WPROWADZENIE DO ARCHITEKTURY REST

Dlaczego warto odRESTaurować API?

PayLane
way ahead

WSTĘP

Strony internetowe nie są już tym, czym były jeszcze kilka czy kilkanaście lat temu. Ewoluowały od prostych witryn informacyjnych do tworów dużo większych i bardziej skomplikowanych. Możemy śmiało powiedzieć, że w ciągu ostatniej dekady technologie zmieniły sposoby, w jakie zbieramy i przetwarzamy informacje, jak dowiadujemy sie o różnych rzeczach i jak się uczymy. Ale co chyba najistotniejsze– zmieniły sposób, w jaki tworzymy nowe rzeczy. Strony internetowe spełniają dziś inne funkcje, niż kilka lat temu; często nazywamy je już „aplikacjami sieciowymi" czy „usługami online'owymi". Stare definicje stron internetowych odchodzą do lamusa.

Za każdą aplikacją sieciową – od prostych platform blogowych, przez zbierające i przetwarzające dane (np. z mediów społecznościowych) do systemów płatności – stoją jej twórcy, projektanci architektury.

Deweloperzy (programiści, projektanci) wciąż tworzą nowe rzeczy i realizują swoje pomysły, by ułatwić nasze życie czy usprawnić naszą działalność w sieci. Są jednak mocno zależni od dostępnych technologii; często korzystają też z wielu różnych API , by stworzyć coś nowego. Wiąże się z tym wiele problemów, a do niedawna nie istniały proste wytyczne, które opisywałyby zasady tworzenia API – każdy robił to w sposób, który sam uważał za najlepszy.

Na szczęście dzisiaj wystarczy jedynie znać REST. Powiemy sobie czym jest ten standard, jak go używać i czy rzeczywiście zasługuje na tak wiele pochwał.

API to skrót od **Application Programming Interface**; opisuje jak poszczególne elementy lub warstwy oprogramowania powinny się komunikować. W praktyce to najczęściej biblioteka oferująca metody, które umożliwiają realizację określonych zadań.

I. CZYM JEST **REST**?

Najprościej rzecz ujmując, REST jest zbiorem reguł, których powinien przestrzegać programista. Z kolei w ujęciu bardziej technicznym, jest to wzorzec architektury oprogramowania, który opisuje jak operować zapytaniami do API i wprowadza zestaw dobrych praktyk. REST ułatwia obsługę żądań i odpowiedzi w nowy i łatwiejszy sposób, bez konieczności odwoływania się do złożonych dokumentacji.

W dzisiejszych czasach mamy styczność z różnymi API niemal na każdym kroku, choć większość z nas najczęściej nie zdaje sobie z tego sprawy. Za każdym razem, gdy klikamy Lubię to pod wpisem na blogu, odświeżamy listę tweetów w ulubionym kliencie Twittera, wyszukujemy lub oglądamy nowy film na Netflixie – wywołujemy zapytania do API. Przykładowo Twitter obsługiwał w 2011 roku 13 miliardów zapytań API dziennie! To robi wrażenie, a można przypuszczać, że od tego czasu wartości zapewne się potroiły. Między innymi z takich powodów standard REST stał się popularnych na świecie – programiści mieli dość źle napisanych API i potrzebowali czegoś, co po prostu dobrze działa.

W ciągu ostatnich kilku lat REST stał się wiodącym standardem architektury sieciowej i zastąpił swojego poprzednika – SOAP (Simple Object Access Protocol). Skąd ta rosnąca popularność?

Przyczyna jest prosta – tworzenie funkcjonalnych API jest obecnie jedną z ważniejszych części budowy aplikacji sieciowych. Architektury restowych API charakteryzuje kilka głównych zasad, które opisujemy w poniżej.

1. PROSTE RZECZY MUSZĄ POZOSTAĆ PROSTE

Nie ma najmniejszego powodu, by komplikować rzeczy, które powinny pozostać proste i łatwo rozpoznawalne, jak np. URL-e albo nazwy metod. REST zaprojektowano tak, by był możliwie najprostszy – pamiętaj o tym podczas tworzenia własnych systemów. Twoim celem jest przecież udostępnienie innym takiego interfejsu, z którego będą potrafili korzystać bez konieczności spędzania długich godzin z dokumentacją.

