

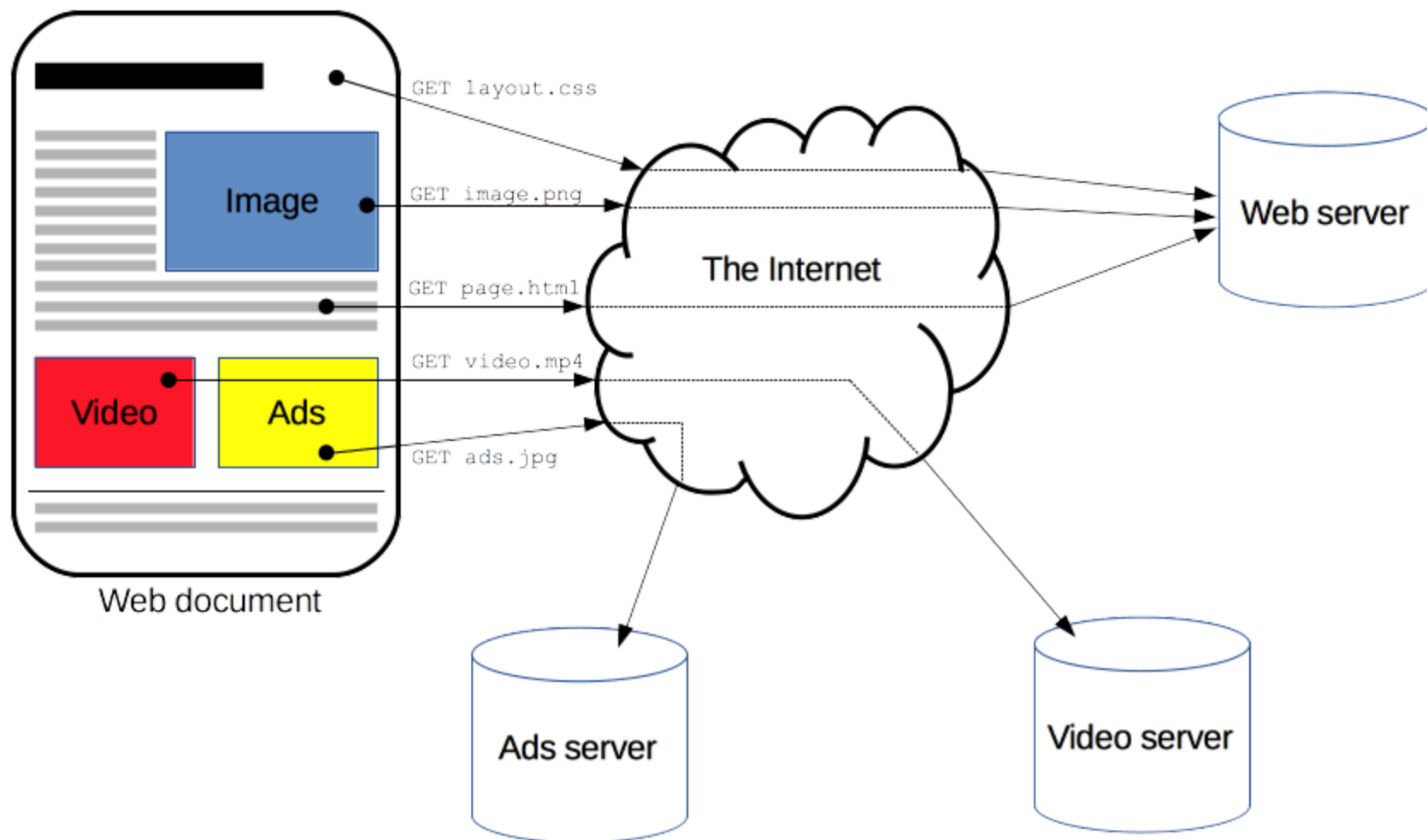
Programowanie Aplikacji Internetowych

API / Komunikacja między serwisami

Plan na dziś

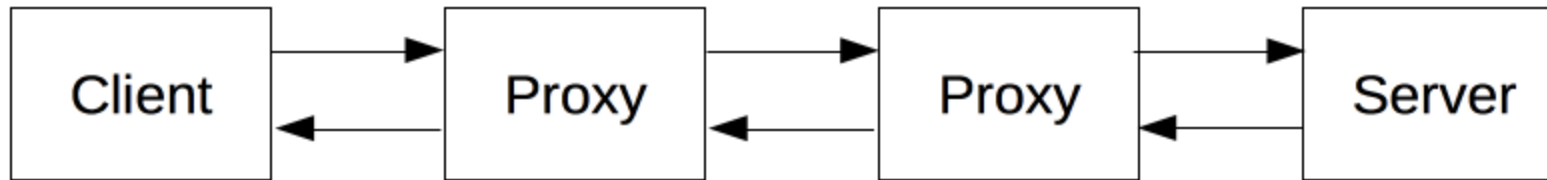
- http
- RPC
- REST
- GraphQL

HTTP



HTTP

Cała infrastruktura przystosowana do pracy z http:



HTTP

Demo:

```
curl -I www.google.com
```

```
curl -I -L google.com
```

HTTP - methods

Methods:

- GET
- POST
- PUT
- DELETE

HTTP - status code

Status code:

- 5xx: 500, 502
- 4xx: 404, 400, 401
- 3xx: 301, 302
- 2xx: 200, 201, 02

Warto wiedzieć, gdzie jest błąd.

