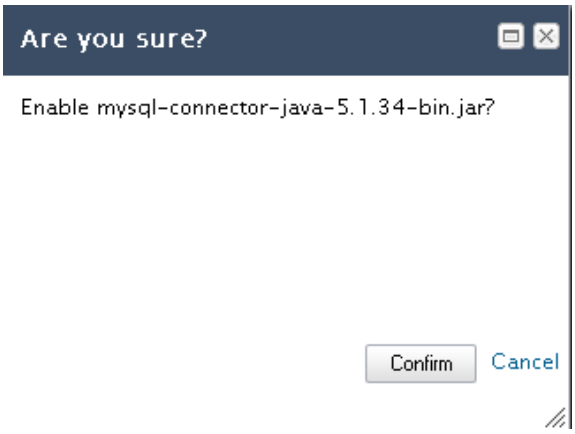
Rysunek 44 Konfiguracja sterownika JDBC

Klikamy save

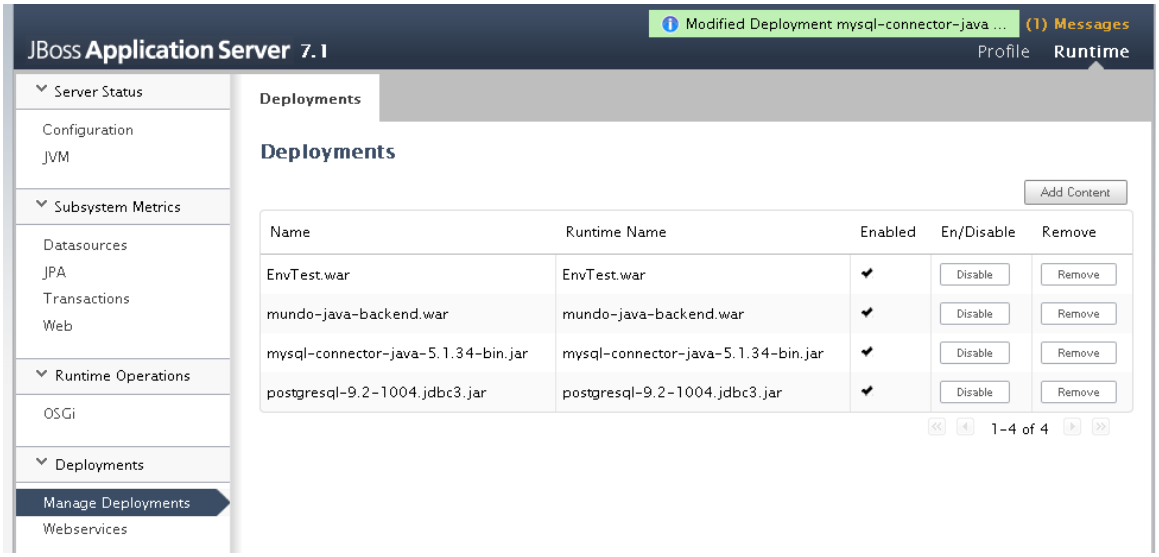
Oraz dokonujemy aktywacji sterownika klikając na przycisk Enable



Rysunek 45 Konfiguracja sterownika JDBC



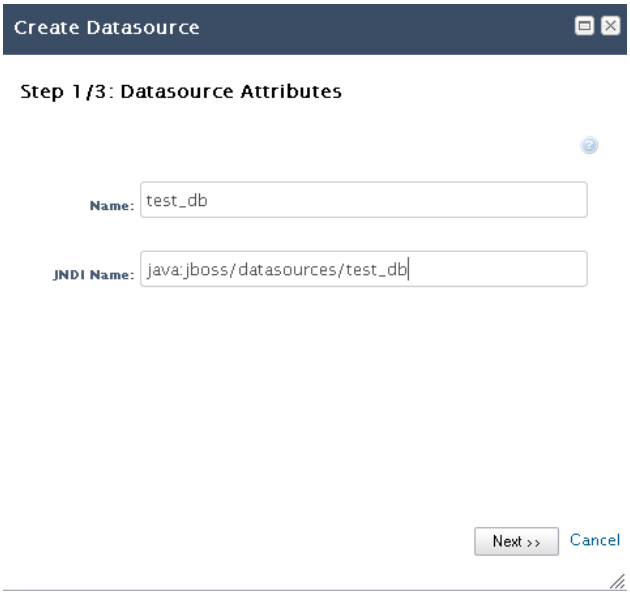
Rysunek 46 Konfiguracja sterownika JDBC



Rysunek 47 Konfiguracja sterownika JDBC

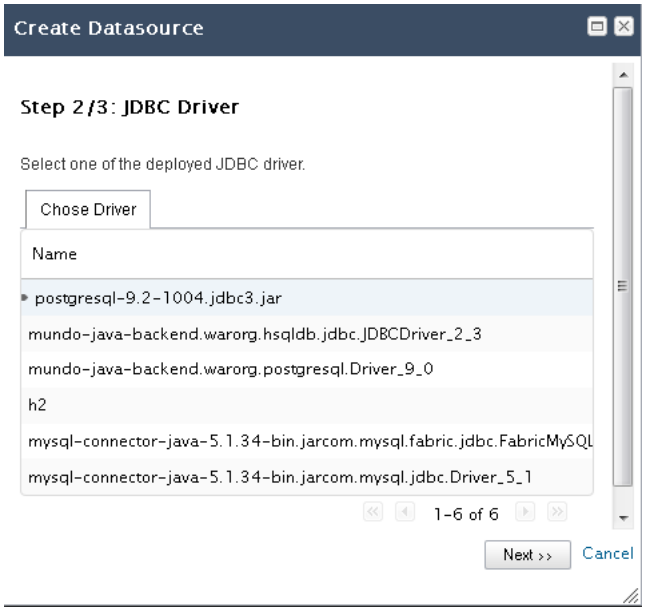
W kolejnym kroku konfigurujemy dostęp do danej bazy danych (na przykładzie konfiguracji sterownika do bazy danych PostgrSQL).