2. TRZYMAJ SIĘ STANDARDU, NIE ROZSZERZAJ GO NIEPOTRZEBNIE

Standard restowych API został stworzony na bazie doświadczeń pochodzących z prac nad standardem

W3C HTTP 1.1. Dlatego nie musisz tworzyć np. własnego mechanizmu uwierzytelniania; wystarczy, że skorzystasz z *HTTP Basic Auth*, by uniemożliwić dostęp nieupoważnionym osobom. Korzystaj z tego, co już jest zrealizowane i dostępne.

3. STANDARYZUJ ŻĄDANIA I ODPOWIEDZI

Powinieneś myśleć o restowych API w ten sposób: oferujesz swoim klientom dostęp do określonych metod; zapytania powinny jednoznacznie opisywać określony zasób, który może być później zmieniany.

4. PODĄŻAJ ZA TRENDAMI, ALE POZOSTAŃ ELASTYCZNY

Śledzenie i podążanie za trendami jest bardzo istotne, nawet jeśli chodzi o tak proste rzeczy, jak *media types* (typy MIME) używane podczas obsługi zapytań i odpowiedzi. Powszechnym i najczęściej używanym w restowych API standardem są struktury JSON. Warto jednak być elastycznym i jeśli klienci sobie tego życzą, udostępnić im również możliwość komunikacji z użyciem innych struktur danych, np. XML.

II. PO CO MI W OGÓLE **REST**OWE **API**?

Projektowanie i realizacja restowego API może wymagać nieco więcej czasu czy wysiłku, ale bez wątpienia takie podejście zwraca się później z nawiązką. Kiedy tworzysz aplikację sieciową i chcesz udostępnić innym programistom możliwość integracji z nią, by rozszerzać jej funkcjonalności, najlepiej udostępnić dobre API restowe wraz z dokumentacją. Taki krok to najlepszy sposób na zwrócenie uwagi klientów i programistów.

Przykłady dokumentacji API

Twitter API

Dropbox API

Zendesk API

PayLane API

Udostępnienie takiego API oznacza coś więcej niż to, że podążasz za największymi trendami dotyczącymi rozwoju architektur sieciowych i będziesz życzliwie przyjmowany wśród wielu społeczności programistycznych. To także sygnał, że jesteś otwarty na wszystkie możliwe technologie – od tych typowo sieciowych, po aplikacje desktopowe i mobilne.

Co więcej, stosowanie standardowych rozwiązań jest zawsze łatwiejsze i szybsze, niż wdrażanie nowych przy tworzeniu kolejnych produktów czy usług. W przypadku restowego API możesz używać za każdym razem tej samej klasy *wrappera* (nakładki programistycznej), której

Wrapper – uniwersalna i podstawowa klasa, która umożliwia realizację np. zapytań restowych. Możesz ją rozszerzać o metody poprzez dziedziczenie. To najbardziej nadrzędna klasa, która realizuje zapytania API do konkretnych serwerów.

używasz do komunikacji z określoną usługą (np. Twitterem). Wystarczy tylko rozszerzyć dostępną klasę i dodać do niej nowe metody opisane w odpowiednich dokumentacjach. Prawda, że brzmi prościej? Takie podejście oznacza dużo zaoszczędzonego czasu dla Ciebie i Twojego biznesu – wdrażanie nowego API może zająć nie godziny czy dni, ale minuty.

Wierzymy, że restowe API są czymś, co zmienia naturę usług sieciowych. Eliminują zbędną pracę i czas inwestowane w integrację – REST jest zwyczajnie prostszy i pozwala zrobić to samo dużo szybciej. Przyszli klienci zupełnie inaczej spojrzą na Twój produkt, gdy zaoferujesz im tego rodzaju narzędzie.

REST pozwala również na realizację tych samych zadań w dużo szybszy sposób. W nowoczesnych środowiskach programistycznych często korzysta się z wielu nowych technologii. Mogą to być *frameworki* JavaScriptowe

(np. **Meteor**) lub nierelacyjne bazy danych (np. **MongoDB**). Wspólną cechą tych technologii jest fakt, że korzystają z formatu JSON – możesz przekazać dane zwrócone przez API restowe do swojego serwisu lub bezpośrednio do bazy danych. Taki sposób postępowania jest dziś powszechny i wszyscy, którzy robią inaczej, są uważani za „branżowe dinozaury". Kiedy używasz współczesnych technologii, zazwyczaj wykorzystujesz REST.

