GDAŃSK UNIVERSITY
OF TECHNOLOGY

Internet Services Architectures
Microservices

Szymon Olewniczak

szyolewn@pg.edu.pl

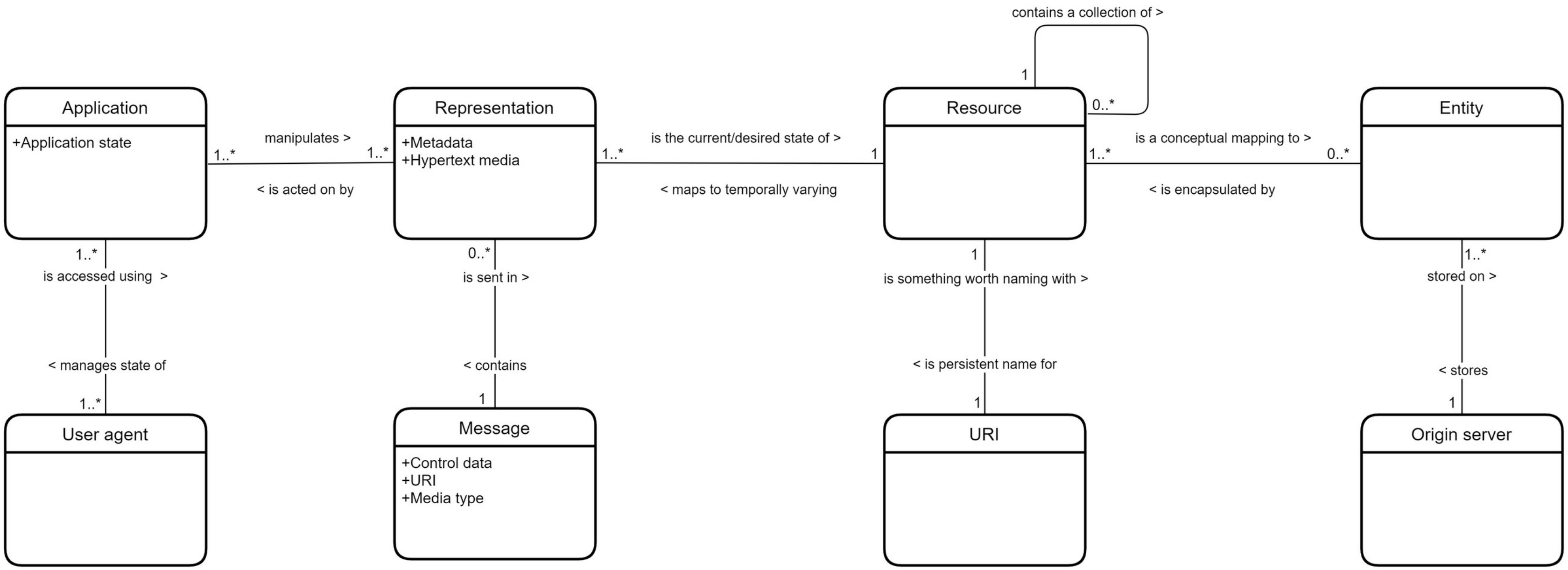
Origin of the idea: Plan 9?

- Plan 9 - distributed operating system developed as an UNIX replacement in Bell Labs since mid-1980s.
- The main idea was to take the best ideas from UNIX and move them to the networking era.
- Plan 9 is built on top of two powerful metaphors:
  - *everything is a file* - much of the interaction with the machine is done in terms of reading and writing virtual files mounted in the file system.
  - every file (or directory) can be shared over network using common 9P protocol.
- Many applications in Plan 9 are developed as so called *file servers* that exposing their logic using hierarchical virtual file systems. They can be exposed to network using 9P protocol what in practice can be treated as a form of a microservice.

GDAŃSK UNIVERSITY OF TECHNOLOGY

Representational state transfer: REST

- Defined in 2000 by Roy Fielding in his PhD dissertation: "Architectural Styles and the Design of Network-based Software Architectures".
- Postulates to create a layer of abstraction called resources that encapsulate entities (file, image, but also virtual like: "today's weather in Gdańsk") for hiding the underlying implementation details.
- The client requests a resource using a URI and the server responds with a representation of the resource in hypertext format.