Na zakładce Profile wybieramy Connector -> Datasources i klikamy Add



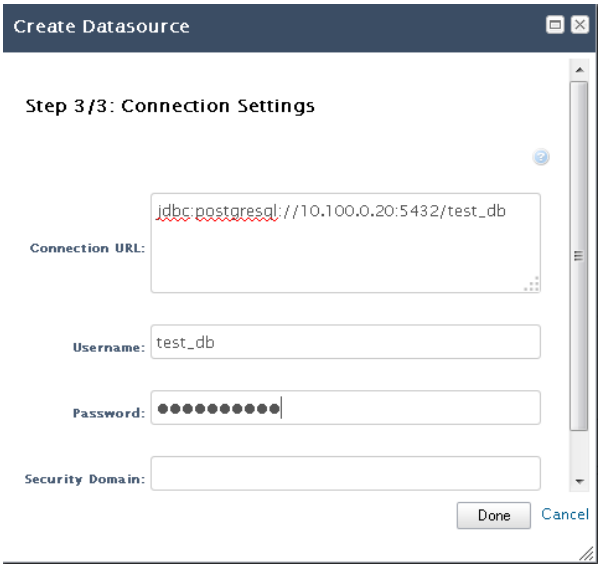
Rysunek 48 Konfiguracja JNDI

Wybieramy sterownik JDBC



Rysunek 49 Konfiguracja JNDI

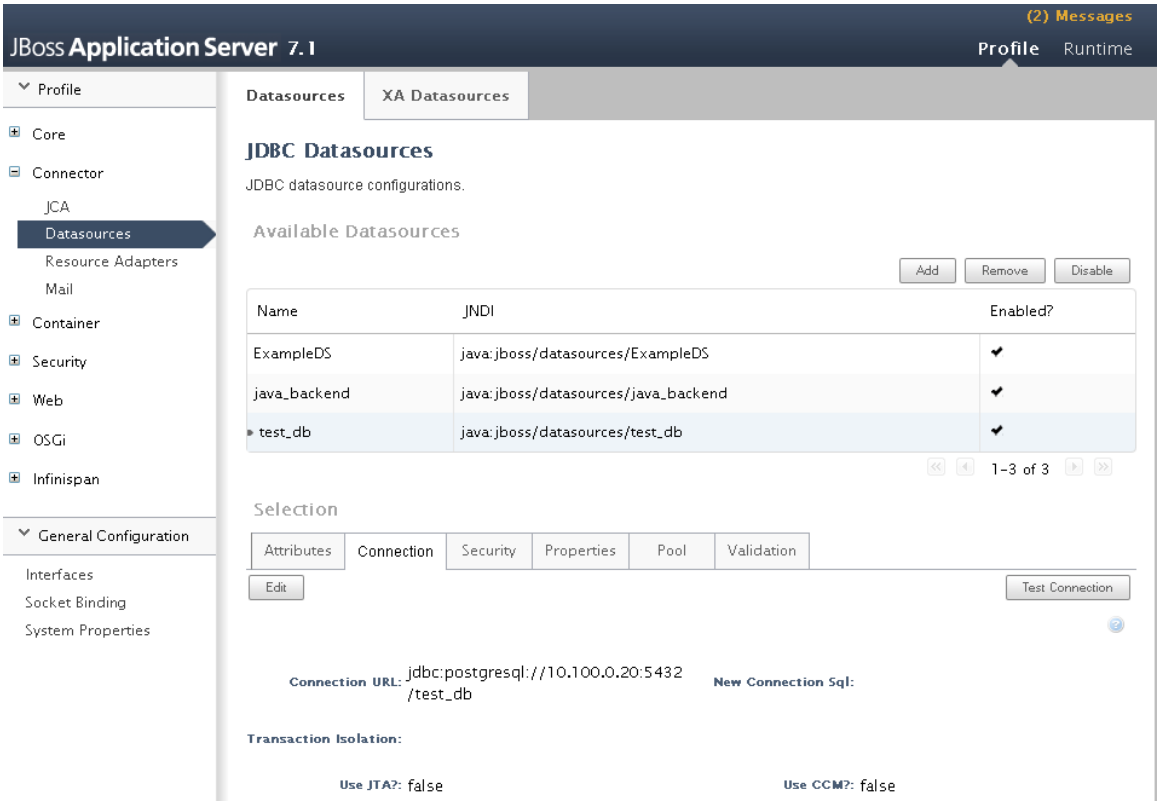
I w kolejnym kroku wpisujemy parametry połączenia np.:



Rysunek 50 Konfiguracja JNDI

i zatwierdzamy klikając na przycisk done

W zakładce Connection naciskając na przycisk Test Connection dokonujemy weryfikacji poprawności połączenia.

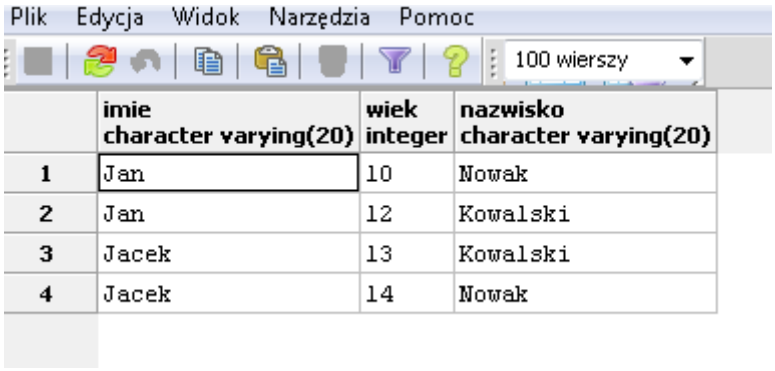


Rysunek 51 Konfiguracja JNDI

Rysunek 52 Konfiguracja JNDI – test połączenia

Konfiguracja zbioru danych typu DB w aplikacji MUNDO Backend

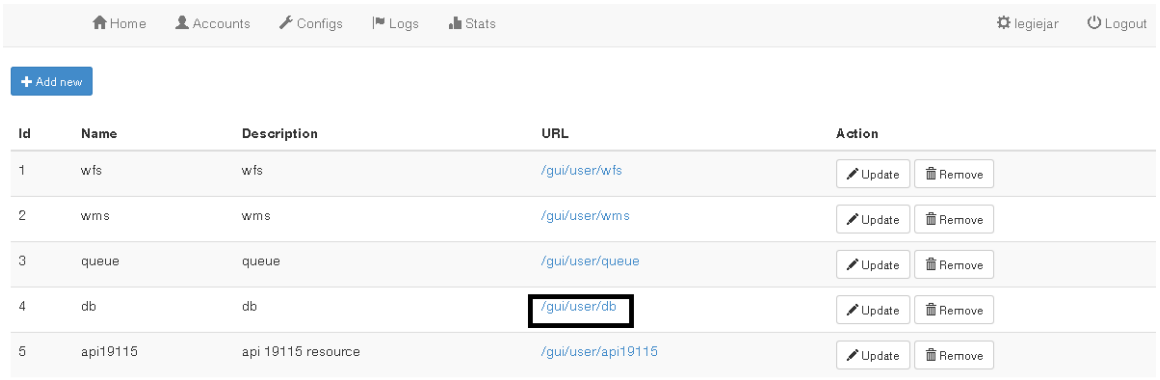
Poniżej przedstawiono przykład definiowana dostępu do tabeli „osoby” zapisanej w bazie danych PostgrSQL o strukturze:



	imie character varying(20)	wiek integer	nazwisko character varying(20)
1	Jan	10	Nowak
2	Jan	12	Kowalski
3	Jacek	13	Kowalski
4	Jacek	14	Nowak

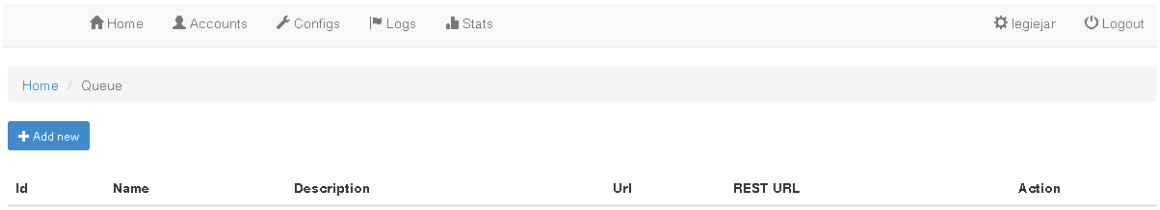
Rysunek 53 Przykładowa tabela – źródło danych DB

W celu skonfigurowania zbiorów danych znajdujących się w bazach danych w zakładce Home kliknąć na URL elementu o nazwie db.



