

Tehnologii Web

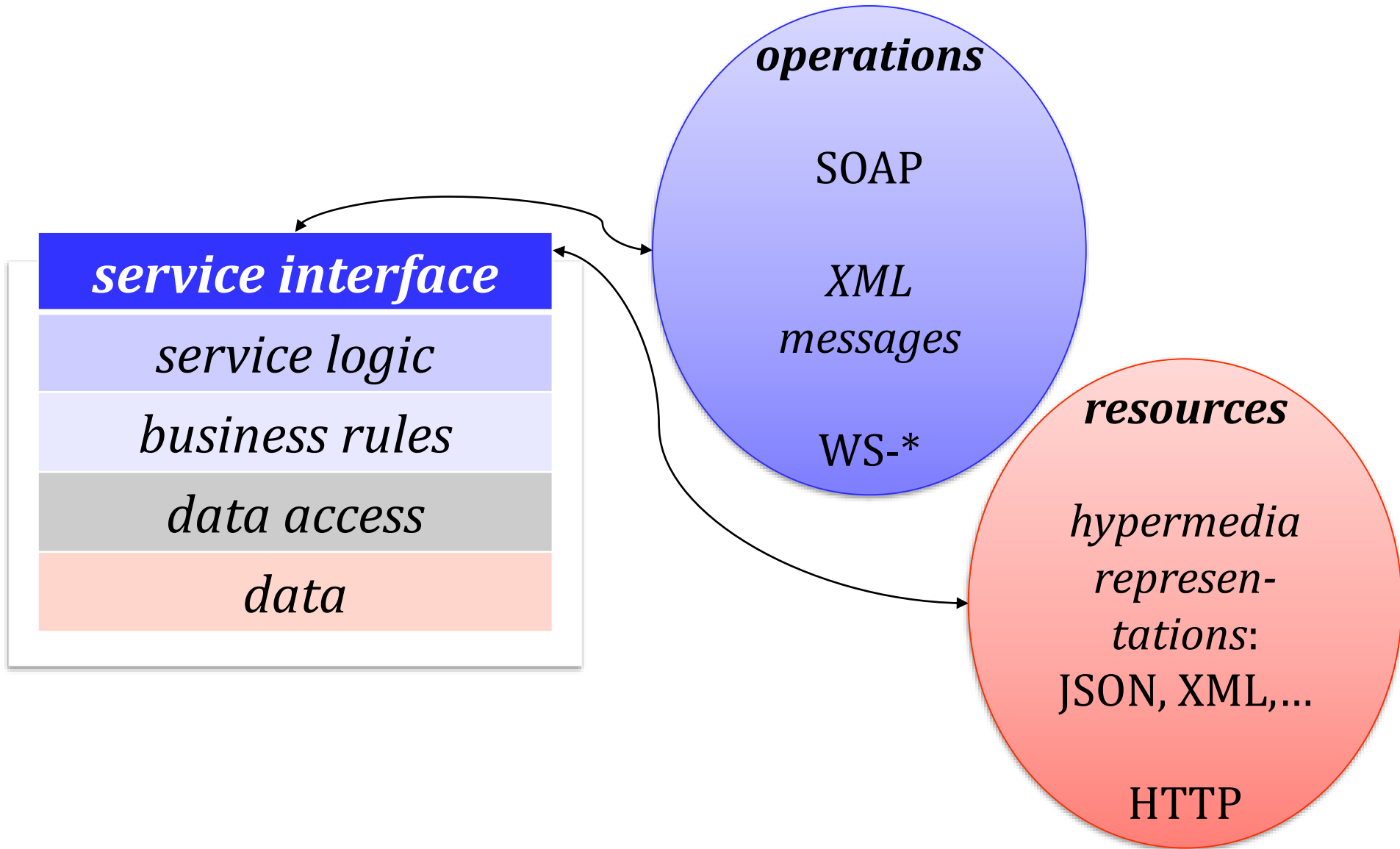


servicii Web (II)

dezvoltarea de aplicații Web prin **REST**

„Concizia este sora talentului.”

Anton Cehov



serviciile Web pot fi dezvoltate via **SOAP** și/sau **REST**

Stefan Tilkov, *REST: Not an Intro* (2013)

speakerdeck.com/stilkov/rest-not-an-intro-1

Există o modalitate de creare/invocare
a serviciilor Web fără a recurge la SOAP?

rest: representational state transfer

Stil arhitectural de dezvoltare a aplicațiilor Web
axat asupra reprezentării datelor

Roy Fielding

teză de doctorat, 2000 (*University of California, Irvine*)

www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm

rest

Rezultatul unei procesări conduce la obținerea unei **reprezentări** a unei resurse

rest

Rezultatul unei procesări conduce la obținerea unei **reprezentări** a unei resurse

resursă Web

utilizator având cont în cadrul unui sistem,
mesaj al unei persoane, fotografie, flux de știri,
componentă software, set de date (*dataset*), model 3D,...

rest

Rezultatul unei procesări conduce la obținerea unei **reprezentări** a unei resurse

reprezentare pe baza unui format de date

textual sau binar

exemple tipice – formate deschise:
HTML, JSON, CSV, PNG, SVG, PDF etc.

rest

Rezultatul unei procesări conduce la obținerea unei **reprezentări** a unei resurse

reprezentare pe baza unui format de date

formatul reprezentării e desemnat de **tipuri MIME**
text/html, text/xml, text/csv, application/json, image/png

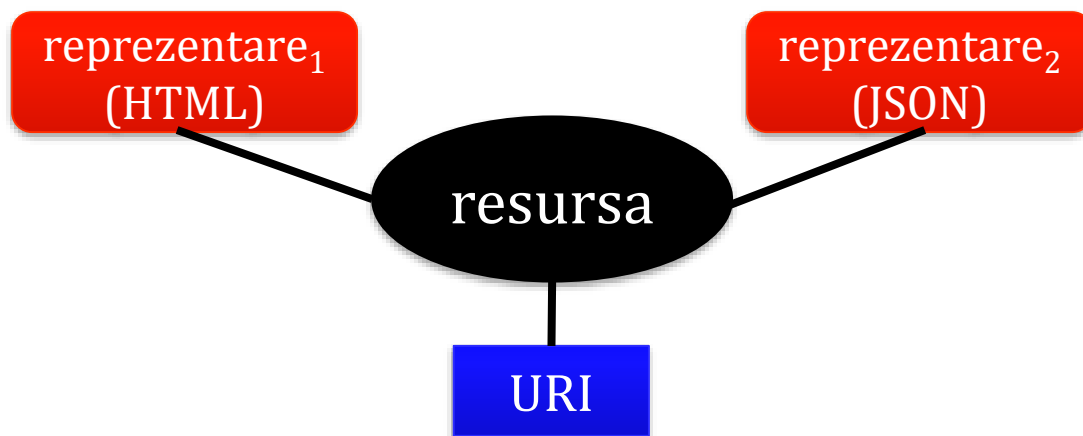
detalii în N.Freed *et al.*, *Media Types*, mai 2019

www.iana.org/assignments/media-types/media-types.xhtml

rest

Rezultatul unei procesări conduce la obținerea unei **reprezentări** a unei resurse

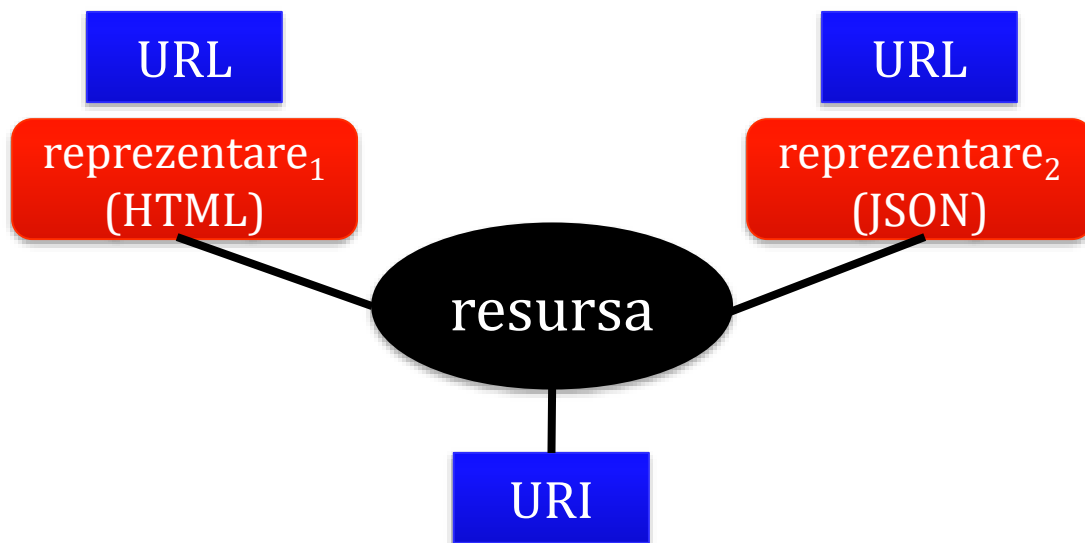
- reprezentările aceleiași resurse
- desemnate de un URI unic – pot fi multiple



rest

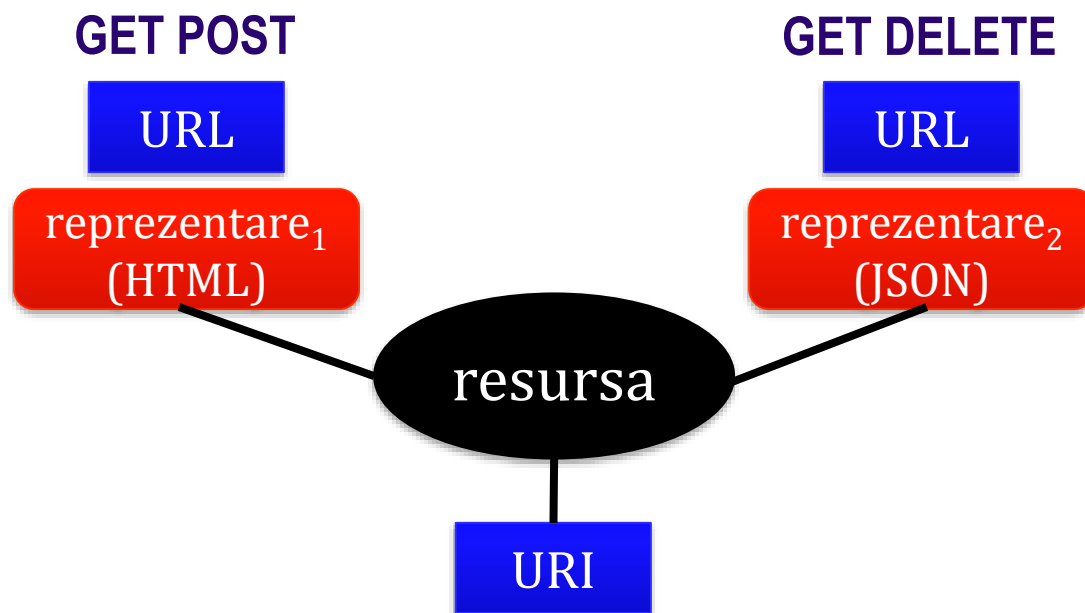
Rezultatul unei procesări conduce la obținerea unei **reprezentări** a unei resurse

fiecare reprezentare a unei resurse are asociat un URL



rest

Clienții (*e.g.*, navigatoare Web, roboți, *player*-e etc.) interacționează cu reprezentările resurselor via **verbe** „accesează”: **GET**, „modifică”: **POST**, „șterge”: **DELETE**,...