Technologia ta została tak zaprojektowana, by złamać konwencje rozdzielającą aplikacje sieciowe, desktopowe i mobilne. Pozwala im wszystkim komunikować się prostymi zapytaniami HTTP (omówimy je za chwilę), oferuje możliwość operowania na praktycznie każdej platformie i ułatwia wymianę informacji między nimi. To potencjał okazji biznesowych dla Twoich obecnych i przyszłych klientów.

Podsumowując, restowe API może pomóc w zdobyciu nowych użytkowników i zwiększeniu zysków. Z punktu widzenia programisty – korzystanie z restowego API pozwala na płynniejszą i efektywniejszą pracę.

III. JAK KORZYSTAĆ Z **REST**OWEGO **API** – IMPLEMENTACJA

W tej sekcji poruszymy nieco technicznych aspektów, by pokazać, jak funkcjonuje REST i jak powinno się go implementować. Samo wdrożenie jest stosunkowo proste. W najprostszym przypadku wystarczy cURL lub podobne narzędzie, które umożliwi realizowanie zapytań HTTP 1.1. Na potrzeby tego tekstu, ograniczymy się właśnie do nich – w ten sposób możemy pozostać możliwie najbliżej standardu REST. Nie powinien być on bezpośrednio związany z żadnym językiem programowania, natomiast należy zachować możliwie największą prostotę.

Najpierw musimy jednak poznać podstawy. Restowe API wykorzystuje typowe metody HTTP, takie jak **POST, GET, PUT, DELETE**. Są także bardziej złożone metody, jak OPTIONS, HEAD, TRACE, CONNECT, jednak skupimy się na pierwszych czterech, ponieważ są najczęściej używane. Jeśli nie czujesz się teraz zbyt pewnie, postaramy się wyjaśnić sens tych metod. Przede wszystkim porównajmy je z typowymi operacjami CRUD. Oto jak sprawy wyglądają:

- metoda POST jest używana do tworzenia nowych rzeczy, to praktycznie odpowiednik Create z terminologii CRUD;
- GET stanowi proste zapytanie z parametrami (odpowiednik *Read*);
- metoda PUT aktualizuje lub zamienia dane, można ją traktować jako *update/replace into* z SQL lub *Update* z CRUD;
- nazwa metody DELETE mówi sama za siebie i jest oczywiście odpowiednikiem *Delete* z CRUD.

Zobaczmy teraz jak to może wyglądać na prostym przykładzie – wyobraźmy sobie, że mamy API fabryki samochodów i chcemy sprawić, by było „bardziej restowe". Na tej bazie przedstawiamy w poniższej tabelce podstawowe założenia metod HTTP – sposób operowania nimi i wynik realizacji.

Request URL	POST	GET	PUT	DELETE
/fabryka	Stwórz nowy samochód	Pobierz wszystkie samochody	Zamień/ aktualizuj wszystkie samochody	Usuń wszystkie samochody
/fabryka/ford	Stwórz nowy samochód marki ford	Pobierz wszystkie samochody marki ford	Zamień/ aktualizuj wszystkie samochody marki ford	Usuń wszystkie samochody marki **ford**

Teraz zobaczmy jak działają restowe API na praktycznych przykładach. Na potrzeby tego tekstu posłużymy się klientem REST dla Google Chrome o nazwie **Postman**. Polecamy Ci to samo – to bardzo przyjazne narzędzie. Omówimy teraz pobieżnie cztery podstawowe metody.

GET

Aby otrzymać pierwsze wyniki i poczuć magię REST, przygotujemy pierwsze zapytanie. Użyjemy do tego bardzo prostego zasobu REST, który ma tylko jedno zadanie – zwrócić numer IP. W tym celu wprowadź adres http://call.jsonlib.com/ip w pole request URL w narzędziu Postman. Wybierz metodę GET i kliknij Preview. Zobaczysz rezultat podobny do tego:

```
GET /ip HTTP/1.1
Host: call.jsonlib.com
Cache-Control: no-cache
```

Jest to proste zapytanie GET przez HTTP, które jest gotowe do wysłania na serwer. Po kliknięciu w przycisk *Send*, otrzymasz odpowiedź zawierającą kod statusu *200* (OK) oraz obiekt JSON zawierający Twój numer IP – w naszym przypadku wyglądało to tak:

```
{
    "ip" : "89.206.36.193"
}
```

Gratulacje, właśnie wykonałeś pierwszą operację restową! A teraz, gdy już znamy podstawy, pora na coś nieco bardziej złożonego.