Rysunek 54 Konfiguracja zbioru typu DB

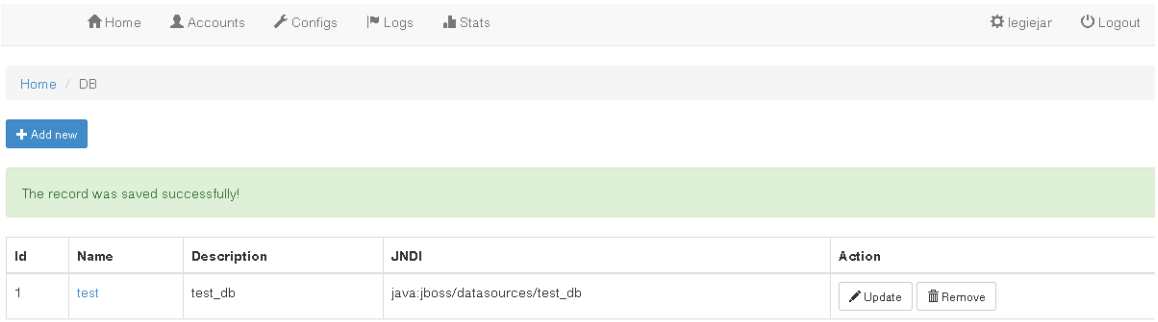
A następnie kliknąć na przycisk Add new



Rysunek 55 Konfiguracja zbioru typu DB

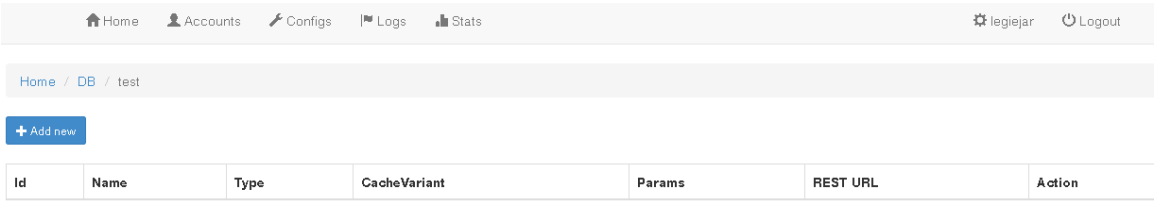
W kolejnym kroku należy podać nazwę , opis i JNDI Name dostępu do usługi np.:

Rysunek 56 Konfiguracja zbioru typu DB



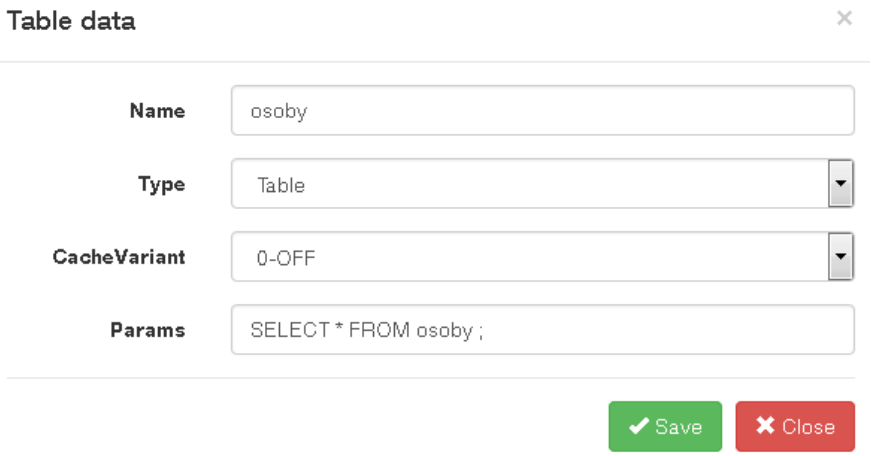
Rysunek 57 Konfiguracja zbioru typu DB

Po kliknięciu na nazwę zdefiniowanego zasobu DB (kolumna Name) definiujemy szczegóły ekspozycji danych z tej bazy poprzez usługi sieciowe platformy MUNDO:



Rysunek 58 Konfiguracja zbioru typu DB

Klikamy add new



Rysunek 59 Konfiguracja zbioru typu DB

Definiowane parametry:

Name – nazwa

Type – typ dostępu do zbioru:

Table – ekspozycja danych z tabeli (brak parametrów)

Procedure – wywoływanie procedury składowanej lub sparametryzowanego zapytania SQL

View – ekspozycja danych dostępnych przez widok

CacheVariant – definicja limitu wywołań do pierwotnego źródła danych z wykorzystaniem mechanizmów cache

0-Off – brak cache

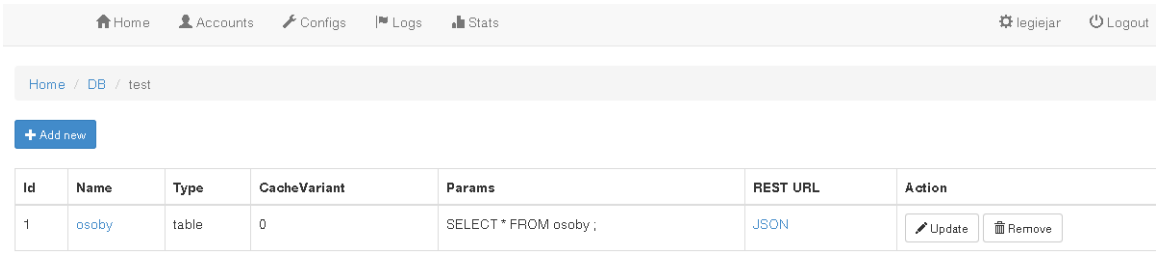
60 –1min – pobieranie danych raz na 1 minutę