rest

Verbele (acțiunile) sunt stipulate de un protocol

de obicei, **HTTP** (*HyperText Transfer Protocol*)



de (re)parcurs
cursul al doilea

rest

GET

accesează (preia) o reprezentare a unei resurse

nu conduce la modificarea stării serverului – *safe*

idempotentă – cereri identice vor conduce la oferirea
aceluiași răspuns (aceeași reprezentare)

rest

HEAD

similară cu GET, dar furnizează doar meta-date
(nu oferă reprezentarea propriu-zisă)

e.g., ultima actualizare, lungimea conținutului,...

rest

PUT

înlocuiește (actualizează) o reprezentare de resursă sau
eventual creează o resursă la nivel de server Web
(al cărei URI e deja cunoscut)

uzual, returnează URI-ul resursei

nu e considerată *safe*, dar este idempotentă

rest

PATCH

permite actualizarea parțială a unei reprezentări
a unei resurse (PUT nu oferă o asemenea facilitate)

nu este *safe* și nici idempotentă

rest

POST

creează o resursă (uzual, subordonată altei resurse)

opțional, pot fi realizate procesări suplimentare

nu este nici *safe*, nici idempotentă

clientul nu cunoaște *a-priori*
care va fi URI-ul resursei ce va fi create

rest

DELETE

șterge (elimină) o resursă desemnată de un URI

este idempotentă

rest

OPTIONS

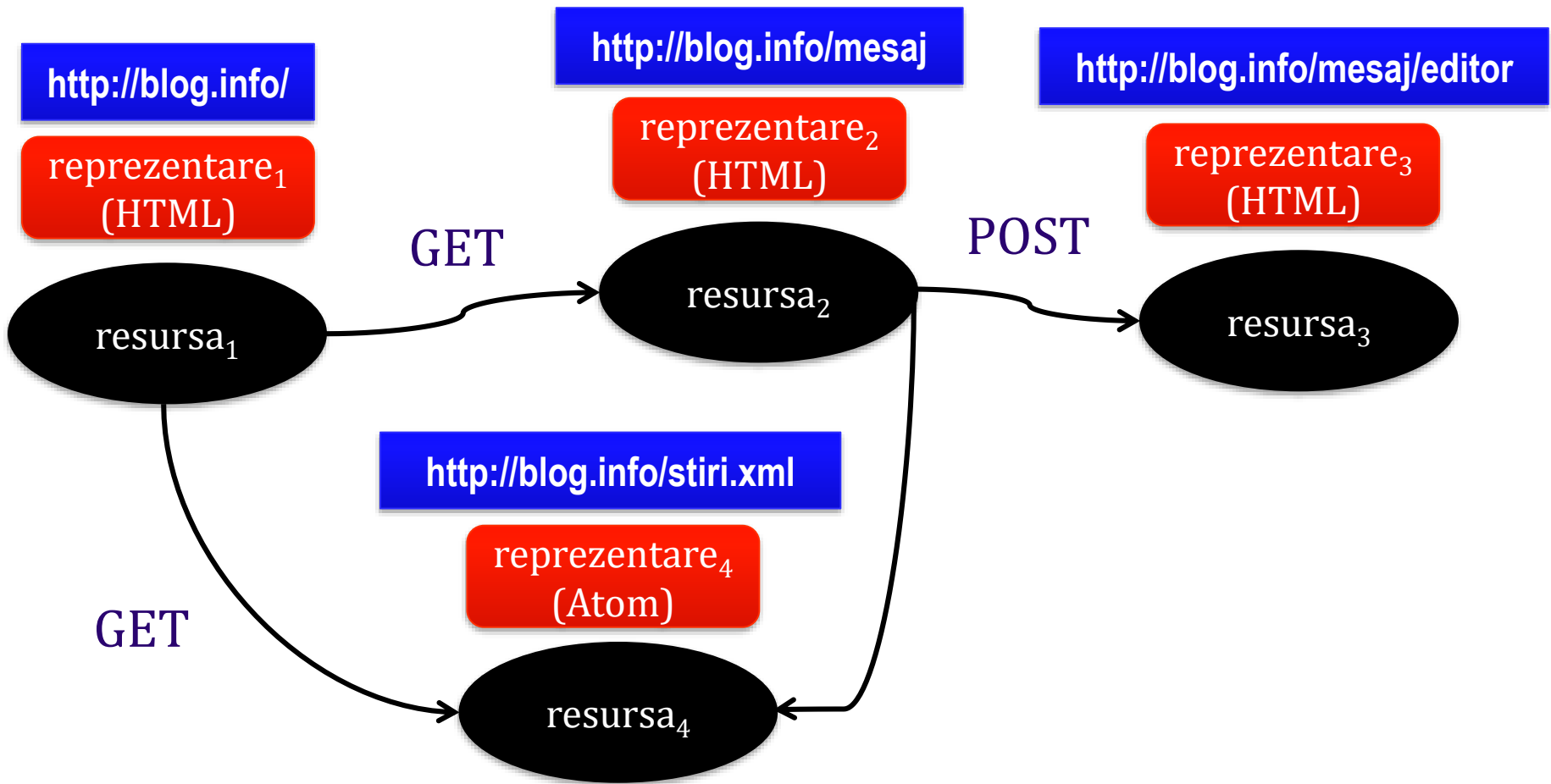
permite clientului să determine diverse cerințe privitoare
la o resursă (*e.g.*, dacă o resursă poate fi ștearsă)
sau facilitățile expuse de un server
(de exemplu, suportul oferit de un *proxy*)

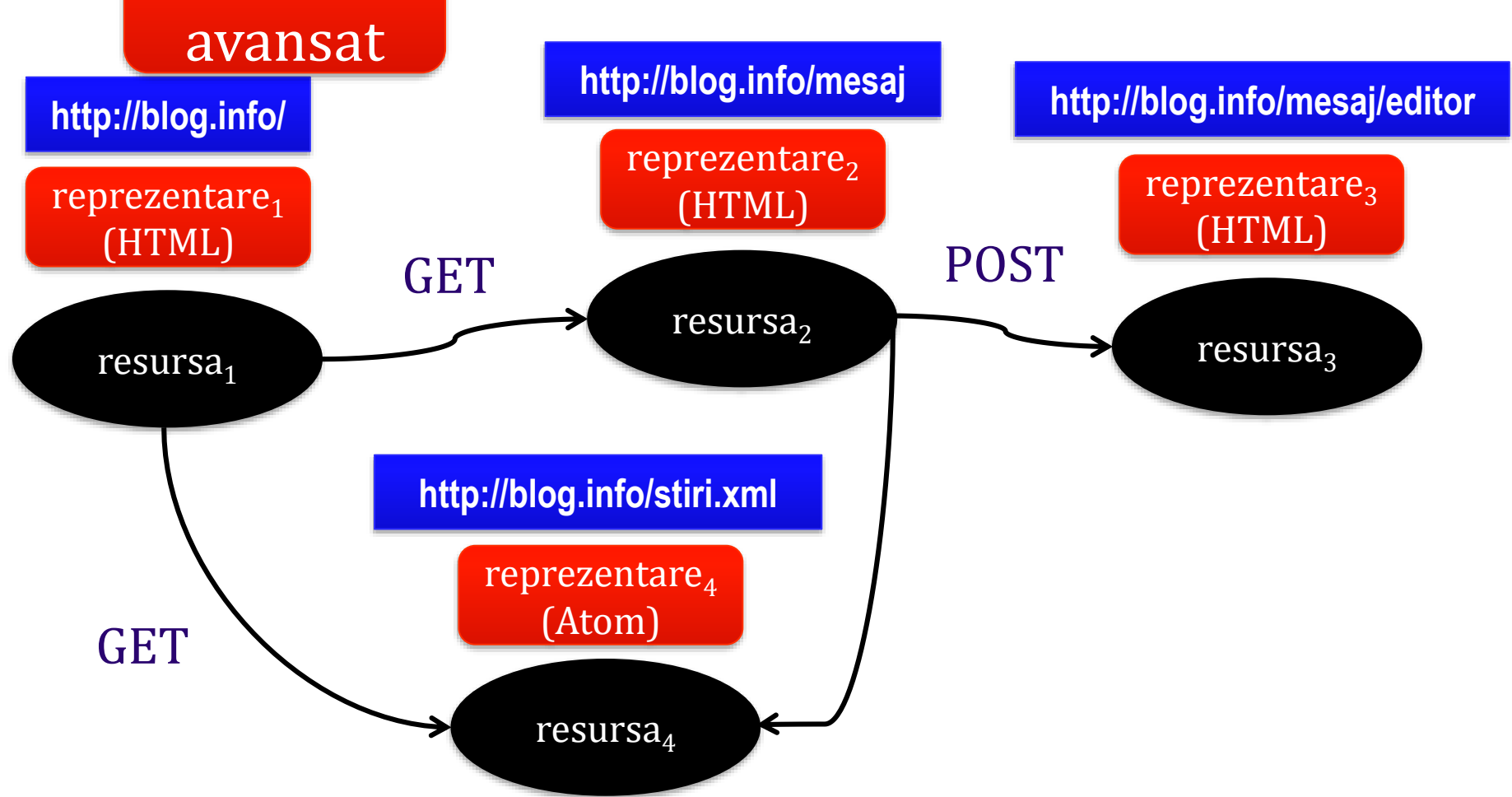
Metoda HTTP	<i>Idempotent</i>	<i>Safe</i>
GET	✓	✓
POST	X	X
PUT	✓	X
PATCH	X	X
OPTIONS	✓	✓
HEAD	✓	✓
DELETE	✓	X

de studiat și articolul Tamas Piros,
RESTful API Design – POST vs PUT vs PATCH, 2018
fullstack-developer.academy/restful-api-design-post-vs-put-vs-patch/

rest

Orice accesare a unei reprezentări
plasează aplicația – ori clientul Web – într-o **stare**
ce va fi schimbată în urma unui **transfer** de date
(accesarea altei reprezentări)





HATEOAS (*Hypermedia As The Engine Of Application State*)

B. Doerrfeld, *Designing a True REST State Machine* (2018)
nordicapis.com/designing-a-true-rest-state-machine/

rest

Transferul se realizează prin protocolul **HTTP**

Reprezentarea este modelată conform unui format

– *e.g.*, **JSON** sau **XML** – și
indicată prin tipuri **MIME** (*media types*)