POST

Wybierz w Postmanie metodę zapytania POST i wprowadź nowy URL (https://call.jsonlib.com/echo) w polu adresu. Gdy zmienisz metodę z GET na POST, pojawi się nowy formularz – to prosta reprezentacja obiektu JSON, który wyślesz do serwera. Wprowadź dowolne dane. Możesz podejrzeć zapytanie klikając w przycisk *Preview* i zobaczyć, jak serwer będzie „widział" Twoje żądanie; kliknij przycisk *Send*, by wysłać zapytanie. Nasze dane są oczywiście inne, niż Twoje, niemniej serwer powinien odpowiedzieć tym samym zestawem danych, które wprowadziłeś w Postmanie. W naszym przypadku odpowiedź wyglądała następująco:

```
{
    "lastname" : "Banks",
    "id"       : "1",
    "name"     : "Tom"
}
```

PUT & DELETE

Nie ma prostego sposobu, by przetestować te dwie metody, a więc załóżmy po prostu, że mamy wpis w platformie blogowej i chcemy zmienić jego tytuł. Użyjemy do tego metody PUT, która została stworzona właśnie po to, by aktualizować/zastąpić dane. Przykładowy obiekt JSON, który mógłby być użyty w zapytaniu, może wyglądać następująco:

```
{
    "title" : "This is a brand new title of my post."
}
```

Zakładamy, że adresem, na który trzeba wysłać zapytanie, jest *blogpost/{post_id}*. W chwili wysłania tego żądania z użyciem metody PUT, serwer powinien rozpoznać tytuł wpisu i nadpisać nim tytuł wpisu identyfikowanego numerem *post_id*.

Podobna koncepcja sprawdzi się w przypadku metody DELETE. Struktura API powinna umożliwiać wysłanie żądania bez żadnych danych – manipulowalibyśmy jedynie samym adresem. Przykładowo zapytanie wysłane na adres *https://api.yourbusiness.com/blogpost/{post_id}*, gdzie *{post_id}* jest zmienną, powodowałoby usunięcie wpisu identyfikowanego podanym w adresie numer ID.

Oczywiście uwierzytelnianie (HTTP 1.1 Basic Auth) jest w tego rodzaju zastosowaniach nieodzowne – nie chcielibyśmy przecież, by ktoś, kto tylko poznał odpowiedni adres naszego API, mógł usunąć nasz ulubiony wpis o kotach, prawda?

IV. **API REST**OWE – PRAKTYCZNY PRZYKŁAD

Gratulacje, dotarłeś już całkiem daleko w naszym whitepaperze! Teraz, gdy już wiesz o co chodzi w REST, mamy przykłady z prawdziwego zdarzenia – mamy nadzieję, że przekonają Cię, by używać tej technologii w swoich projektach. Do poniższego przykładu potrzebna nam będzie linia poleceń oraz cURL.

Zacznijmy od przygotowania - załóżmy konto testowe na stronie PayLane.pl i zapiszmy dane dostępowe do API. Teraz możemy spojrzeć na opis funkcji API (dostępny na Developer Zone) i wybrać jedną z metod, której spróbujemy użyć do realizacji prostego zapytania. W naszym przykładzie użyjemy jednej z popularniejszych metod – *cards/sale*.

POST cards/sale

Resource URL: `POST https://direct.paylane.com/rest/cards/sale`

Metoda `cards/sale` umożliwia przeprowadzenie pojedynczej płatności z użyciem karty (kredytowej, debetowej, przedpłaconej itd.). Uwzględnia dodatkowe mechanizmy bezpieczeństwa, takie jak *AVS* czy *fraud check*.

Wiemy zatem co ta metoda robi i potrzebujemy już tylko odpowiedniej struktury, której użyjemy w naszym zapytaniu. Wystarczy, że skopiujemy podany w dokumentacji przykład.

Następnie musimy przygotować odpowiednie zapytanie cURL, które zostanie wysłane do systemów PayLane. W tym przykładzie zrobiliśmy już to za Ciebie. Oczywiście nadal możesz używać Postmana – my wykorzystujemy teraz cURL, ponieważ w ten sposób dużo prościej zademonstrować to pojedyncze zapytanie do API. Jednocześnie pokazuje to nam, że technologia REST jest w zasadzie niezależna od programów czy bibliotek, których używamy do komunikacji.