360-1hour – pobieranie danych raz na 1 godzinę

43200-12hours – pobieranie danych raz na 12 godzin

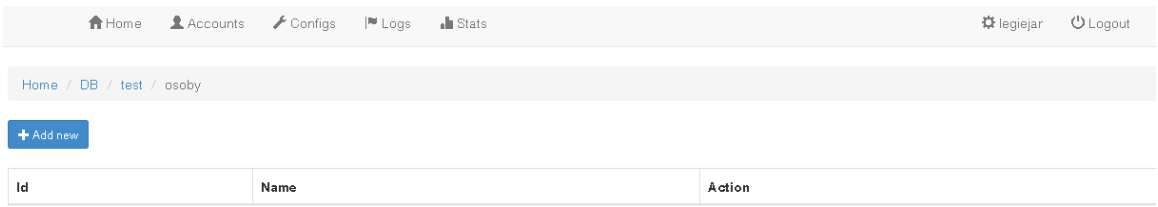
86400-24hours – pobieranie danych raz na 24 godziny

Params – definicja zapytania SQL lub stored procedure



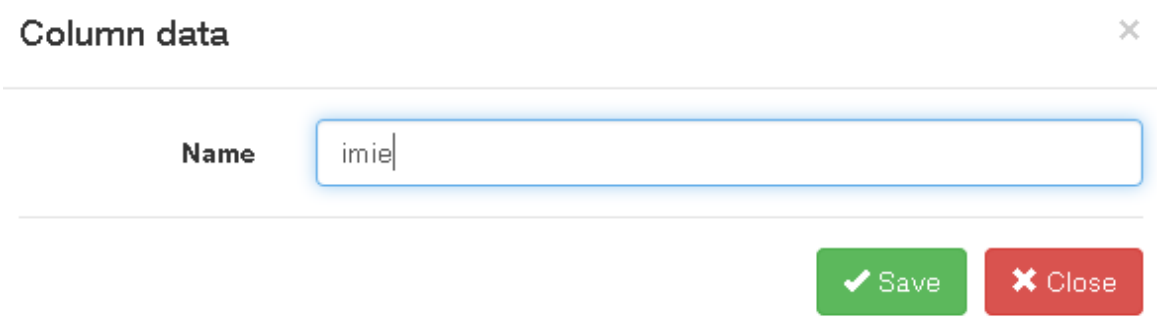
Rysunek 60 Konfiguracja zbioru typu DB

Po kliknięciu na nazwę zdefiniowanego zasobu (kolumna Name) definiujemy listę kolumn:



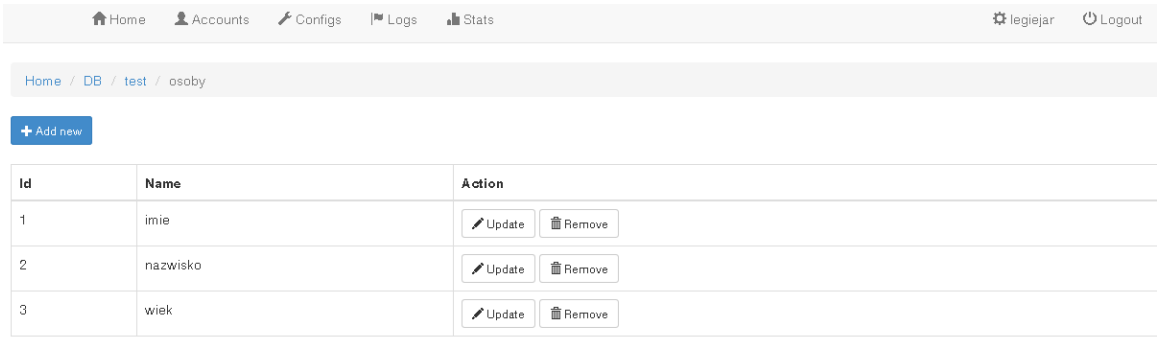
Rysunek 61 Konfiguracja zbioru typu DB

Klikamy add new:



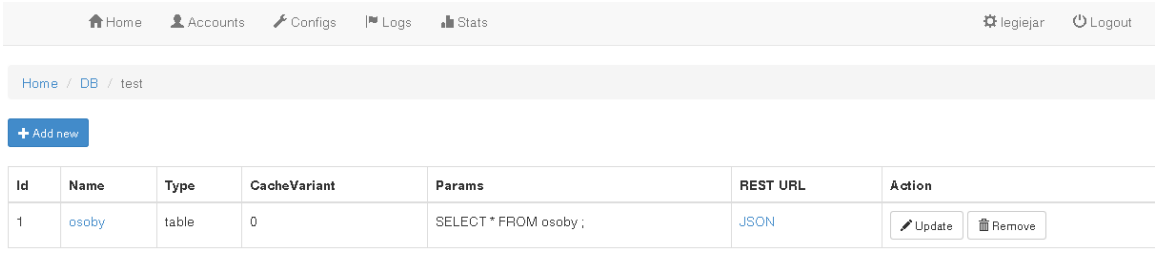
Rysunek 62 Konfiguracja zbioru typu DB

definiując kolumny jakie mają być eksponowane poprzez Web Services.



Rysunek 63 Konfiguracja zbioru typu DB

Po kliknięciu na link JSON (kolumna REST URL) wywołujemy API umożliwiające dostęp do danych.



Rysunek 64 Konfiguracja zbioru typu DB

Np.:

http://10.10.10.11:8080/cbr/mundo-java-backend/api/db/test/osoby

Rysunek 65 Zbiór typu DB – wywołanie API

Konfiguracja po stronie CKAN

Do konfiguracji zbioru danych typu DB po stronie Data Server (CKAN) służy rozszerzenie ckanext-dbstore.

Do dodania i modyfikacji zasobów (resource) po stronie CKAN służy zaimplementowana w w/w rozszerzeniu funkcja API dbstore_create.

Utworzenie zbioru danych z wybranymi zasobami

Utworzenie pojedynczego zbioru danych (dataset) zawierającego wybrane zasoby następuje poprzez wywołanie url:

http://adres_ip_ds/api/3/action/dbstore_create?name=**nazwa_dataset**&db_url=**http://adres_ip_fs:8080/cbr/mundo-java-backend/api/db/nazwa_db**

gdzie:

adres_ip_ds – adres IP Data Server (CKAN),

name – nazwa tworzonego zbioru danych,

adres_ip_fs – adres IP Function Server,

db_url – url dostępu do API w Function Server.

np:

http://10.10.10.10/api/3/action/dbstore_create?name=osoby&db_url=http://10.10.10.11:8080/cbr/mundo-java-backend/api/db/test/osoby

tworzy Dataset o nazwie kolejki z jednym resource osoby i zwraca jego id

{

*"help": "Adds a new DB store.\n\n **Params:**\n :name [String]: Dbstore name.\n :db_url [String]: Db url.\n :package_id [String] Existing package id. \n\n **Results:**\n\n :returns: The newly created data object.\n :rtype: dictionary\n ",*