Adresabilitatea se rezolvă via **URI**

rest

Aplicațiile care invocă funcționalități (servicii) consumă reprezentări de resurse – în stilul *pull*

rest

Fiecare cerere este considerată independentă,
fără a se lua în calcul contextul

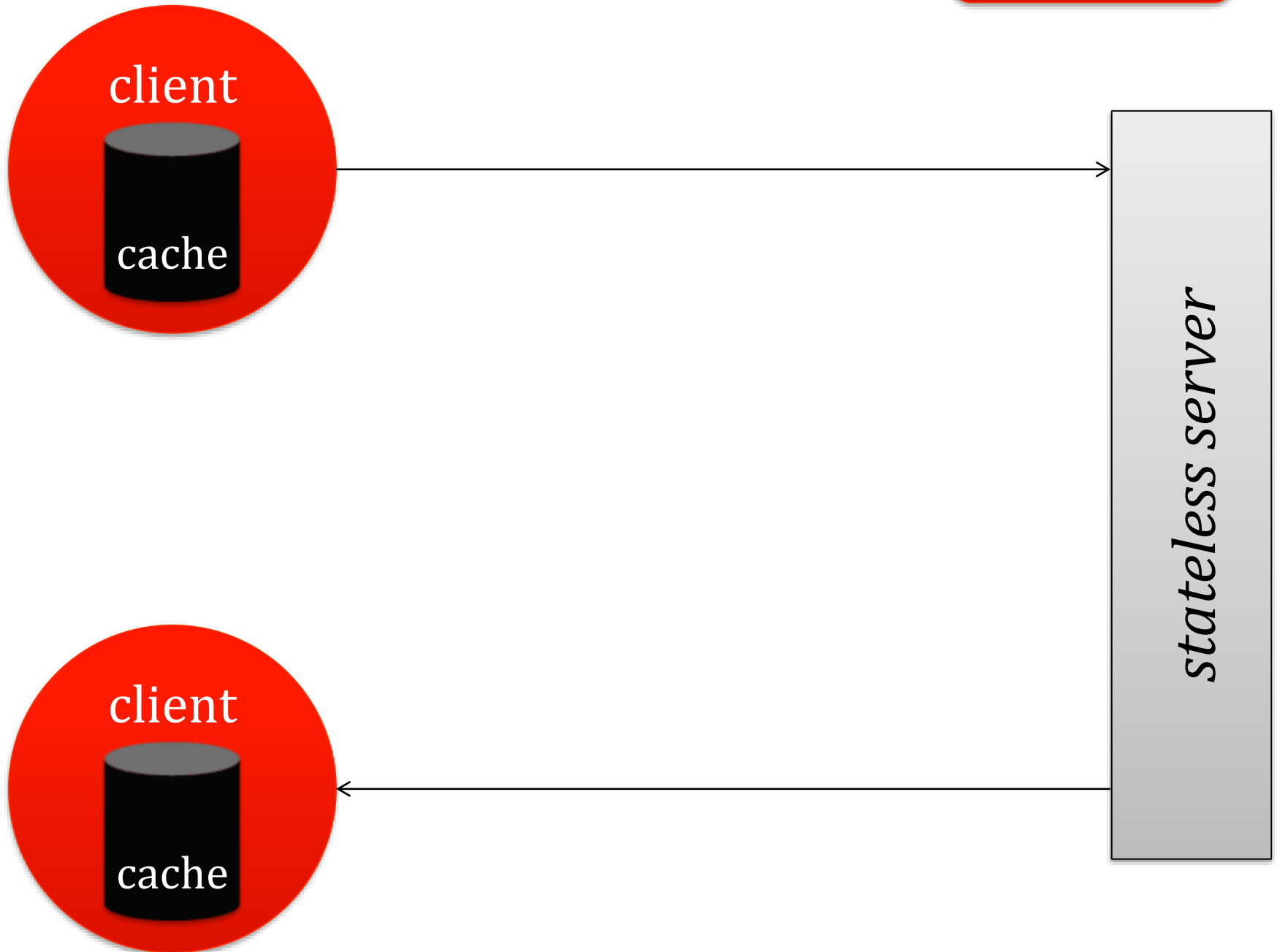
stateless server

rest

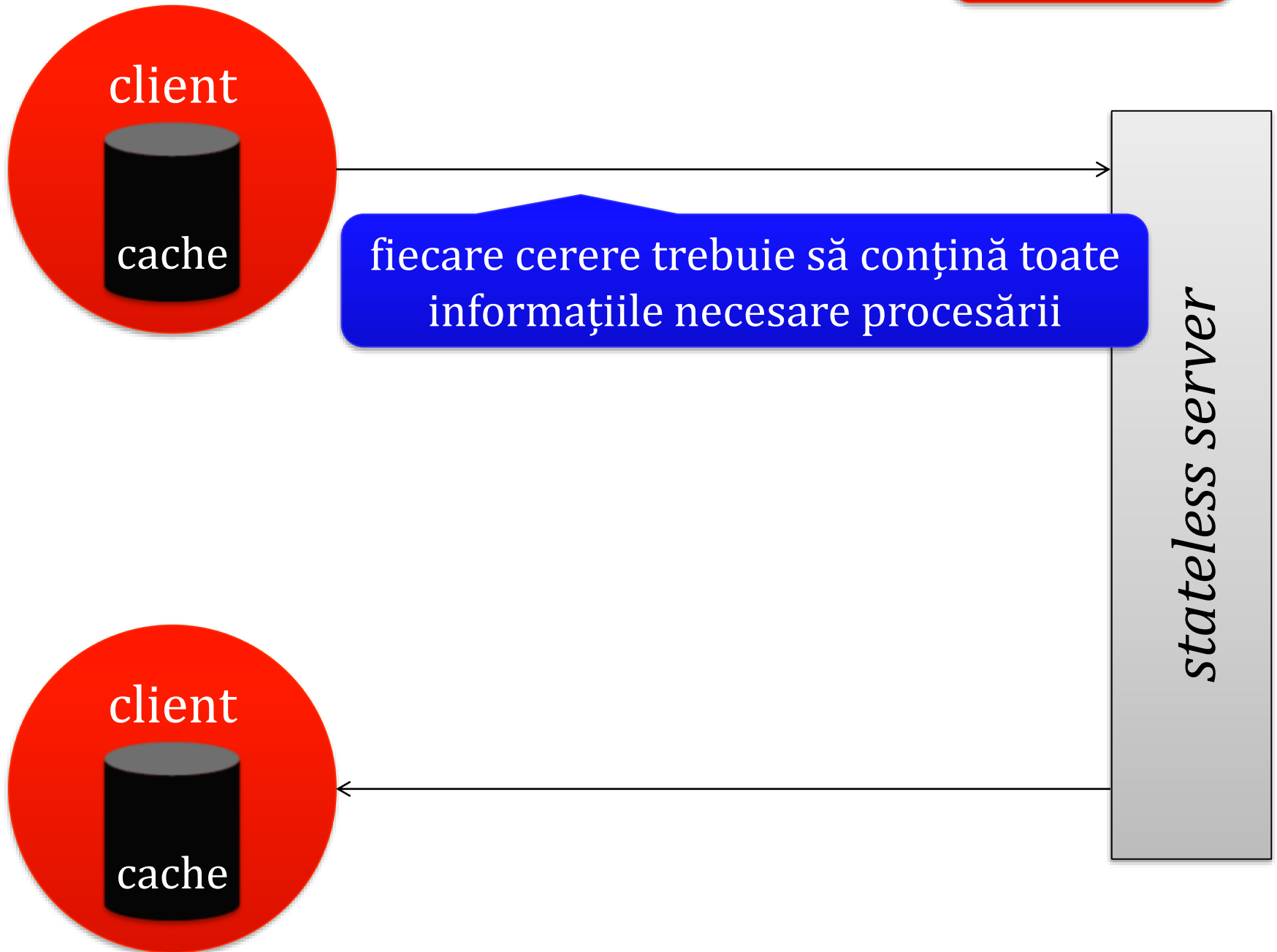
Reprezentările de resurse pot fi stocate temporar

