1 API

W ramach otwartej, niekomercyjnej i niezależnej technologicznie hurtowni danych PRZYJAZDY.PL przetwarzamy rozkłady jazdy środków polskiego transportu publicznego. Powiązane dane o komunikacji kolejowej, drogowej, miejskiej i lotniczej służą prezentacji informacji o publicznych połączeniach multimodalnych w Polsce, realizując ważny interes społeczny.

Udostępnione API umożliwia wykorzystywanie danych z serwisu PRZYJAZDY.PL w zewnętrznych aplikacjach tworzonych przez kreatywnych zewnętrznych programistów, pod warunkiem zachowania zasad projektu i warunków licencji.

Korzystając z API PRZYJAZDY.PL należy mieć na uwadze, że projekt PRZYJAZDY.PL realizowany jest w ramach obchodów stulecia odzyskania niepodległości (Program Wieloletni „Niepodległa") oraz objęty jest Patronatem Honorowym Ministra Infrastruktury, Przedstawicielstwa Komisji Europejskiej w Polsce, szeregu marszałków województw, prezydentów i burmistrzów miast, a także Ambasady Węgier w Polsce.

https://**przyjazdy.pl/patronaty**

1.1 Licencjonowanie

Dane oferowane są nieodpłatnie do wykorzystania niekomercyjnego w aplikacjach tworzonych przez kreatywnych zewnętrznych programistów, pod następującymi warunkami:

- poszanowania licencji **CC-BY-SA**,
- wyświetlenie licencji (pole **License**) w sposób widoczny dla użytkownika,
- wyświetlenie daty pobrania informacji przetworzonej (pole **Data**),
- umieszczenie łącza (pole **Url**) w sposób umożliwiający użytkownikowi jego użycie,
- umieszczenie czytelnej informacji o pochodzeniu danych z API PRZYJAZDY.PL,
- umieszczenie łącza do https://**przyjazdy.pl** w aplikacji

1.2 Format zapytania

Serwer reaguje na zapytania HTTP/1.1 metodą GET.

Przykład w PHP:

```
$APIkey = "MojaApka_v.1.0_programista@domena.pl";
$baseURL = "https://przyjazdy.pl/api/";
$id = "przystanki/gdansk";
$url = $baseURL.$id."?key=".$APIkey;

$ch = curl_init();
curl_setopt($ch, CURLOPT_URL, $url);
curl_setopt($ch, CURLOPT_RETURNTRANSFER, 1);
curl_setopt($ch, CURLOPT_FOLLOWLOCATION, 1);
curl_setopt($ch, CURLOPT_TIMEOUT, 1);
curl_setopt($ch, CURLOPT_CONNECTTIMEOUT, 1);
curl_setopt($ch, CURLOPT_HTTPGET, 1);
curl_setopt($ch, CURLOPT_ENCODING, "gzip,deflate");
curl_setopt($ch, CURLOPT_USERAGENT, "MojaApka-v.1.0");
curl_setopt($ch, CURLOPT_REFERER, "https://strona-programisty.domena.pl");

$res = curl_exec($ch);
$code = curl_getinfo( $ch, CURLINFO_HTTP_CODE);
curl_close($ch);

if ($code>=500) return false;
if ($code>=400) exit;
$wynik = json_decode($json, 1);
```

1.3 Klucz API

W okresie testowym serwer reaguje na dowolnie wybrany przez użytkownika klucz. Klucz ten należy stosować konsekwentnie do wszystkich zapytań. Kluczem może być adres e-mail, nick programisty albo

unikalna nazwa aplikacji. Klucz należy wybrać tak, aby był unikalny. Klucz może zawierać litery, cyfry, proste znaki przestankowe, nie może zawierać slashy ani znaków specjalnych.

1.4 Format odpowiedzi

Dane zwracane są w formacie JSON. Przykładowa odpowiedź ma postać:


```
{

        "API": "PRZYJAZDY.PL v.2018.10.9.1",
        "License": "CC-BY-SA - Studio JZK sp. z o.o., 2018",
        "Data": "Tue, 16 Oct 2018 16:30:17 +0200",
        "IP": "255.255.255.255",
        "TwojeZapytanieWTymMiesiacu": 119,
        "TwojeZapytanieDzis": 8,
        "TwojLimit": 99999,
        "TwojLimitDzienny": 9999,

        "Przystanki": {
         "gdansk": {
           "1856_0": {
                 "nazwa": "Kolorowa",
                 "gps": "54.31462,18.60027",
                 "url": "https://przyjazdy.pl/gdansk/st/1856_0",
                 "api": "https://przyjazdy.pl/api/rozklad/gdansk/1856_0"
                       }
                }
            },
    "Time": "3.3 ms"
}
```


Należy rozpoznać i reagować odpowiednio na informację o udzielonym i wykorzystanym limicie, a także o czasie realizacji zapytania. Odpowiedź merytoryczna zależy od złożonego zapytania, może zawierać szereg informacji dodatkowych lub więcej tabel.