"success": true,

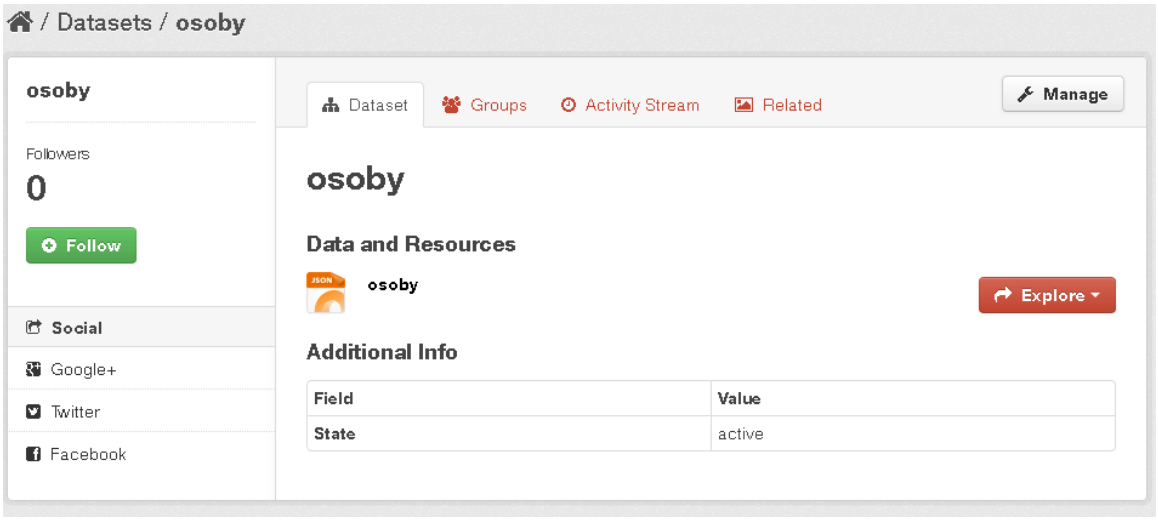
"result": {

"package_id": "e2657e71-31b3-4ea6-a199-627c0f85844d"

}

}

jednocześnie tworząc zbiór danych w CKAN:



Rysunek 66 Konfiguracja zbioru typu DB po stronie Data Server (CKAN)

Dodanie zasobu do istniejącego datasetu

http://adres_ip_ds/api/3/action/dbstore_create?name=**nazwa_zasobu**&db_url=**http://adres_ip_fs:8080/cbr/mundo-java-backend/api/db/nazwa_bazy**&package_id=**id_zbioru_danych**

gdzie:

adres_ip_ds – adres IP Data Server (CKAN),

name – nazwa tworzonego zasobu,

adres_ip_fs – adres IP Function Server,

layers – nazwa warstwy,

package_id – identyfikator zbioru_danych,

db_url – url dostępu do API w Function Server.

np.:

Dodanie resource do istniejącego datasetu o package_id e2657e71-31b3-4ea6-a199-627c0f85844d

http://10.10.10.10/api/3/action/dbstore_create?name=persons&db_url=http://10.10.10.11:8080/cbr/mundo-java-backend/api/db/test/osoby&package_id=e2657e71-31b3-4ea6-a199-627c0f85844d

zwraca odpowiedź:

{

*"help": "Adds a new DB store.\n\n **Params:**\n :name [String]: Dbstore name.\n :db_url [String]: Db url.\n :package_id [String] Existing package id. \n\n **Results:**\n\n :returns: The newly created data object.\n :rtype: dictionary\n ",*

"success": true,

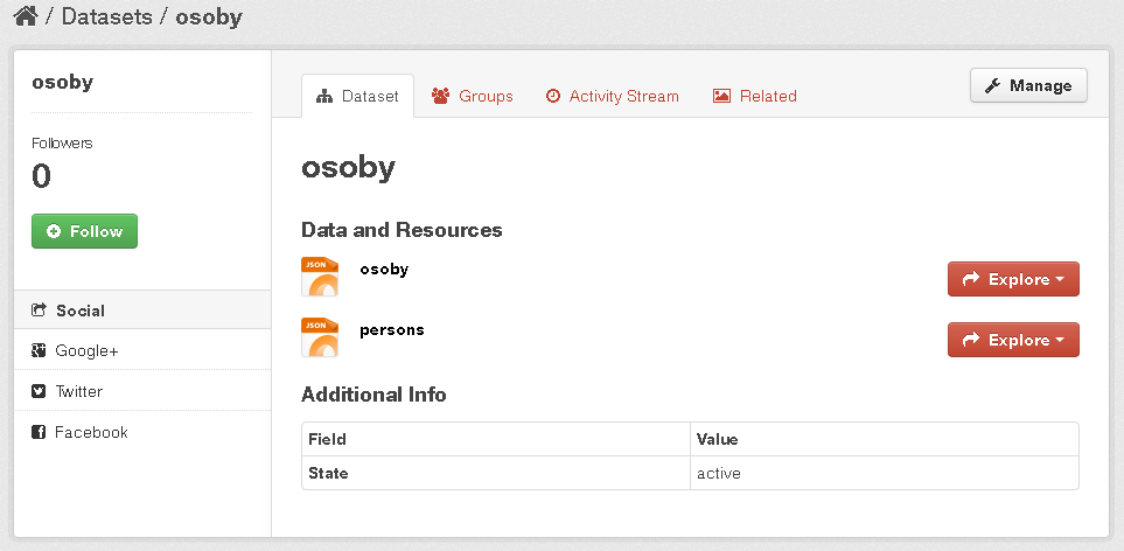
"result": {

"package_id": "e2657e71-31b3-4ea6-a199-627c0f85844d"

}

}

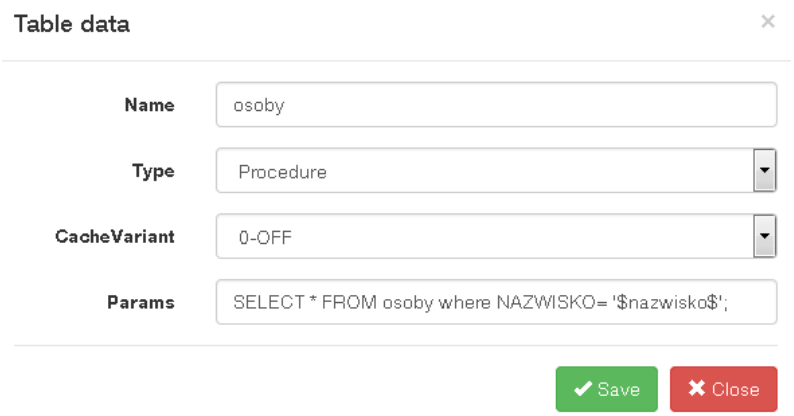
i modyfikuje zbiór danych w systemie CKAN.