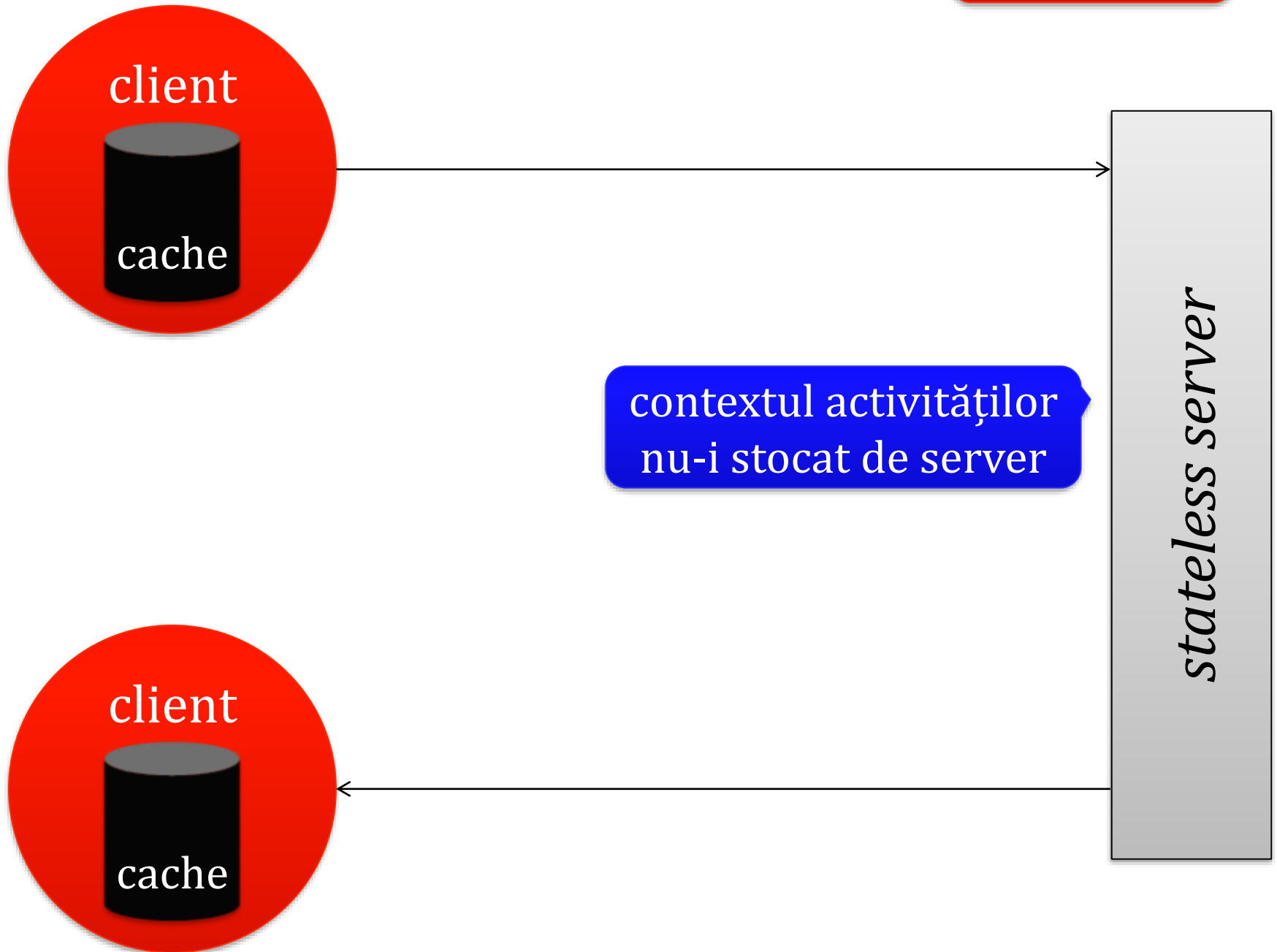
caching

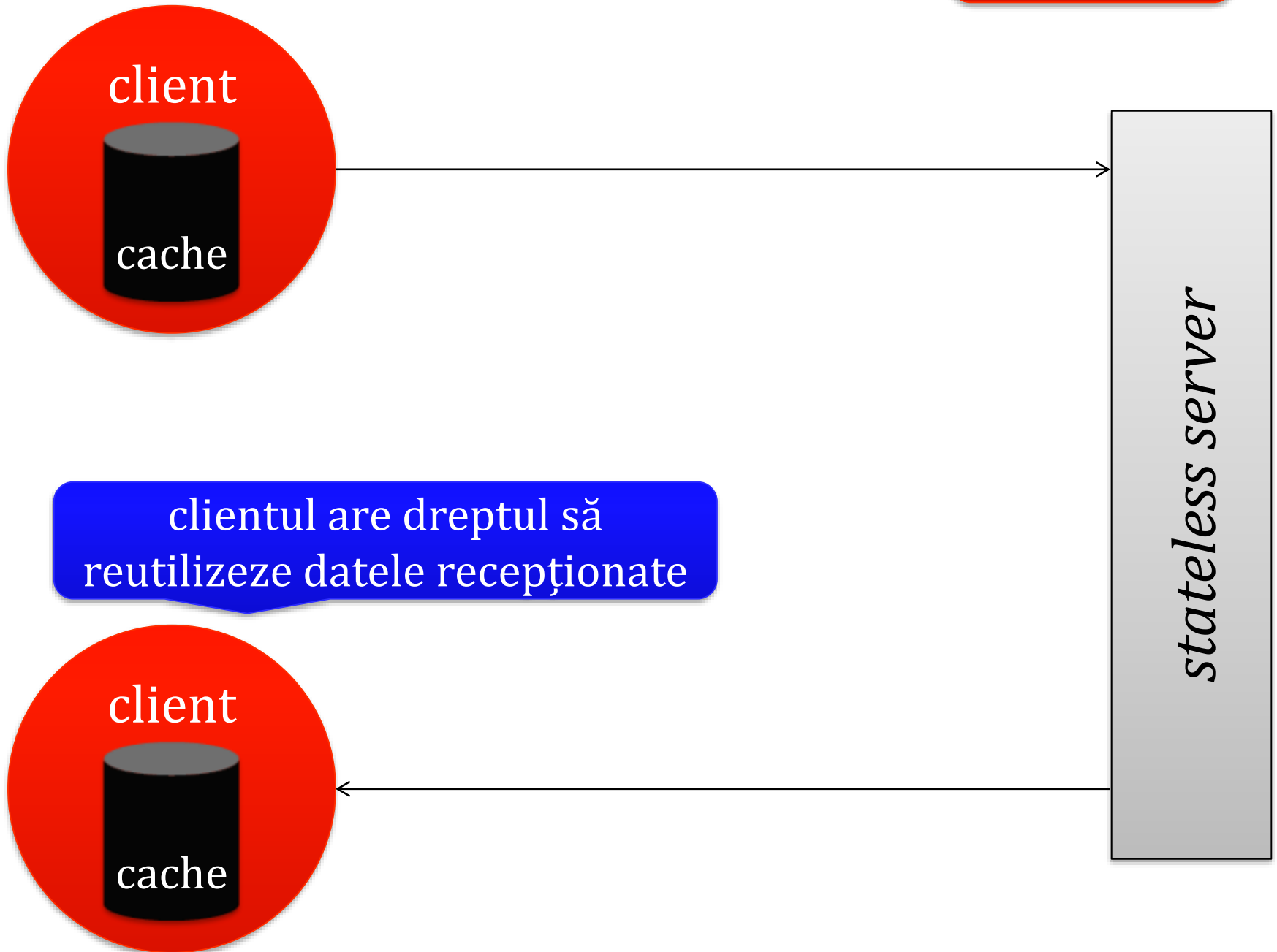
avansat



adaptare după B. Mulloy (2012)