1.5 Zasady dobrych praktyk

Tylko jedno zapytanie z jednego źródła w jednym czasie. Należy używać metod synchronicznych, aby zapewnić spełnienie tego warunku.

Nie więcej niż jedno zapytanie na sekundę. Należy wprowadzić licznik czasu po stronie aplikacji lub alternatywnie opóźnienie **(sleep)** między zapytaniami. Dobre praktyki zalecają ustawienie dla odstępu o losowej wartości tak, aby średnia kilku kolejnych odstępów wynosiła 1000 ms lub więcej.

Reagowanie na sytuacje krytyczne. Po otrzymaniu kodu HTTP serii 4XX (np. 401, 404, 410, 406) należy przerwać pracę aplikacji. Po otrzymaniu kodu HTTP serii 5XX (np. 500, 505, 510, 550) należy przerwać ewentualną pętlę na założony czas (minimum 60 sekund), po n-tym wystąpieniu błędu serii 5XX należy czas oczekiwania zwiększać geometrycznie (lub silnia), a przy przekroczeniu 10 błędów przerwać pracę. Adresy IP powodujące efekt DoS lub DdoS mogą być blokowane automatycznie.

Liczyć zapytania. Należy odczytać ustawiony dla danego klucza limit i tak realizować zapytania, aby nie dochodziło do stałego przekraczania limitów. Klucze stale przekraczające limity mogą być blokowane.

Sprawdzać czas reakcji. Należy odczytywać i analizować czas, podany w polu „Time" zwracanej informacji. Jeśli czas reakcji netto przekracza 10 ms, oznacza to, że serwer jest przeciążony, wobec czego należy zmniejszyć ilość zapytań. Jeśli czas reakcji przekracza 100 ms, należy szczególnie zmniejszyć ilość zapytań. Jeśli czas reakcji przekracza 1000 ms, należy wstrzymać zapytania.

Keep-Alive i gzip/deflate. Należy w zapytaniu ująć nagłówki:

```
Accept-Encoding: gzip, deflate
Connection: keep-alive
```

SSL i TLS. Serwer obsługuje połączenia szyfrowane oraz nieszyfrowane. W przypadku danych, które nie są krytyczne, można używać połączeń nieszyfrowanych, które są szybsze. Użycie szyfrowania zwiększa czas odpowiedzi brutto o około 100 ms.

Prezentacja tożsamości. Minimalnym wymogiem jest podanie prawidłowych nagłówków **Referer** (URL strony programisty) oraz **User-Agent** (nazwa aplikacji). Zalecamy dodanie nagłówka informacyjnego:

```
X-Przyjazdy: "Nazwa aplikacji, wersja, kontakt do programisty"
```

Pozwoli to na lepszą współpracę z kreatywnym programistą i reagowanie na sytuacje wyjątkowe.

Cache zapytań. Dane na serwerze odświeżane są raz na dobę, w godzinach nocnych. Raz zrealizowane zapytanie nie powinno być ponawiane w tej samej dobie, zamiast tego należy przechować jego wynik w aplikacji.

Weryfikacja merytoryczna odpowiedzi. Zestaw i format zwracanych danych może zmieniać się w czasie. Jeśli aplikacja stwierdzi, że serwer nie zwraca prawidłowych danych mimo kodu odpowiedzi serii 2XX, należy przerwać pracę i wyświetlić użytkownikowi komunikat o konieczności zaktualizowania aplikacji.

2 Dostępne zestawy danych

Zestaw danych dostępny dla programisty zależy od klucza, zapytania oraz od wersji API.

2.1 Lista dostępnych zestawów

Pierwsze wywołanie powinno mieć postać

```
https://przyjazdy.pl/api?key=[APIKEY]
```

Serwer zwróci zestaw udostępnianych danych, na przykład:

```
"Valid": {
     "/api/miasta": "lista regionów, dla których dziś mam dostępne dane",
     "/api/przystanki/[miasto]": "lista słupków dostępnych dziś w danym mieście",
     "/api/wezly/[miasto]": "lista węzłów dostępnych dziś w danym mieście",
     "/api/rozklad/[miasto]/[przystanek]": "odjazdy z danego przystanku dziś",
     "/api/pociag/[miasto]/[trip]": "szczegółowy rozkład konkretnego pociągu"
 },
```

Serwer może także zwrócić listę zestawów, które będą udostępnione w najbliższym czasie (pole **Future**).

Zestawy danych są samoopisujące, zaś ich wykorzystanie powinno następować w kolejności od ogółu do szczegółu.