Rysunek 67 Konfiguracja zbioru typu DB po stronie Data Server (CKAN)

Parametryzacja wywołań

W przypadku, gdy chcemy wykorzystać parametr w zapytaniu do zasobu zbioru danych typu DB, nazwę parametru należy podać w polu Params w postaci **'$parametr$'** a konfiguracja po stronie Function Server wygląda następująco:



Rysunek 68 Konfiguracja zbioru typu DB – parametryzacja wywołań

Co skutkuje możliwością parametryzowania wywołania API np. w postaci:

http://10.10.10.11:8080/cbr/mundo-java-backend/api/db/test/osoby?nazwisko=Nowak

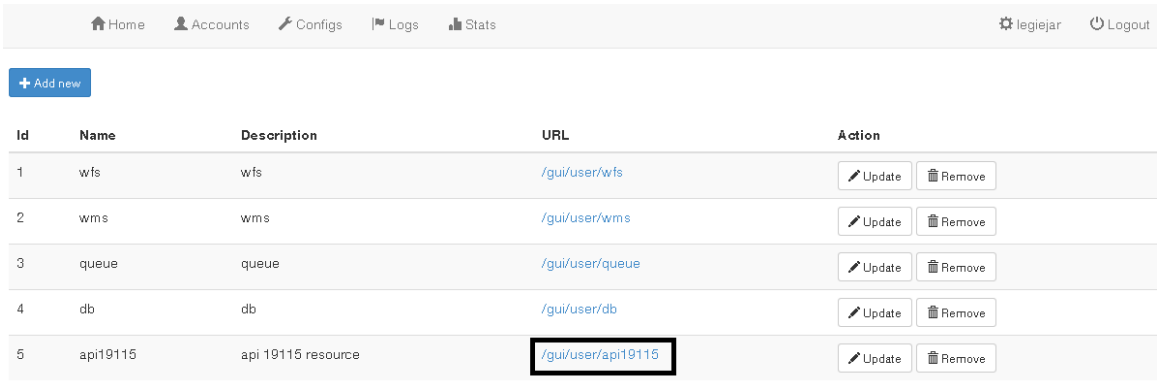
Rysunek 69 Konfiguracja zbioru typu DB – parametryzacja wywołań

6. Konfiguracja elementu typu 19115

Zbiory 19115 to element eksponujący dane z systemu zgłaszania problemów 19115 UM Warszawa, które są udostępniane jako usługi sieciowe (Web Services)

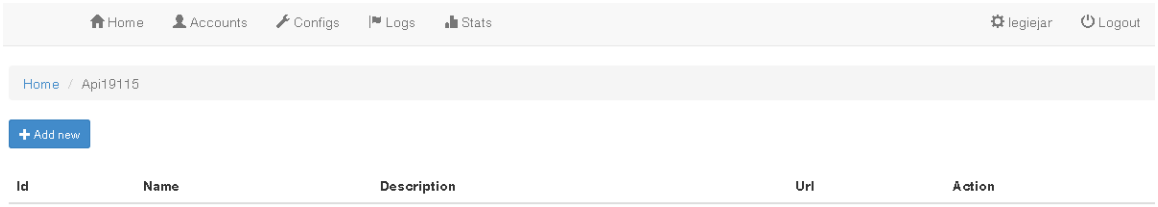
Konfiguracja po stronie Function Server

W celu skonfigurowania zbioru danych związanych usługa 19115 należy w zakładce Home kliknąć na URL elementu o nazwie api19115



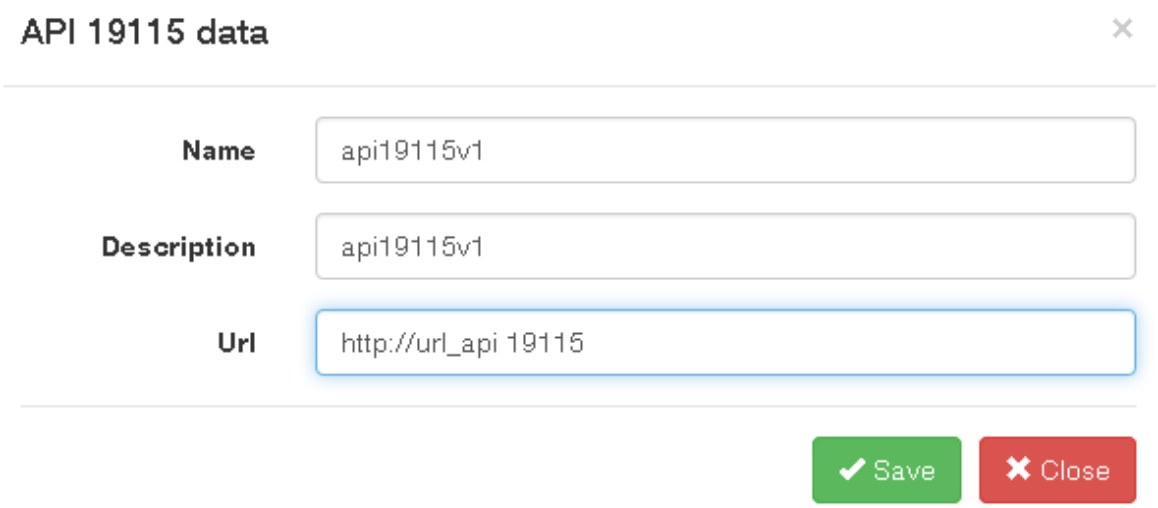
Rysunek 70 Konfiguracja zbioru typu 19115

A następnie kliknąć na przycisk Add new



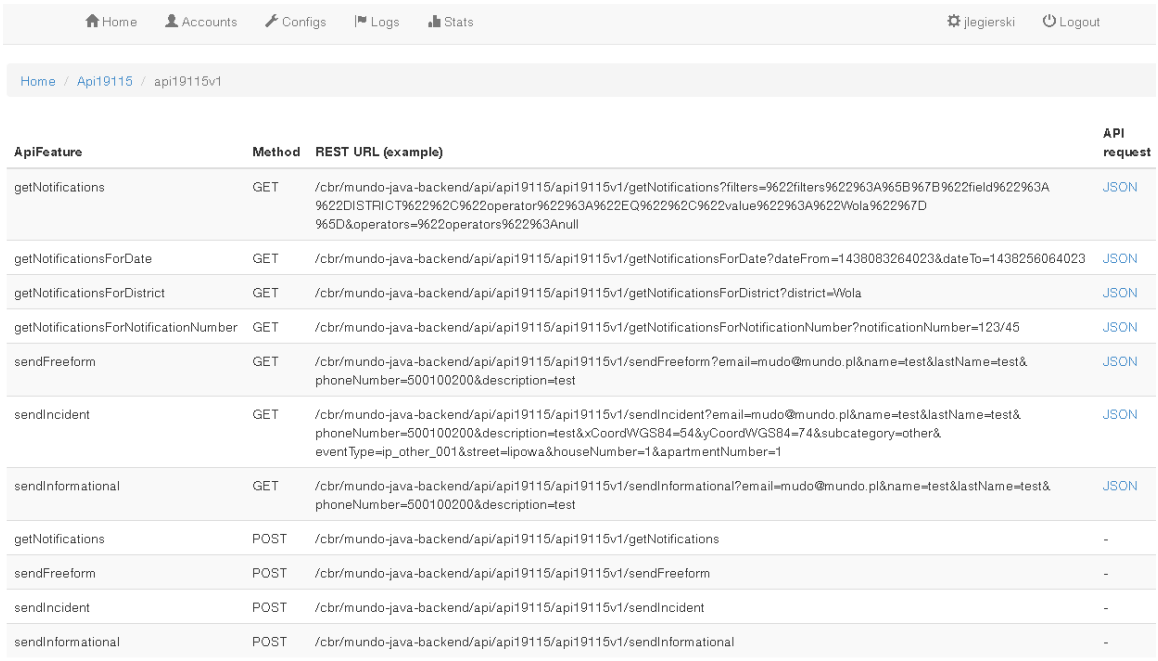
Rysunek 71 Konfiguracja zbioru typu 19115