Source: https://en.wikipedia.org/wiki/Representational_state_transfer

REST architectural constraints

- Client–server architecture - we separate the user interface concerns from the storage concerns.
- Statelessness - no session information is retained by the server.
- Cacheability - response must explicitly define themselves as either cacheable or non-cacheable.
- Layered system - client cannot tell whether it is connested direct to the server or to an intermediary.
- Code on demand (optional) - servers can temporarily extend or customize the functionality of a client by transferring executable code (JavaScript)
- Uniform interface:
  - Resource identification in requests - resources are conceptually separate from representations.
  - Resource manipulation through representations
  - Self-descriptive messages - we now how to process a messge (for example media type defining the message format)
  - Hypermedia as the engine of application state (HATEOAS) - having accessed an initial URI for the REST application—analogous to a human Web user accessing the home page of a website—a REST client should then be able to use server-provided links dynamically to discover all the available resources it needs

HTTP-based RESTful API

- Web API that obeys the REST constraints is informally described as RESTful.
- It make use of HTTP methods: GET, POST, PUT, PATCH, DELETE, OPTIONS.
- The most popular information extend format are JSON and XML.

Richardson Maturity Model

- Level Zero:
  - Single URI and a single HTTP method (typically POST).
- Level One:
  - Many URIs but only a single HTTP verb – generally HTTP POST.
- Level Two:
  - Many URIs and all HTTP verbs for CRUD operations.
  - We need a documentation to understand the API.
  - Most popular one among Web RESTful services.
- Level Three:
  - The API is self-descriptive by using HATEOAS.

HATEOAS example

```
GET /accounts/12345 HTTP/1.1
Host: bank.example.com

HTTP/1.1 200 OK
{
    "account": {
        "account_number": 12345,
        "balance": {
            "currency": "usd",
            "value": 100.00
        },
        "links": {
            "deposits": "/accounts/12345/deposits",
            "withdrawals": "/accounts/12345/withdrawals",
            "transfers": "/accounts/12345/transfers",
            "close-requests": "/accounts/12345/close-requests"
        }
    }
}
```

SOAP

- Appeared in 1998 initially as XML-RPC.
- REST is not a standard but architectural style.
- SOAP is a protocol specification for exchanging structured information.
- Can be considered Level Zero in RMM - single URI with single method - all the information stored in XML.
- Maintained by XML Protocol Working Group - part of W3C until 2009.
- Underlying layer for Web Services Description Language - WSDL
- Should be considered legacy now.