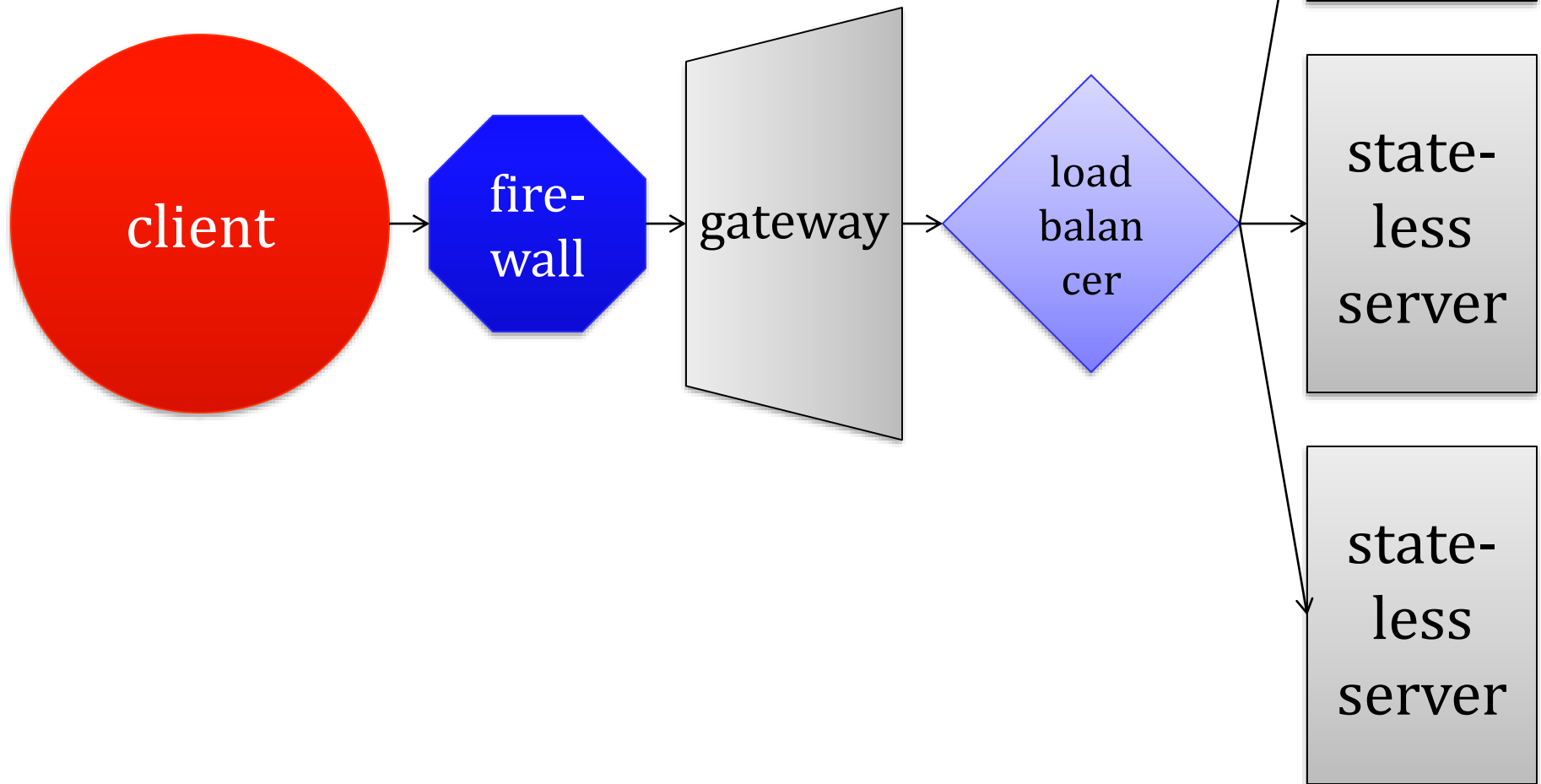


rest

Aplicația Web dezvoltată va fi stratificată

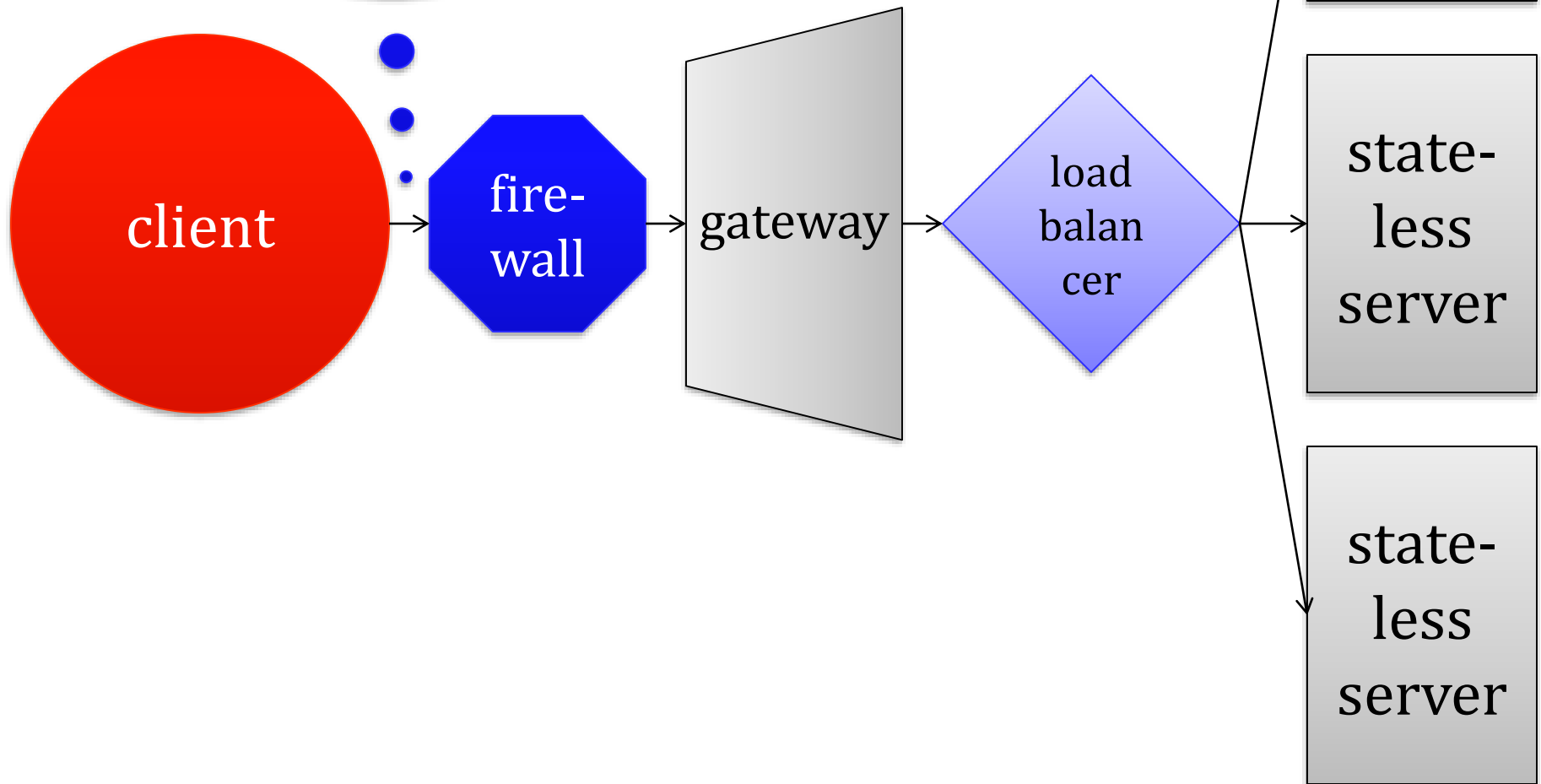
layered system

avansat

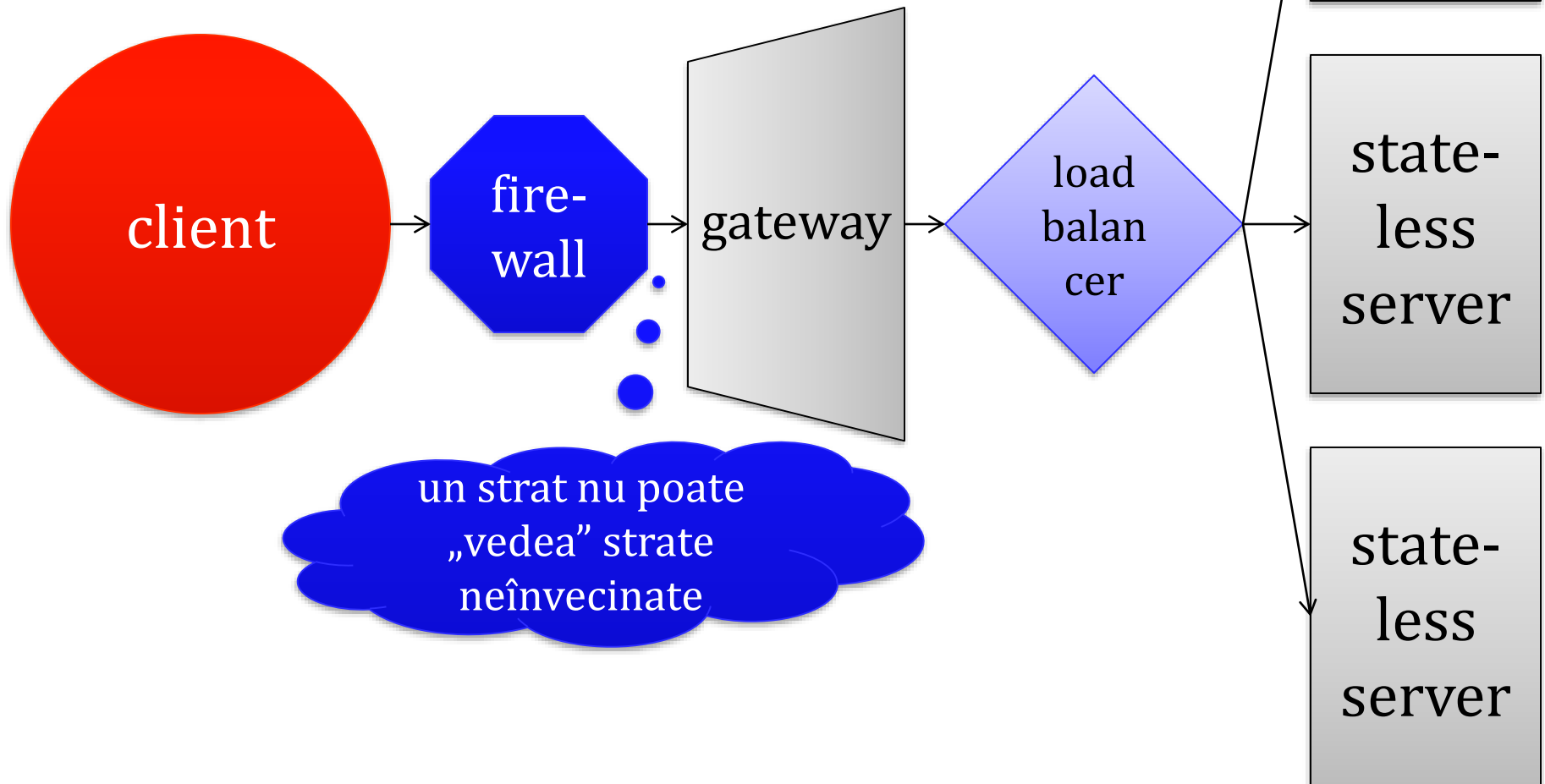


avansat

fiecare strat oferă
servicii stratelor
vecine

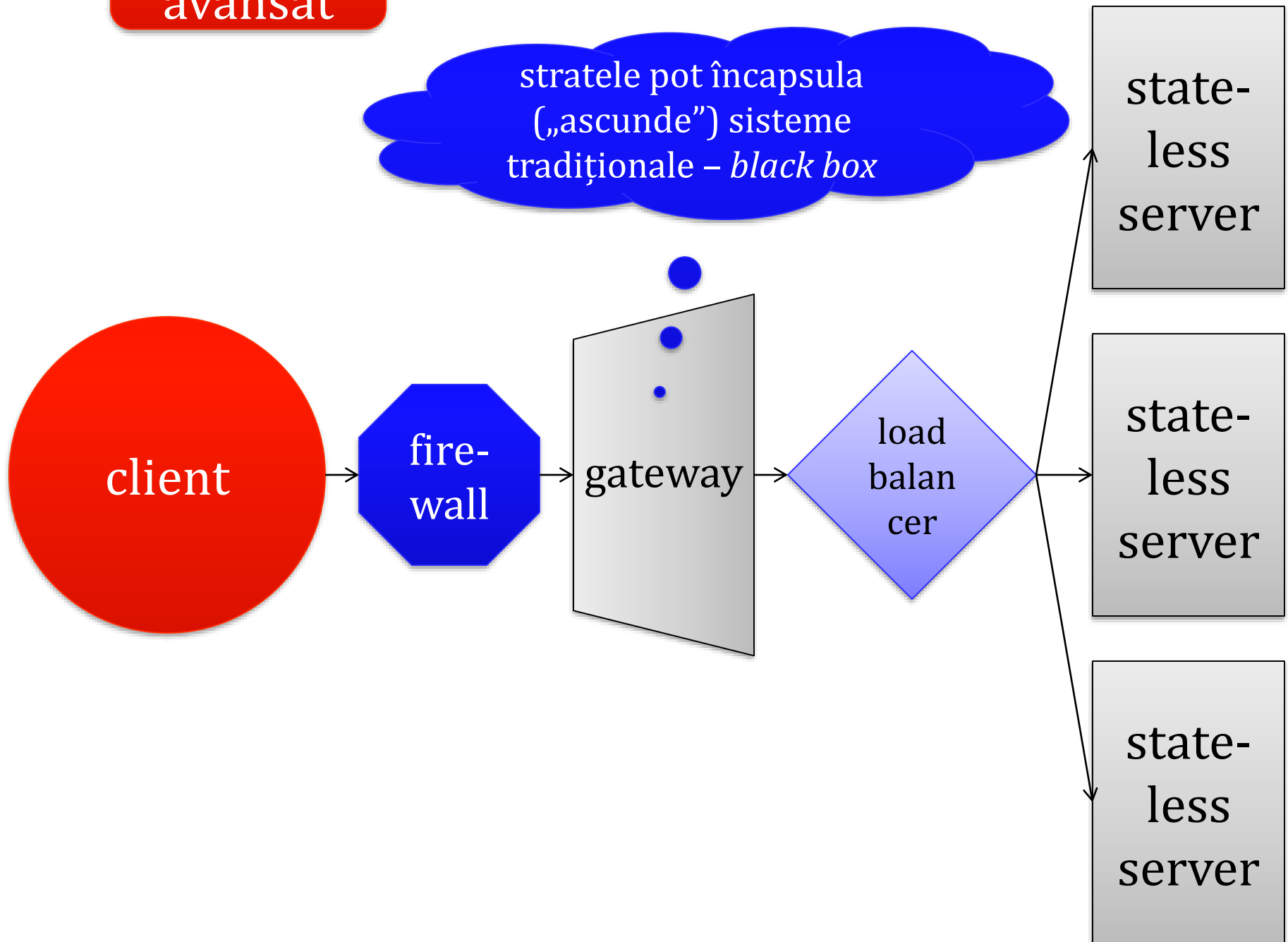


avansat

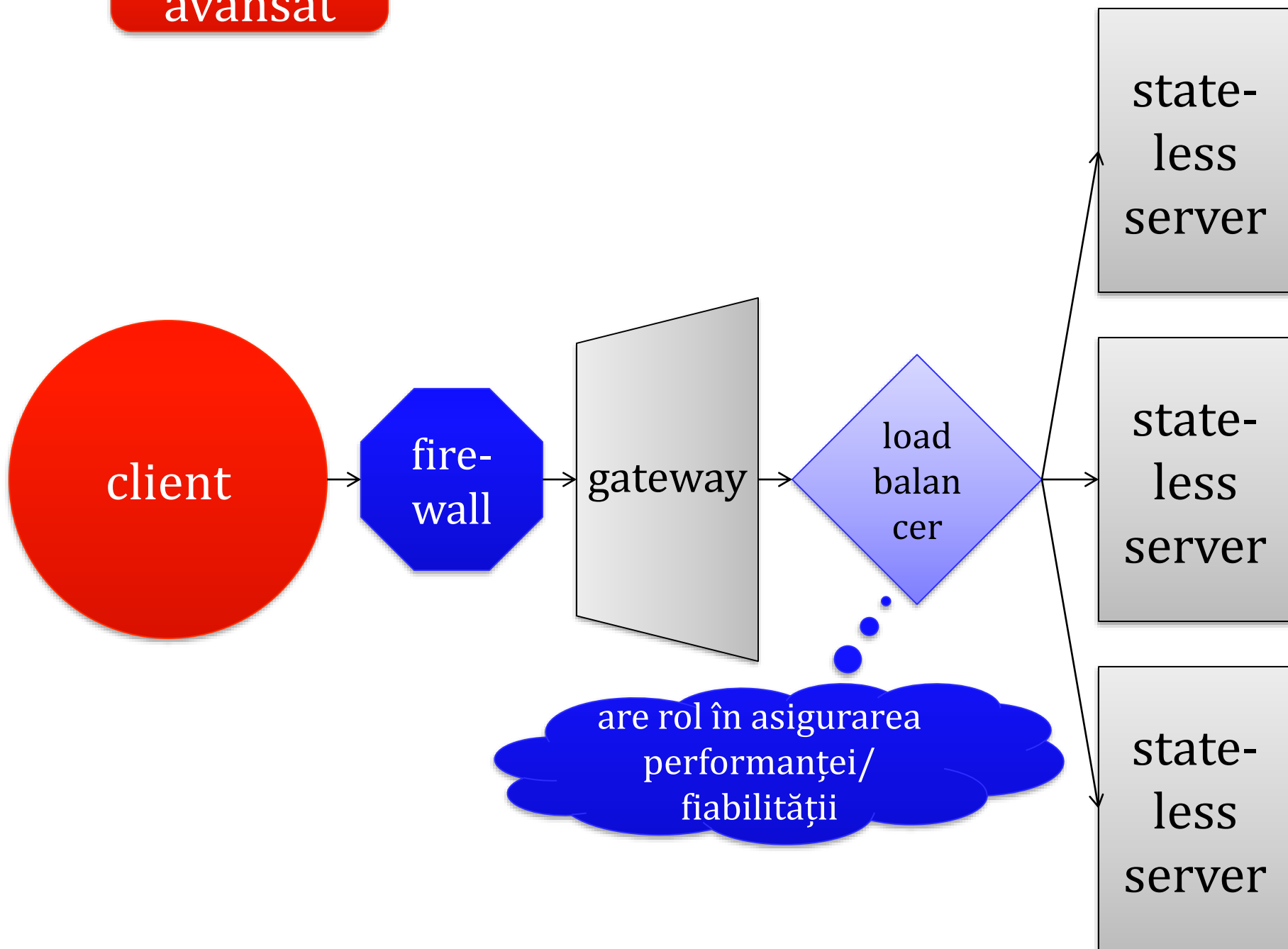


avansat

stratele pot încapsula
(„ascunde”) sisteme
tradiționale – *black box*



avansat



rest: exemplu

Implementarea unui magazin *on-line*
oferind dulciuri

rest: exemplu

Implementarea unui magazin *on-line*
oferind dulciuri

stilul „clasic” **SOAP** – conform RPC

operații privind produsele:

furnizeazăSortim(), adaugăSortim(), listeazăSortim(), cautăSortim()

operații ce vizează managementul utilizatorilor:

furnizeazăUtiliz(), adaugăUtiliz(), ștergeUtiliz(), cautăUtiliz(),...