W kolejnym kroku należy podać nazwę , opis i url dostępu do usługi np.:



Rysunek 72 Konfiguracja zbioru typu 19115

W przypadku poprawnej konfiguracji po kliknięciu na nazwę zdefiniowanego zasobu 19115 (kolumna Name) otrzymamy listę dostępnych metod dla tej usługi



ApiFeature	Method	REST URL (example)	API request
getNotifications	GET	/cbr/mundo-java-backend/api/api19115/api19115v1/getNotifications?filters=%22filters%22%3A%5B%7B%22field%22%3A%22DISTRICT%22%2C%22operator%22%3A%22EQ%22%2C%22value%22%3A%22Wola%22%7D%5D&operators=%22operators%22%3Anull	JSON
getNotificationsForDate	GET	/cbr/mundo-java-backend/api/api19115/api19115v1/getNotificationsForDate?dateFrom=1438083264023&dateTo=1438256064023	JSON
getNotificationsForDistrict	GET	/cbr/mundo-java-backend/api/api19115/api19115v1/getNotificationsForDistrict?district=Wola	JSON
getNotificationsForNotificationNumber	GET	/cbr/mundo-java-backend/api/api19115/api19115v1/getNotificationsForNotificationNumber?notificationNumber=123/45	JSON
sendFreeform	GET	/cbr/mundo-java-backend/api/api19115/api19115v1/sendFreeform?email=mudo@mundo.pl&name=test&lastName=test&phoneNumber=500100200&description=test	JSON
sendIncident	GET	/cbr/mundo-java-backend/api/api19115/api19115v1/sendIncident?email=mudo@mundo.pl&name=test&lastName=test&phoneNumber=500100200&description=test&xCoordWGS84=54&yCoordWGS84=74&subcategory=other&eventType=ip_other_001&street=lipowa&houseNumber=1&apartmentNumber=1	JSON
sendInformational	GET	/cbr/mundo-java-backend/api/api19115/api19115v1/sendInformational?email=mudo@mundo.pl&name=test&lastName=test&phoneNumber=500100200&description=test	JSON
getNotifications	POST	/cbr/mundo-java-backend/api/api19115/api19115v1/getNotifications	-
sendFreeform	POST	/cbr/mundo-java-backend/api/api19115/api19115v1/sendFreeform	-
sendIncident	POST	/cbr/mundo-java-backend/api/api19115/api19115v1/sendIncident	-
sendInformational	POST	/cbr/mundo-java-backend/api/api19115/api19115v1/sendInformational	-

Rysunek 73 Konfiguracja zbioru typu 19115

Po kliknięciu na przycisk JSON otrzymujemy predefiniowane przykładowe wywołanie API danej metody np.:

http://10.10.10.11:8080/cbr/mundo-java-backend/api/api19115/api19115v1/getNotifications?filters=%22filters%22%3A[{%22field%22%3A%22DISTRICT%22%2C%22operator%22%3A%22EQ%22%2C%22value%22%3A%22Wola%22}]&operators=%22operators%22%3Anull

```
{
   responseCode: "SUCCESS",
   responseDesc: "OK",
 ▾ notifications: [
     ▾ {
           siebelEventId: "1-MOZP",
           deviceType: "UNKNOWN",
           street: "Znana",
           street2: null,
           district: "Wola",
           city: "Warszawa",
           houseNumber: "1",
           aparmentNumber: null,
           category: "Infrastruktura",
           subcategory: "Komunikacja/Transport",
           event: "Stan techniczny pojazdu, uwagi dotyczące niepunktualności",
           description: "Autobus linii nr 197 o godzinie 11:12  oraz 11:27 nie przyjechał",
           createDate: 1350985209000,
           notificationNumber: "403/12",
           xCoordOracle: 7495031.243,
           yCoordOracle: 5788551.377,
           xCoordWGS84: 0,
           yCoordWGS84: 0,
           notificationType: "INCIDENT",
           statuses: [ ],
           source: "CALL"
       },
     ▾ {
           siebelEventId: "1-5QF6R",
           deviceType: "UNKNOWN",
           street: "Skwer gen. J. Jura-Gorzechowskiego",
           street2: null,
           district: "Wola",
           city: "Warszawa",
           houseNumber: null,
           aparmentNumber: null,
           category: "Porządek",
           subcategory: "Śmieci",
           event: "Zgłoszenie dla Zakładu Oczyszczania Miasta",
           description: "W Parku jest bardzo brudno. Śmieci wysypują się na ścieżki, plac zabaw jest zanieczyszczony, ukliczki w Parku są brudne i ludzie
           brzydza się tamtędy chodzić",
           createDate: 1368172061000,
           notificationNumber: "18156/13",
           xCoordOracle: 7499176.174991,
           yCoordOracle: 5790209.876242,
```


Rysunek 74 Konfiguracja zbioru typu 19115 – wywołanie API

Konfiguracja po stronie CKAN

Do konfiguracji zbioru danych typu 19115 po stronie Data Server (CKAN) służy rozszerzenie 19115store.

Dodanie zbioru danych 19115 na w systemie CKAN ma miejsce poprzez metodę API rozszerzenia 19115store_create.

Utworzenie zbioru danych z wybranymi zasobami

Utworzenie pojedynczego zbioru danych (dataset) zawierającego wybrane zasoby wymaga wywołania url:

http://adres_ip_ds/api/3/action/19115store_create?name=**nazwa_dataset**&api_url=**http://adres_ip_fs:8080/cbr/mundo-java-backend/api/api19115/api19115v1/getNotifications**

gdzie:

adres_ip_ds – adres IP Data Server (CKAN)

name – nazwa tworzonego zbioru danych

adres_ip_fs – adres IP Function Server

api_url – url dostępu do API w Function Server

Np.:

http://10.72.1.80/api/3/action/19115store_create?name=19115v1&api_url=http://10.72.1.81:8080/cbr/mundo-java-backend/api/api19115/api19115v1/getNotifications

W przypadku poprawnego wywołania funkcja zwraca package_id

{

*"help": "Adds a new 19115store.\n\n **Params:**\n :name [String]: 19115store name.\n :api_url [String]: api url.\n :package_id [String] Existing package id.\n\n **Results:**\n\n :returns: The newly created data object.\n :rtype: dictionary\n ",*