Wróćmy jednak do zapytania – mamy gotową strukturę JSON, którą skopiowaliśmy z dokumentacji:

```
{
    "sale"      :   {
                    "amount"          :   100.00,
                    "currency"        :   "EUR",
                    "description"     :   "A brilliant product, #52704",
                    "fraud_check_on"  :   true,
                    "avs_check_level" :   2
    },
    "customer"  :   {
                    "name"    :   "John Doe",
                    "email"   :   "john@doe.com",
                    "ip"      :   "123.456.78.90",
                    "address" :   {
                                "street_house" :   "1600 Pennsylvania Avenue Northwest",
                                "city"         :   "Washington",
                                "state"        :   "DC",
                                "zip"          :   "20500",
                                "country_code" :   "US"
                    }
    },
    "card"      :   {
                    "card_number"       :   "4111111111111111",
                    "expiration_month"  :   "03",
                    "expiration_year"   :   "2017",
                    "name_on_card"      :   "John Doe",
                    "card_code"         :   "123"
    }
}
```

Musimy teraz jedynie przygotować poprawne zapytanie cURL, skorzystać z danych dostępowych do API i wywołać całość w terminalu.

```
curl --request POST 'https://test_login:test_password@direct.paylane.com/rest/cards/sale' \
     --data '{"sale":{"amount":19.99,"currency":"EUR","description":"Product
#1"},"customer":{"name":"John Doe","email":"john@doe.
com","ip":"127.0.0.1","address":{"street_house":"1600 Pennsylvania Avenue
Northwest","city":"Washington","state":"DC","zip":"500","country_code":"US"}},"card":{"card_
number":"4111111111111111","expiration_month":"03","expiration_year":"2017","name_on_
card":"John Doe","card_code":"123"}}' \
     -i \
     --header 'Content-type:application/json'
```

Ostatni parametr – *header* – nie jest obowiązkowy, jednak podaliśmy go, aby upewnić się, że dane, które wysyłamy, są w formacie JSON.

Z chwilą, gdy realizujemy przygotowane zapytanie, żądanie HTTP POST jest wysyłane do API i system tworzy transakcję – w naszym przypadku odpowiedź serwera jest jednoznaczna:

```
{
    "success"     :   true,
    "id_sale"     :   1234567,
    "avs_result"  :   "M",
    "fraud_score" :   10
}
```

Udało się i przyjęliśmy fikcyjną płatność od równie fikcyjnego klienta. Otrzymaliśmy identyfikator transakcji, który możemy zapisać w swoim systemie i wykorzystać do innych operacji z użyciem API w przyszłości (np. ponownej płatności czy pozyskania szczegółowych informacji o transakcji).

PODSUMOWANIE

Mamy nadzieję, że ten krótki dokument przybliżył Ci nieco tematykę technologii REST oraz restowych API. Najprostszym podsumowaniem jest to, że REST został zaprojektowany jako prosty standard sieciowy, który wykorzystuje podstawowe zapytania HTTP i pomaga deweloperom na efektywną wymianę informacji między różnymi usługami czy zasobami. Dzięki prekursorom restowych API, takim jak Twitter, Facebook czy Netflix, dochodzimy wreszcie do momentu, kiedy integracja z nowymi usługami nie stanowi już problemu, a my sami możemy pracować wydajniej.

Udanego od**REST**aurowywania! ;)

Ekipa PayLane

ŹRÓDŁA I ZASOBY

Statystyki zapytań do API
http://blog.programmableweb.com/2011/05/25/who-belongs-to-the-api-billionaires-club/

Postman dla Chrome:
https://chrome.google.com/webstore/detail/postman-rest-client/fdmmgilgnpjigdojojpjo-ooidkmcomcm

Teach a dog to REST:
https://blog.apigee.com/detail/restful_api_design

Beginners guide to creating a REST API:
http://www.andrewhavens.com/posts/20/beginners-guide-to-creating-a-rest-api

call.jsonlib:
http://call.jsonlib.com

Dokumentacja PayLane:
http://devzone.paylane.pl

Tworzenie konta testowego na potrzeby przykładu:
http://paylane.pl/utworz-konto/

OPRACOWAŁ MACIEJ KOŁEK

Maciej pracuje jako **Web Application Developer** w PayLane.
Poza programowaniem interesuje się wszystkimi sprawami związanymi z technologiami mobilnymi oraz ich biznesowym aspektem.
Chętnie poznaje nowe technologie usprawniające tworzenie aplikacji.
Jeśli masz jakieś pytania do Maćka, znajdź go na Twitterze: @ferusinfo.