WebSockets

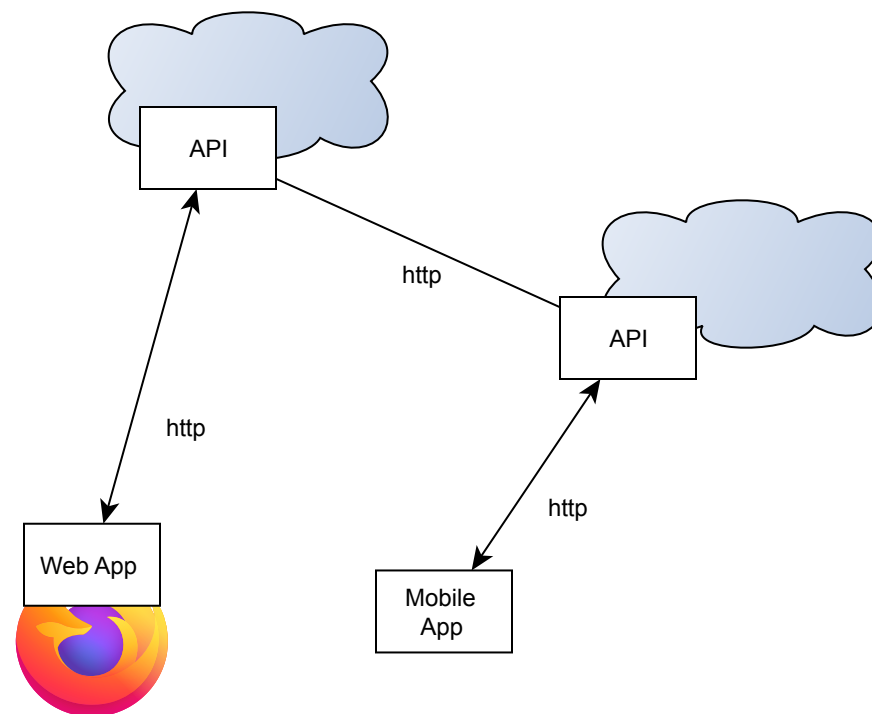
- dwustronnej szybkiej komunikacji
- Alternatywa dla *long polling*

Więcej później o [websocketach i socketio](#) później.

A co z serwisami?

- +/- Wiemy jak działają przeglądarki
- co z serwisami?

A co z serwisami?



Protokoły

Najpopularniejsze:

- (web) RPC
- REST API
- GraphQL

(web) RPC

- RPC (remote procedure call)
- po prostu wywołanie zewnętrznej funkcji

(web) RPC

Przykłady / o czym należy pamiętać wywołując zewnętrzny serwis:

- [example_py_call_rest_api](#)
- [example_js_call_rest_api](#)
- [example_js_call_rest_api](#)

REST API

- Inspiracja: jak działa komunikacja między przeglądką, a serwerem,
- Istniejąca infrastruktura,
- Najbardziej popularne podejście.

Przykład - Github

- [commits](#)
- [prs](#)
- [authentication](#)

Przykład - Github

Co warto sprawdzić:

- [verbs](#)
- [errors](#)
- [rate limiting](#)

Przykład - Github

Często mamy już dostępne biblioteki:

- oficjalne - <https://github.com/octokit>
- nieoficjalne - <https://github.com/google/go-github>

Jeśli budujesz API

Warto się wzorować na:

- shopify API - [przykład](#),
- twilio - <https://www.twilio.com/docs/usage/api>,
- ably - <https://ably.com/docs/api/rest-api#publish>,
- [dobre praktyki](#).

REST API

Zasady:

- Logical organization of resources
- Logical nesting
- Stateless
- cacheable data
- większości JSON-based

REST API

- Pragmatyczne vs dogmatycznie ([HATEOAS](#))
- Własne często ad hoc

REST API

Implementacje:

- własny
- [json API](#)
- [OpenAPI](#) - wsparcie dla generacji kodu i discovery

REST API

Wiele godzin rozstało przepalone na dyskusjach co to jest REST API i czy dane API jest rzeczywiście REST...

REST API

Dla purystów -

Wyzwania REST API

- musimy składać dane po stronie klienta
- za każdym razem backend musi pisać API dla frontendu
- jak można przewidzieć co frontend potrzebuje...

GraphQL

- <https://graphql.org/>
- <https://graphql.org/learn/>

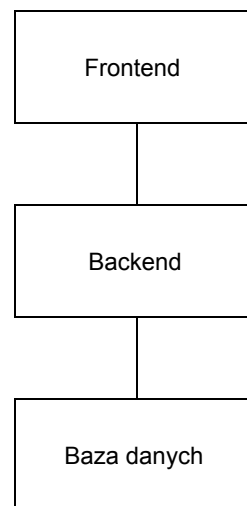
Narzędzia

- [insomnia](#) lub [postman](#)
- [curl](#)
- [jq](#)
- biblioteki [jmespath](#)

Warto wiedzieć

- gRPC
- [OData](#) - less popular

Architektura



Zauważ

- JS/TS w przeglądarce to też aplikacja,
- Docelowo - JS/TS powinna komunikować się przez API.

Dziękuję za uwagę

Backup slides

3-tier architecture

Jak hostować?

- PaaS: [vercel](#), [netify](#), [heroku](#);
- CaaS (AWS EKS, GCP) - container-as-a-service
- XaaS (AWS, GCP):
 - IaaS