"success": true,

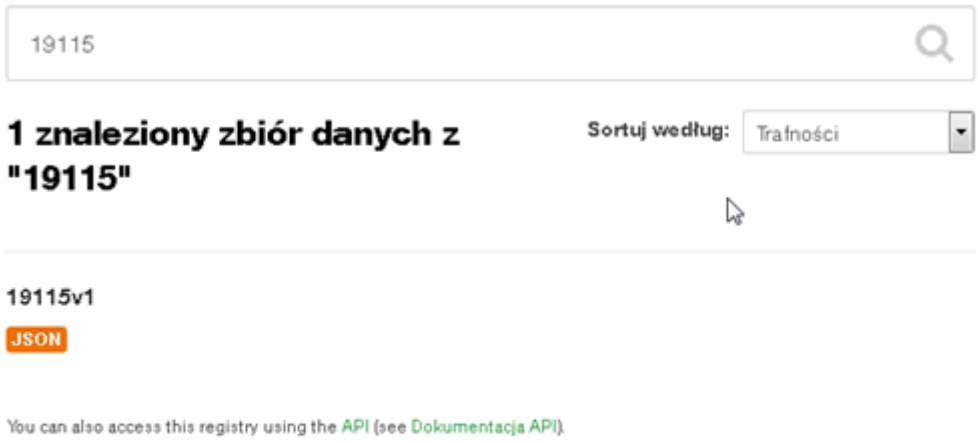
"result": {

"package_id": "7ad5c04d-1d3d-400b-af3c-50244e381d2c"

}

}

Oraz zbiór zostaje umieszczony na platformie:



Rysunek 75 Zbiór typu 19115 po stronie Data Server (CKAN)

W przypadku błędnego wywołania jest zwracany kod błędu wraz z opisem:

{
*"help": "Adds a new 19115store.\n\n **Params:**\n :name [String]: 19115store name.\n :api_url [String]: api url.\n :package_id [String] Existing package id.\n\n **Results:**\n\n :returns: The newly created data object.\n :rtype: dictionary\n ",*
"success": false,
"error": {

"__type": "Validation Error",
"name": [
"name required"
]
}
}

W podobny sposób są obsługiwane błędy w wywołaniach api we wszystkich pozostałych rozszerzeniach CKAN.

4 Podsumowanie

Niniejszy dokument zawiera dokumentację konfiguracyjną platformy MUNDO. Należy podkreślić, iż przedstawione w niniejszym dokumencie pierwotne zbiory danych są zbiorami przykładowymi w większości bazującymi na danych eksponowanych przez Miasto Warszawa i służą jedynie zobrazowaniu sposobu konfiguracji elementów platformy MUNDO. Dane te mogą być wykorzystane jedynie po otrzymaniu zgody ich właściciela i spełnieniu warunków dotyczących udostępniania tych danych zamieszczonych na stronach https://api.um.warszawa.pl oraz http://www.mapa.um.warszawa.pl/warunki.html.

5 Słowniczek – lista skrótów

Glossary		
Nb.	**Abbreviation**	**Explanation**
1	API	Application Programming Interface
2	GPL	GNU Public License – Licencja wolnego i otwartego oprogramowania
3	LGPL	Lesser GNU Public License - słabsza powszechna licencja publiczna GNU, powszechna licencja publiczna dla bibliotek
4	FDL	Free Documentation License - Licencja Wolnej Dokumentacji
5	CC	Creative Commons
6	CKAN	Comprehensive Knowledge Archive Network – sieć archiwizacyjna dla kompleksowej wiedzy – system www będący repozytorium danych i eksponujący dane otwarte stworzony przez Open Knowledge Foundation
7	WS	Web Service – usługa sieciowa oparta o SOA/SOAP lub ROA/REST
8	SOA	architektura zorientowana na usługi (ang. Service Oriented Architecture)
9	SOAP	SOAP (ang.) Simple Object Access Protocol – protokół wywoływania zdalnego dostępu do obiektów oparty o XML
10	ROA	Architektura oparta o zasoby (ang. resource-oriented architecture)
11	REST	Representational State Transfer – wzorzec architektury oprogramowania
12	MD	metadane (dane o danych),
13	WMS	Web Map Service - stworzony przez Open Geospatial Consortium (OGC) standard udostępniania map w postaci rastrowej za pomocą interfejsu HTTP
14	WFS	Web Feature Service stworzony przez Open Geospatial Consortium (OGC) standard udostępniania map w postaci wektorowej za pomocą interfejsu http
15	DS	DS – data set zbiór danych
16	FS	FS - function set - zbiór funkcji (np. realizowanych przez web service)
17	APIKey	Kod alfanumeryczny przekazanego do serwera API przez programy komputerowe wywołujące API
18	RDF	Ang. Resource Description Framework - język/metoda pozwalająca na opisywanie zasobów sieci Web, ze składnią opartą na XML, opracowana przez W3C
19	Dublin Core	Dublin Core Metadata Element Set, DC, DCES) – ogólny standard metadanych. Przyjęty jako standard ISO 15836-2003.
20	DCAT	W3C Data Catalog Vocabulary - standard schematu katalogowego dla danych
21	CSV	Comma separated values – plik z wartościami oddzielanymi przecinkami
22	CMS	system zarządzania treścią (content management system)
23	SPARQL	SPARQL Protocol And RDF Query Language - język zapytań i protokół dla plików RDF.

24	OWL	Web Ontology Language – język ze składnią opartą na XML, a semantyką opartą na logice opisowej
25	HTTP	Hypertext Transfer Protocol
26	SSL	Secure Socket Layer

6 Bibliografia

[1] Instalacja CKAN2.0 na CentOS 6.3 https://github.com/ckan/ckan/wiki/How-to-install-CKAN-2.0-on-CentOS-6.3-%28new%29

[2] Instalacja Tomcat 6.0 na CentOS 7.0 http://www.howtoforge.com/how-to-install-tomcat-on-centos-7

[3] Instalacja Tomcat 6 z JDK 7 na CentOS RHEL Fedora Debian Ubuntu http://linuxdrops.com/install-tomcat-6-with-jdk-7-on-centos-rhel-fedora-debian-ubuntu/

[4] Instalacja Tomcat 6 na CentOS lub RHEL http://www.davidghedini.com/pg/entry/install_tomcat_6_on_centos

[5] Instalacja CKAN Filestore http://docs.ckan.org/en/latest/maintaining/filestore.html

[6] Instalacja CKAN DataStore http://docs.ckan.org/en/latest/maintaining/datastore.html

[7] Instalacja CKAN DataPusher http://docs.ckan.org/projects/datapusher/en/latest/

[8] CKAN filestore preview problem https://github.com/ckan/ckan/pull/1160

[9] Wniosek do 1 konkursu Innowacje Społeczne NCBIR projektu MUNDO, Warszawa, 2013

[10] Portal CKAN http://ckan.org/ [30.10.2014]

[11] Portal The Open Knowledge Foundation https://okfn.org/ [30.10.2014]