rest: exemplu

Implementarea unui magazin *on-line*
oferind dulciuri

stilul „nou” REST

tipuri de resurse (**Sortim** + **Utiliz**), identificate unic de URI
<http://www.dulciuri.biz/sortim/portocale>

rest: exemplu

Implementarea unui magazin *on-line*
oferind dulciuri

stilul „nou” REST

tipuri de resurse (**Sortim** + **Utiliz**), identificate unic de URI
<http://www.dulciuri.biz/sortim/portocale/albastre>

URI intuitiv
“user/SEO friendly”

rest: exemplu

Serviciu pentru managementul adreselor Web favorite (*bookmark-uri*), cu posibilitatea atașării de termeni de conținut (*tag-uri*) și comentarii

social bookmarking

abordări similare: Delicious, Digg, Pocket, Reddit etc.

rest: exemplu

Serviciu pentru managementul adreselor Web favorite (*bookmark*-uri), cu posibilitatea atașării de termeni de conținut (*tag*-uri) și comentarii

funcționalitate de bază: listarea tuturor *bookmark*-urilor (eventual, filtrate după diverse criterii)

managementul *bookmark*-urilor:
adăugare, editare, ștergere, partajare

Resursa	URL	Metoda	Reprezentare
<i>Bookmark</i>	/bookmarks/{ <i>hash</i> }	GET	application/bookmark+xml
<i>Bookmark</i>	/bookmarks/{ <i>hash</i> }	PUT	application/bookmark+xml
<i>Bookmark</i>	/bookmarks/{ <i>hash</i> }	DELETE	
Lista de adrese	/bookmarks	GET	application/atom+xml
Lista de utilizatori	/users	GET	application/atom+xml
Lista de <i>tag</i> -uri	/tags	GET	application/atom+xml
Pagina principală	/	GET	application/xml

GET /bookmarks

200 OK

Content-type: **application/atom+xml**...

răspuns XML (Atom)
oferit de serviciu

<?xml version="1.0"?>

<feed xmlns="http://www.w3.org/2005/Atom">

<title>Bookmarks</title>

<entry>

<title>O resursă interesantă</title>

<link

href="/bookmarks/a211528f...bdcf"/>

<summary>

http://undeva.info/o-resursa-interesanta

</summary>

</entry>

<!-- eventual, alte elemente <entry>... -->

</feed>

digest – hash
(SHA-1, SHA-3,...)

obținerea
bookmark-urilor

GET /bookmarks/a211528f...bdcf

200 OK

Content-type: application/bookmark+xml

```
<bookmark>
  <title>O resursă interesantă</title>
  <url>http://undeva.info/o-resursa-interesanta</url>
  <user href="/users/tux">tux</user>
  <tags>
    <tag href="/tags/interesting">interesting</tag>
    <tag href="/tags/penguin">penguin</tag>
  </tags>
</bookmark>
```

preluarea unui *bookmark*:
răspunsul XML oferit de serviciul Web

crearea
unui *bookmark*

POST /bookmarks

Content-type: application/bookmark+xml

...

201 Created

Location: /bookmarks/a211528f...bdcf

PUT /bookmarks/a211528f...bdcf

Content-type: application/bookmark+xml

...

200 OK

înlocuirea
unui *bookmark*

rest

Resursele se denumesc folosind URI-uri (URL-uri)

Reprezentările sunt interconectate prin URL-uri

Pot exista intermediari (*e.g., proxy, cache, gateway*)
între clienți și resurse ► performanță, securitate,...

rest

Resursele se denumesc folosind URI-uri (URL-uri)

Reprezentările sunt interconectate prin URL-uri

Pot exista intermediari (*e.g., proxy, cache, gateway*)
între clienți și resurse ► performanță, securitate,...

Transferul de date poate fi și asincron – Ajax/Comet



într-un curs viitor

O resursă poate avea asociate reprezentări
ce pot fi accesate/alterate via operații HTTP

CRUD – *Create, Retrieve, Update, Delete*

<i>Operation</i>	SQL	HTTP
<i>Create</i>	INSERT	PUT POST
<i>Read (Retrieve)</i>	SELECT	GET
<i>Update (Modify)</i>	UPDATE	PUT POST PATCH
<i>Delete (Destroy)</i>	DELETE	DELETE

caz concret: *framework*-ul **LoopBack** – loopback.io/doc/ operații↔REST↔model de implementare↔SQL

Model create, retrieve, update, and delete operations

When you connect a model to a persistent data source such as a database, it becomes a *connected model* with a full set of create, read, update, and delete operations from the [PersistedModel](#) class:

Operation	REST	LoopBack model method (Node API)*	Corresponding SQL Operation
Create	PUT /modelName POST /modelName	create()	INSERT
Read (Retrieve)	GET /modelName?filter=...	find()	SELECT
Update (Modify)	PUT /modelName	updateAll()	UPDATE
Delete (Destroy)	DELETE /modelName/ /modelID	destroyById()	DELETE

(*) Methods listed are just prominent examples; other methods may provide similar functionality; for example: `findById()`, `findOne()`, and `findOrCreate()`.

rest

Interacțiunea cu un serviciu Web dezvoltat
în stilul REST se poate face via un API
(*Application Programming Interface*)



vezi prelegerea
următoare

Putem adopta o metodologie vizând dezvoltarea de servicii Web (API-uri) aliniate paradigmei REST?

rest: metodologie

Divizarea în resurse a setului de date
ale problemei

clase tipice de resurse:

Utilizatori

Documente – alternative: **Fotografii, Produse, Software,...**

Metadata – *e.g.*, **Comentarii, Formate, Locații, Platforme** etc.

rest: metodologie

Divizarea datelor problemei în categorii

cazuri concrete:

SoundCloud – developers.soundcloud.com/docs/api/

Tracks Users Me Playlists Groups Comments

StackExchange – api.stackexchange.com

Answers Badges Comments Questions Revisions Tags Users

World of Warcraft – dev.battle.net/io-docs

Characters Guilds Realms Auctions Items

rest: metodologie

„Numirea” prin URI a fiecărei resurse

exemplificări:

<http://web.info/Utilizatori/tux>

<http://web.info/Documente/pinguini-cu-mere-albastre>

rest: metodologie

„Numirea” prin URI a fiecărei resurse

cazuri concrete:

accesarea știrilor referitoare la un subiect de interes

<https://www.reddit.com/r/programming/>

acces la prezentările SlideShare ale utilizatorului **busaco**

<https://www.slideshare.net/busaco/presentations>

obținerea listei celor ce urmăresc un utilizator autentificat

<https://twitter.com/followers>

rest: metodologie

Organizarea resurselor

aceste resurse (*object instances*) pot fi organizate în colecții (*collections*) sau depozite (*stores*)

a se consulta D. Denicola, *Creating Truly RESTful APIs* (2013)
www.slideshare.net/domenicdenicola/creating-truly-res-tful-apis

rest: metodologie

Organizarea resurselor

colecție

catalog de resurse gestionate de **server**
clienții pot propune alterarea colecției
serverul decide care-i rezultatul unei operații

exemple (GitHub): [/orgs/openstack/repos](#), [/orgs/openstack/events](#)

rest: metodologie

Organizarea resurselor

depozit

„rezervă” de resurse gestionată de **client**
(inclusiv filtrare, sortare, paginare, accesare meta-date,...)

exemplu (GitHub): /users/openstack/repos?page=2&per_page=3

rest: metodologie

Proiectarea reprezentării(lor) *acceptate*
ce pot fi trimise de aplicația client
și reprezentării(lor) *întoarse* spre client

rest: metodologie

Proiectarea reprezentării(lor) *acceptate*
ce pot fi trimise de aplicația client
și reprezentării(lor) *întoarse* spre client

de considerat formatele standard, comune:

CSV – *Comma Separated Values*

JSON(-LD) – *JavaScript Object Notation (-Linked Data)*

XML – *Extensible Markup Language*

YAML – *Yet Another Markup Language*

rest: metodologie

Integrarea resurselor
via legături hipermedia + formulare

exemplificare (GitHub):

*“All resources may have one or more *_url properties linking to other resources. These are meant to provide explicit URLs so that proper API clients don’t need to construct URLs on their own.”*

developer.github.com/v3/#hypermedia

rest: metodologie

Crearea de studii de caz

specificarea condițiilor de eroare și/sau de excepție, inclusiv aspecte privind controlul versiunilor API-ului



Users

[Show/Hide](#) | [List Operations](#) | [Expand Operations](#) | [Raw](#)

POST	/Users/login	Login a user with username/email and password
POST	/Users/logout	Logout a user with access token
GET	/Users/confirm	Confirm a user registration with email verification token
POST	/Users/reset	Reset password for a user with email
GET	/Users/{id}/accessTokens/{fk}	Find a related item by id for accessTokens
DELETE	/Users/{id}/accessTokens/{fk}	
PUT	/Users/{id}/accessTokens/{fk}	
GET	/Users/{id}/accessTokens	
POST	/Users/{id}/accessTokens	Creates a new instance in accessTokens of this model.
DELETE	/Users/{id}/accessTokens	Deletes all accessTokens of this model.
GET	/Users/{id}/accessTokens/count	Counts accessTokens of User.
POST	/Users	Create a new instance of the model and persist it into the data source
PUT	/Users	Update an existing model instance or insert a new one into the data source
GET	/Users	Find all instances of the model matched by filter from the data source
GET	/Users/{id}/exists	Check whether a model instance exists in the data source
HEAD	/Users/{id}	Check whether a model instance exists in the data source
GET	/Users/{id}	Find a model instance by id from the data source
DELETE	/Users/{id}	Delete a model instance by id from the data source
PUT	/Users/{id}	Update attributes for a model instance and persist it into the data source

StrongLoop API
operații cu resurse specifice – aici **Users**

POST /Users

Create a new instance of the model

Source

PUT /Users

Update an existing model instance or insert a new one into the data source

Response Class

Model Model Schema

```
{
  "realm": "",
  "username": "",
  "credentials": "object",
  "challenges": "object",
  "email": "",
  "emailVerified": false,
  "verificationToken": "",
  "status": "",
  "created": "",
  "lastUpdated": ""
}
```