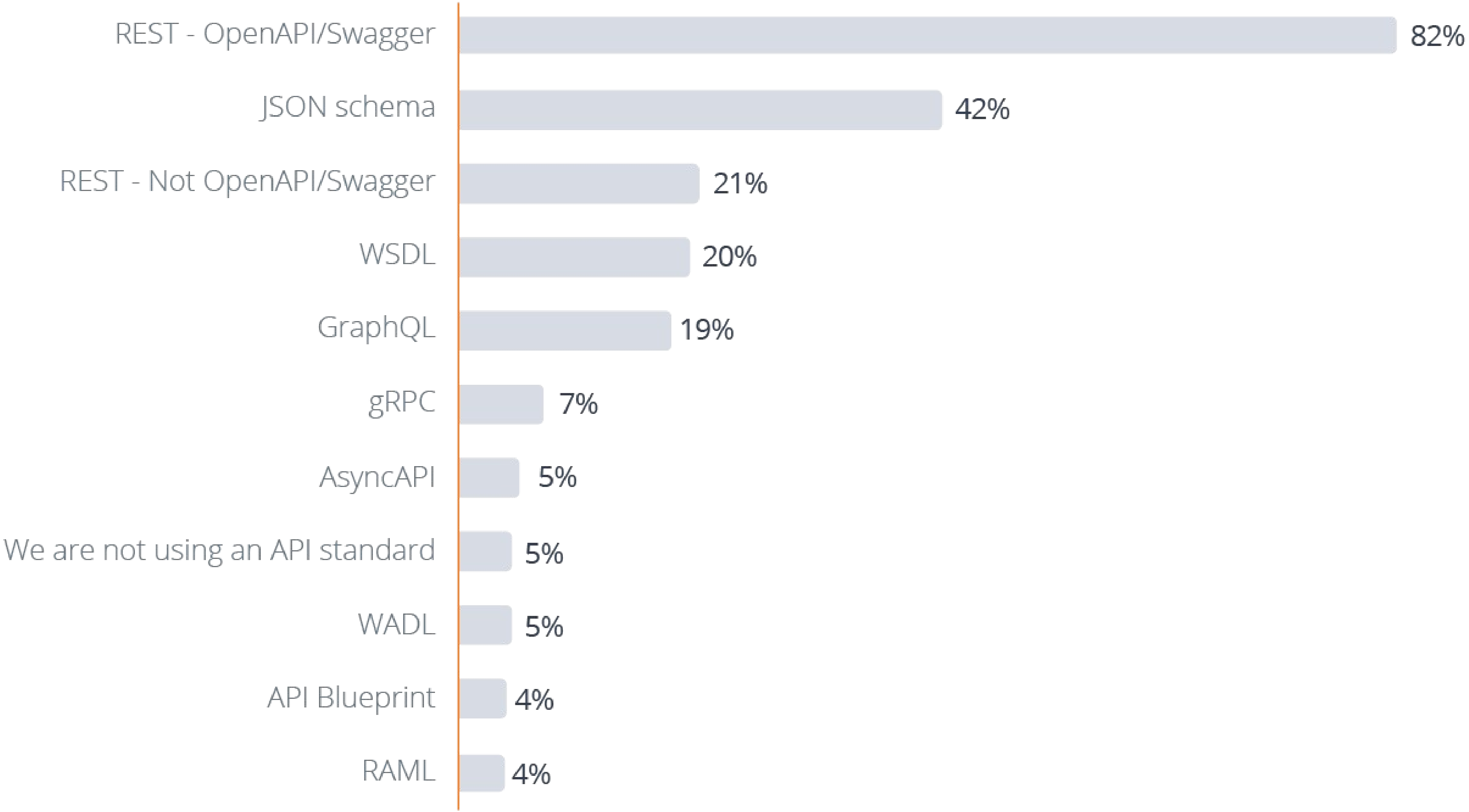
gRPC

- Released by Google in 2015.
- Build on top of HTTP/2 spec:
  - gRPC has limited browser support because numerous browsers (usually the older versions) have no mature support for HTTP/2. So, it may require gRPC-web and a proxy layer to perform conversions between HTTP 1.1 and HTTP/2. Therefore, at the moment, gRPC is primarily used for internal services.
- Uses Protocol Buffers to encode requests and responses (lower overhead)
- Good alternative to RESTful API:
  - When we need performance:
    - REST utilizing HTTP 1.1 requires a TCP handshake for each request. Hence, REST APIs with HTTP 1.1 can suffer from latency issues.
    - gRPC relies on HTTP/2 protocol, which uses multiplexed streams - several clients can send multiple requests simultaneously without establishing a new TCP connection for each one. Also, the server can send push notifications to clients via the established connection.
  - When the bi-directional communication is important (full-duplex streaming).

GraphQL

- Data query and manipulation language for APIs published by Facebook in 2015.
- Was developed to solve common disadvantage of REST - When we request the data from the endpoint we usually get the entire resource, not the data we exactly need. This can impact the application performance, mostly on mobile devices.
- Similarly to RESTful service transfers the data using HTTP.
- Requires the definition of structure of data stored on the server in the form of graph.
- https://graphql.org/learn/queries/

GraphQL



What API do you use in your company? Source: https://nordicapis.com/breaking-down-smartbears-2020-state-of-api-report/

GDAŃSK UNIVERSITY
OF TECHNOLOGY

HISTORY IS WISDOM
FUTURE IS CHALLENGE