2.2 Przykładowe wykorzystanie API

Załóżmy, że chcemy umożliwić wyświetlenie rozkładu jazdy z wybranego przystanku. Najpierw należy przedstawić użytkownikowi listę miast, które są dostępne:

```
"https://przyjazdy.pl/api/miasta?key=[APIKEY]"
```

Powiedzmy, że użytkownik wybierze **Gdańsk**, identyfikator: **gdansk**.

Należy zwrócić uwagę, że w miarę możliwości dla każdej informacji dołączone będzie pole **Url** wskazujące łącze, które należy wyświetlić użytkownikowi, oraz pole **Api**, wskazujące gotowy adres URI kolejnego zapytania, które wystarczy uzupełnić o klucz API.

Następnie dla wybranego miasta należy przygotować graficzną i tekstową listę przystanków:

```
"https://przyjazdy.pl/api/przystanki/gdansk?key=[APIKEY]"
```

Z tej listy użytkownik wybierze kod swojego przystanku, załóżmy **8308_0** o nazwie **Gdynia Karwiny PKM** o lokalizacji **54.46929,18.50572**. Wybrane miasto i numer przystanku jest gwarantowane do czasu kolejnego przetwarzania nocnego, czyli w kolejnym dniu już nie jest gwarantowane. Aplikacja musi odpowiednio zareagować, jeśli w kolejnym dniu wybrana informacja nie będzie już dostępna.

Dla wybranego przystanku pobierzemy rozkład jazdy:

```
"https://przyjazdy.pl/api/rozklad/gdansk/8308_0?key=[APIKEY]"
```

Otrzymany rozkład jazdy jest na bieżący dzień i nie ulegnie zmianie aż do przetwarzania nocnego, dlatego należy go zapisać i analizować już na poziomie aplikacji. W rozkładzie będą rodzaje transportu (np. **autobus**), numery linii komunikacyjnych (np. **181**), a także relacje kursów (np. **Sopot Reja - Gdynia Kacze Buki**), numery kursów (pole trip, np. **10181D2018-10-16T31S181-14O1**).

Sformatowanie i wyświetlenie tych danych jest już po stronie programisty, na przykład jako rozkład przystankowy, tabela najbliższych odjazdów, mapa połączeń, trasa przejazdu na mapie etc.

Powiedzmy, że użytkownik zainteresuje się kursem linii **181**, odjeżdżającym o **5:15**, którego kod **10181D2018-10-16T31S181-14O1** pobraliśmy wcześniej. Pobieramy szczegóły tego kursu:

```
"https://przyjazdy.pl/api/pociag/gdansk/r10181d2018-10-16t31s181-14o1?key=[APIKEY]"
```

Warto pamiętać, że jutro tego kursu może nie być, może mieć inny kod, ale w dniu dzisiejszym te dane również pozostaną niezmienione, więc należy je zachować. Serwer zwróci pełny rozkład danego kursu:


```
"Autobus": {
    "gdansk": {
        "r10181d2018-10-16t31s181-14o1": {
            "1": {
                "id": "14481_0",
                "stacja": "Sopot Reja",
                "gps": "54.43132,18.56197",
                "dep": "04:59:00",
            },
            "2": {
                "id": "14512_0",
                "stacja": "Sopot 3 Maja",
                "gps": "54.43542,18.56141",
                "dep": "05:00:00",
            },
           ....
            "24": {
                "id": "32550_0",
                "stacja": "Gdynia Kacze Buki",
                "gps": "54.45581,18.45318",
                "arr": "05:30:00",
            }
        }
    }
}
```


Pobrawszy rozkład kursu możemy go wyświetlić na mapce w formie graficznej.

2.3 Informacje rozszerzone

W ramach API dostępny jest także pewien zakres informacji rozszerzonych (na przykład **najbliższe odjazdy** w zadanej czasoprzestrzeni), informacji statystycznych (na przykład **ilości przejazdów** na danej trasie lub prędkości handlowe), informacji usługowych (na przykład **wyszukiwanie połączenia** od-do).

Szczegółowe informacje zawarte są w menu API i mogą zależeć od wykorzystanego klucza.

2.4 Pobieranie pełnego rozkładu

Udostępnione API w żadnym razie NIE służy do pobierania pełnego rozkładu jazdy i NIE może być wykorzystywane w taki sposób. Jeśli celem jest pobranie pełnego rozkładu, należy pobrać pliki GTFS udostępniane bezpośrednio w serwisie

```
https://przyjazdy.pl/gtfs
```

Pobierając dane z tego źródła należy zwrócić szczególną uwagę na odrębne licencjonowanie, zaś przed wykorzystywaniem danych w sposób masowy konieczny jest indywidualny kontakt na adres:

```
rozklady@przyjazdy.pl
```