StrongLoop API

testarea interactivă a API-ului

strongloop.com/strongblog/node-js-rest-api-openshift-redhat/

Response Content Type

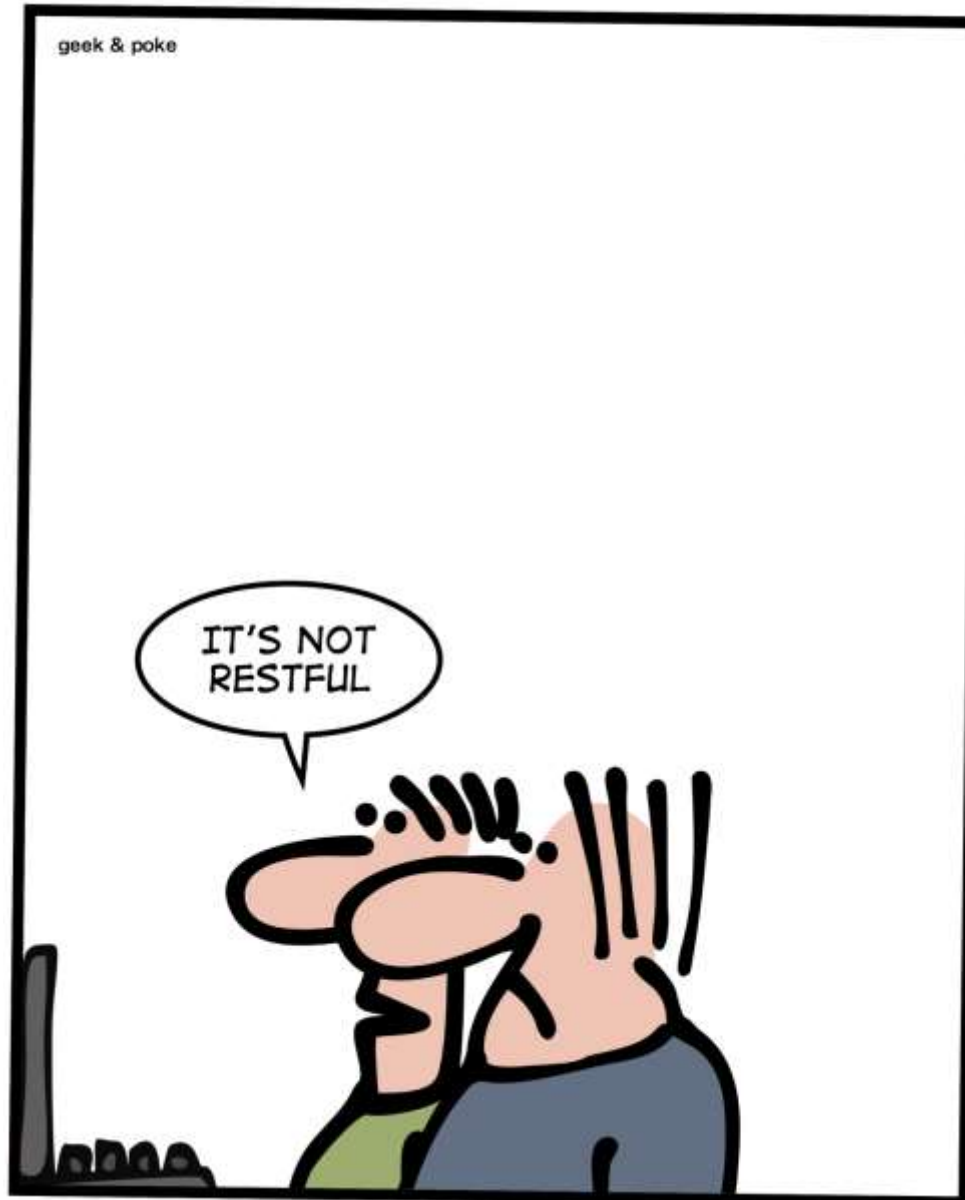
Parameters

Parameter	Value	Description	Parameter Type	Data Type
data	<div><div></div><div>Parameter content type: <input type="text" value="application/json"/> <input type="text" value="application/json"/> <input type="text" value="application/x-www-form-urlencoded"/></div></div>	Model instance data	body	Model Model Schema User { realm (string, optional), username (string, optional), credentials (object, optional), challenges (object, optional), email (string), emailVerified (boolean, optional), verificationToken (string, optional), status (string, optional), created (string, optional), lastUpdated (string, optional), id (number, optional) }

Try it out!

(în loc de) pauză

HOW TO INSULT A DEVELOPER



Aspecte practice de interes pentru dezvoltatori?

rest: **privire pragmatică**

URL-urile desemnând resurse (concepte)
de interes trebuie să fie simple și intuitive

utilizarea substantivelor pentru fiecare „lucru” (entitate)

rest: **privire pragmatică**

URL-urile desemnând resurse (concepte)
de interes trebuie să fie simple și intuitive

colecții de resurse – uzual, la plural
/students

rest: privire pragmatică

URL-urile desemnând resurse (concepte) de interes trebuie să fie simple și intuitive

colecții de resurse – uzual, la plural
/students

identificatori unici pentru membrii unei colecții
/students/tuxy (concret) vs. **/students/69** (abstract)

rest: **privire pragmatică**

URL-urile desemnând resurse (concepte) de interes trebuie să fie simple și intuitive

structura ierarhică a URL-urilor reprezintă ierarhia resurselor din cadrul domeniului modelat al aplicației

exemplu (GitHub):

[/repos/Microsoft/PTVS/commits/e95e15...7a3bf91baff88](#)

rest: privire pragmatică

Folosirea verbelor (metodelor) HTTP pentru efectuarea de operații asupra unor (colecții de) resurse

resursa (URI)	POST (creează)	GET (accesează)	PUT (actualizează)	DELETE (șterge)
/students	creează un student nou	listează studenții existenți	actualizează un set de studenți	șterge toți studenții
/students/69 (un URL deja existent)	eroare ☹️	oferă date despre student	dacă există, actualizează, altfel eroare	șterge studentul respectiv

rest: privire pragmatică

Raportarea erorilor

folosirea codurilor de stare HTTP – httpstatuses.com

exemple tipice:

200 OK, 204 No Content, 206 Partial Content

303 See Other, 304 Not Modified

400 Bad Request, 401 Unauthorized, 403 Forbidden,

404 Not Found, 405 Method Not Allowed

500 Internal Server Error, 503 Service Unavailable

rest: **privire pragmatică**

Raportarea erorilor

mesajele oferite trebuie să includă informații utile

rest: privire pragmatică

Raportarea erorilor

mesajele oferite trebuie să includă informații utile

GitHub – cod de stare HTTP oferit: 404

```
{ "message": "Not Found",  
  "documentation_url": "https://developer.github.com/v3" }
```

versus

New York Times (Semantic API) – cod de stare HTTP: 401

```
{ "message": "No API key found in headers or querystring" }
```

rest: privire pragmatică

Controlul versiunilor API-ului dezvoltat

“Never release an API without a version and make the version mandatory.” (Mulloy, 2012)

considerații de interes în articolul J. Curry,
Introduction to API Versioning Best Practices (2017):
nordicapis.com/introduction-to-api-versioning-best-practices/

rest: **privire pragmatică**

Controlul versiunilor API-ului dezvoltat

specificarea versiunii

soluții uzuale:

în antetul cererii HTTP

în cadrul URL-ului

rest: privire pragmatică

Controlul versiunilor API-ului dezvoltat

specificarea versiunii

în antetul cererii HTTP

Accept: `application/vnd.heroku+json; version=3`

în unele cazuri, folosind un antet propriu: `X-API-Version`

rest: **privire pragmatică**

Controlul versiunilor API-ului dezvoltat

specificarea versiunii

în cadrul URL-ului – eventual, ca parametru

roads.googleapis.com/v1/nearestRoads

ec2.amazonaws.com/?Action=RunInstances&Version=2016-11-15

rest: privire pragmatică

Controlul versiunilor API-ului dezvoltat

specificarea versiunii

continuous versioning

acces via același URI, indiferent de versiunea curentă

- ▶ practica preferată (“*Cool URIs don’t change*”)

nordicapis.com/continuous-versioning-strategy-for-internal-apis/

rest: privire pragmatică

Paginarea și oferirea de răspunsuri parțiale de obicei, se folosesc parametri precum **limit** și **offset**
/students?limit=33&offset=54

filtrele opționale pot fi delimitate de virgulă
/students?fields=name,age,year,email

GET Reviews by Keyword `movies/v2/reviews/search.format`

GET Reviews and NYT Critics' Picks `movies/v2/reviews/resource-type.format`

Parameter	Value	Type	Description
format	<input type="text" value="json"/>	extension	Select the response format.
resource-type	<input type="text" value="picks"/>	string	Set to retrieve reviews of all movies, reviews of NYT Critics' Picks currently in theaters or NYT Critics' Picks on DVD.
offset	<input type="text"/>	integer	The first 20 results are shown by default. To page through the results, set offset to the appropriate value.
order	<input type="text" value="by-title"/>	string	Set to specify the sort order of the results. See full documentation.
api-key	<div> <input type="text" value="by-title"/> <ul style="list-style-type: none"> by-title by-publication-date by-opening-date by-dvd-release-date </div>		

[Try it!](#) [Clear Results](#)

Request URI

`http://api.nytimes.com/svc/movies/v2/reviews/picks.json?order=by-title&api-key=sample-key`

interogări interactive asupra API-ului
oferit de *The New York Times*
developer.nytimes.com

```
X-Mashery-Responder: prod-j-worker-atl-04.mashery.com
Server: nginx/1.4.1
Date: Tue, 13 Oct 2015 18:16:35 GMT
Content-Type: application/json; charset=UTF-8
Content-Length: 33806
Last-Modified: Thu, 26 Feb 2015 23:58:51 GMT
Etag: "Thu, 26 Feb 2015 23:58:51 GMT"
Cache-Control: max-age=7100
Access-Control-Allow-Origin: *
Access-Control-Allow-Methods: GET, PUT, POST
Accept-Ranges: bytes
X-Varnish: 2002035251
Age: 0
Via: 1.1 varnish
X-Cache: MISS
```

diverse (meta-)date
oferite de serverul Web

```
"nyt_movie_id": 475603,
"display_title": "'71",
"sort_name": "'71",
"mpaa_rating": "R",
"critics_pick": 1,
"thousand_best": "0",
"byline": "Manohla Dargis",
"headline": "Review: In \u2018\u201971, \u2019 Young, Green and Behind",
"capsule_review": "",
"summary_short": "In Belfast, 1971 was an eventful year of demands and
clear who was the enemy.",
"publication_date": "2015-02-27",
"opening_date": "2015-02-27",
"dvd_release_date": "2015-07-07",
"date_updated": "2015-02-26 18:58:51",
"seo_name": "-71",
"link": {
  "type": "article",
```

răspuns în
format JSON

rest: **privire pragmatică**

Eterogenitatea formatelor reprezentărilor întoarse

indicarea formatului în URL via un parametru opțional
?alt=json (Google Data)

specificarea formatului acceptat în antetul cererii HTTP
Accept: application/json (Digg)

precizarea formatului în numele resursei solicitate
/venue.json (Foursquare)

rest: privire pragmatică

Utilizarea subdomeniilor pentru API-uri diferite
ale aceluiași ofertant de servicii

exemplificare:

search.twitter.com

stream.twitter.com

api.twitter.com

Cum pot fi accesate
reprezentări de resurse Web prin REST?

rest: implementare

Biblioteci/API-uri implementând HTTP

Apache HttpComponents (Java): hc.apache.org

Guzzle (PHP7): docs.guzzlephp.org/en/stable/

hackage (Haskell): hackage.haskell.org/package/HTTP

http (pachet Go): golang.org/pkg/net/http/

http.client (Python 3): docs.python.org/3/library/http.client.html

rest: implementare

Biblioteci/API-uri implementând HTTP

Httpful (bibliotecă PHP): phphttpclient.com

Hyper • Reqwest (biblioteci Rust):
hyper.rs • docs.rs/reqwest

libcurl (C; portări Perl, PHP, Ruby,...): curl.haxx.se/libcurl/

LibHTTP (bibliotecă C): www.libhttp.org

rest: implementare

Biblioteci/API-uri implementând HTTP

LWP (bibliotecă Perl): github.com/libwww-perl/libwww-perl

restify (*framework* Node.js): www.npmjs.com/package/restify

RestKit (macOS + iOS): github.com/RestKit/RestKit

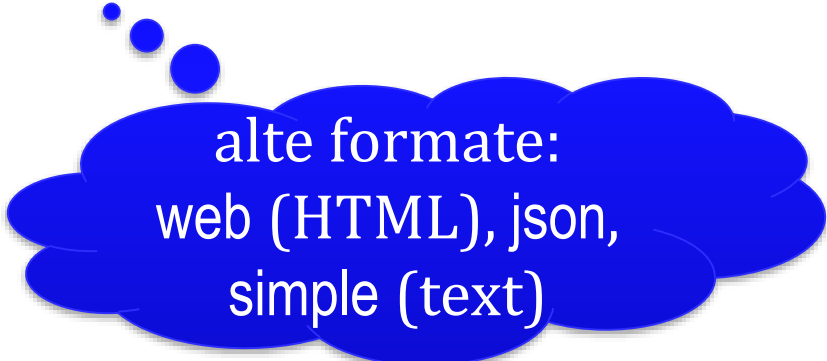
Requests for PHP: requests.ryanmccue.info/

RestSharp (pentru .NET): restsharp.org

rest: implementare – exemplul 1

Invocarea unui serviciu Web de prescurtare
a URL-urilor – <https://is.gd/>

un nou URL prescurtat va fi creat folosind adresa
<https://is.gd/create.php?format=xml&url=adresaWeb>



alte formate:
web (HTML), json,
simple (text)

Cererea HTTP ce invocă serviciul Web:

GET /create.php?format=xml&url=profs.info.uaic.ro/~busaco HTTP/1.1
Host: is.gd

Cererea HTTP ce invocă serviciul Web:

GET /create.php?format=xml&url=profs.info.uaic.ro/~busaco HTTP/1.1
Host: is.gd

Răspunsul obținut, transmis de serverul Web:

HTTP/1.1 200 OK

Server: nginx

Date: Mon, 06 May 2019 10:32:07 GMT+2

Content-Type: text/xml; charset=UTF-8

<?xml version="1.0" encoding="UTF-8" ?> ..

<output>

<shorturl>https://is.gd/K3oomj</shorturl>

</output>



reprezentare POX
(Plain Old XML)

apelarea serviciului Web
via **libcurl** (PHP)

```
$c = curl_init (); // inițializăm cURL
// stabilim URL-ul serviciului Web invocat
curl_setopt ($c, CURLOPT_URL,
    'https://is.gd/create.php?format=xml&url=profs.info.uaic.ro/~busaco');
// rezultatul cererii va fi disponibil ca șir de caractere
curl_setopt ($c, CURLOPT_RETURNTRANSFER, true);
// nu verificăm certificatul digital utilizat pentru transferul datelor cu HTTPS
curl_setopt ($c, CURLOPT_SSL_VERIFYPEER, false);
// preluăm reprezentarea oferită de server (aici, un document XML)
$res = curl_exec ($c);
curl_close ($c); // închidem conexiunea cURL

$doc = new DOMDocument (); // procesăm rezultatul via DOM
$doc->loadXML ($res);
// preluăm conținutul elementului <shorturl>
$urls = $doc->getElementsByTagName ('shorturl');
foreach ($urls as $url) {
    echo '<p>New short URL is ' . $url->nodeValue . '</p>';
}
```

rest: implementare – exemplul 1

Invoking Web service from `https://is.gd`
`/create.php?format=xml&url=profs.info.uaic.ro/~busaco`

Server response:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<output><shorturl>https://is.gd/K3oomj</shorturl></output>
```

New short URL is <https://is.gd/K3oomj>

rest: implementare – exemplul 2

Accesarea datelor publice vizând universul fictiv
„Războiul stelelor” (*Star Wars*)

colecții de resurse:

Planets, Spaceships, Vehicles, People, Films, Species

fiecare categorie de resurse are proprietăți specifice
e.g., orice instanță de Films include title, director, characters,...

Need a hint? try [people/1/](#) or [planets/3/](#) or [starships/9/](#)

Result:

```
{
  "name": "Alderaan",
  "rotation_period": "24",
  "orbital_period": "364",
  "diameter": "12500",
  "climate": "temperate",
  "gravity": "1 standard",
  "terrain": "grasslands, mountains",
  "surface_water": "40",
  "population": "2000000000",
  "residents": [
    "http://swapi.co/api/people/5/",
    "http://swapi.co/api/people/68/",
    "http://swapi.co/api/people/81/"
  ],
  "films": [
    "http://swapi.co/api/films/6/",
    "http://swapi.co/api/films/1/"
  ]
}
```

răspuns
disponibil
în format
JSON

```
{
  "name": "Leia Organa",
  "height": "150",
  "mass": "49",
  "hair_color": "brown",
  "skin_color": "light",
  "eye_color": "brown",
  "birth_year": "19BBY",
  "gender": "female",
  "homeworld": "http://swapi.co/api/planets/2/",
  "films": [ "http://swapi.co/api/films/6/",... ],
  "species": [ "http://swapi.co/api/species/1/" ],
  "vehicles": [
    "http://swapi.co/api/vehicles/30/" ],
  "starships": [],
  "created": "2014-12-10T15:20:09.791000Z",
  "edited": "2014-12-20T21:17:50.315000Z",
  "url": "http://swapi.co/api/people/5/"
}
```

detalii la swapi.co/documentation

rest: implementare – exemplul 3

Preluarea datelor despre artefacte culturale oferite
de *Metropolitan Museum of Art*

metmuseum.github.io

disponibile liber sub licența *Creative Commons Zero*

Objects – colecții de resurse

Object – include (meta-)date de interes

rest: implementare – exemplul 3

Preluarea datelor despre artefacte culturale oferite
de *Metropolitan Museum of Art*

metmuseum.github.io

API-ul REST oferit poate fi accesat fără autentificare
folosind domeniul collectionapi.metmuseum.org
rezultatele interogărilor sunt disponibile în format JSON

parte componentă a inițiativei Google Arts & Culture
artsandculture.google.com

rest: implementare – exemplul 3

Preluarea datelor despre artefacte culturale oferite
de *Metropolitan Museum of Art*

pasul #1:

căutarea – cerere GET – unor resurse de interes

collectionapi.metmuseum.org/public/collection/v1/search?q=Romania

rest: implementare – exemplul 3

Preluarea datelor despre artefacte culturale oferite
de *Metropolitan Museum of Art*

pasul #1:

căutarea – cerere **GET** – unor resurse de interes

se obține o colecție JSON de identificatori – objectIDs
{ "total": 47, "objectIDs": [98440, 32843,..., 730799] }

rest: implementare – exemplul 3

Preluarea datelor despre artefacte culturale oferite
de *Metropolitan Museum of Art*

pasul #2:

accesul – cu GET – la datele vizând o resursă specifică

collectionapi.metmuseum.org/public/collection/v1/objects/32843

objectID

rest: implementare – exemplul 3

Preluarea datelor despre artefacte culturale oferite
de *Metropolitan Museum of Art*

pasul #2:

accesul – cu GET – la datele vizând o resursă specifică

rezultatul reprezintă un obiect JSON încapsulând diverse
(meta-)date furnizate de proprietăți: imagini ale
artefactului (`primaryImage`, `additionalImages`), proveniența
(`country`, `state`, `region`, `city`), clasificarea (`classification`),...

câmpurile-antet ale cererii HTTP:

Accept: application/json,application/xml;q=0.9,*/*;q=0.8

Accept-Encoding: gzip, deflate, br

Accept-Language: en,en-GB;q=0.5

Connection: keep-alive

Host: collectionapi.metmuseum.org

User-Agent: Mozilla/5.0 ... Gecko/20100101 Firefox/66.0

câmpurile-antet ale răspunsului furnizat de API:

Access-Control-Allow-Origin: *

Content-Encoding: gzip

Content-Type: application/json; charset=UTF-8

Date: Sun, 05 May 2019 10:38:18 GMT

Transfer-Encoding: chunked

X-CDN: Incapsula

```
objectID: 102377
isHighlight: false
accessionNumber: "09.50.2472"
isPublicDomain: true
▶ primaryImage: "https://images.metmuseum.../original/09.50.2472.jpg"
▶ primaryImageSmall: "https://images.metmuseum...web-large/09.50.2472.jpg"
additionalImages: []
constituents: null
department: "Costume Institute"
objectName: "Apron"
title: "Apron"
culture: "Romanian"
```

răspunsul JSON
întors de serviciul Web apelat



rest: implementare

Biblioteci/API-uri implementând HTTP

permit dezvoltarea de aplicații *desktop*, *mobile* etc.

suport pentru crearea de aplicații hibride (*mash-up-uri*)
la nivel de server

nu funcționează în navigatorul Web

atenție la problemele de securitate ce pot apărea!

rest: implementare

Navigatoarele Web actuale

nu necesită un API distinct pentru acces via HTTP

disponibilitate pe orice platformă

suport pentru REST prin obiectul **XMLHttpRequest** (Ajax),
Fetch API (HTML5) ori **WebSocket API** (HTML5)

într-un viitor curs

rest: dezvoltare – exemplificări

ASP.NET MVC + Web API (C# *et al.*):

www.asp.net/web-api

Express, LoopBack, Sails, Superagent (Node.js)

www.npmjs.com/search?q=REST&ranking=popularity

JAX-RS (*Java API for RESTful Web Services*)

github.com/jax-rs

Restlet (Java)

restlet.com/open-source/

rest: dezvoltare – exemplificări

Cornice, Django, Eve, Pecan (Python)

Grape, RESTRack, Ruby on Rails (Ruby)

www.ruby-toolbox.com/categories/API_Builders

micro-framework-uri PHP populare:

Fat-Free – fatfreeframework.com

Lumen – lumen.laravel.com

Slim – www.slimframework.com

Siler – siler.leocavalcante.dev

a se consulta și github.com/marmelab/awesome-rest

rest: dezvoltare

Servicii publice ce pot fi consumate via REST – exemple:

AIDSInfo, Amazon, Basecamp, Blip.tv, DBpedia, eBay,
Ericsson, Facebook, GitHub, Google, ISBNdb, LinkedIn,
Mastercard, Open Movie Database, Pipl, Quora, Tumblr,
Wikidata,...

de explorat situl **ProgrammableWeb**
www.programmableweb.com/category/all/apis

+

lista API-urilor publice
github.com/abhishekbanthia/Public-APIs

Groupon Goods

Sort by

Relevance



Categories

Auto & Home Improvement (34972)

Baby, Kids & Toys (22628)

Electronics (31391)

Entertainment (6637)

For the Home (104773)

Grocery & Household (8011)

Health & Beauty (60546)

Jewelry & Watches (49666)

Men's Fashion (41195)

Pet Supplies (8576)

Sports & Outdoors (36464)

Personalized Items (857)

Women's Fashion (43508)

Discount Types

☒ All

☐ Clearance



Microfiber Luxury Home Ultra Soft Sheet Set (6-Piece)

This sheet set is made of an extra soft microfiber material which resists stains, wrinkles, and pilling, for crisp look and cozy feel

100,000+ bought

★★★★★ (13,050)

~~\$99.99~~ **\$12.99**

Sale Ends 5/12

[View Deal](#)

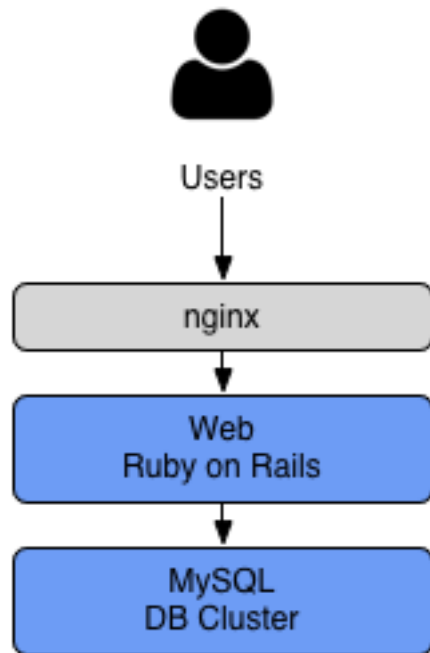


studiu de caz: **Groupon**

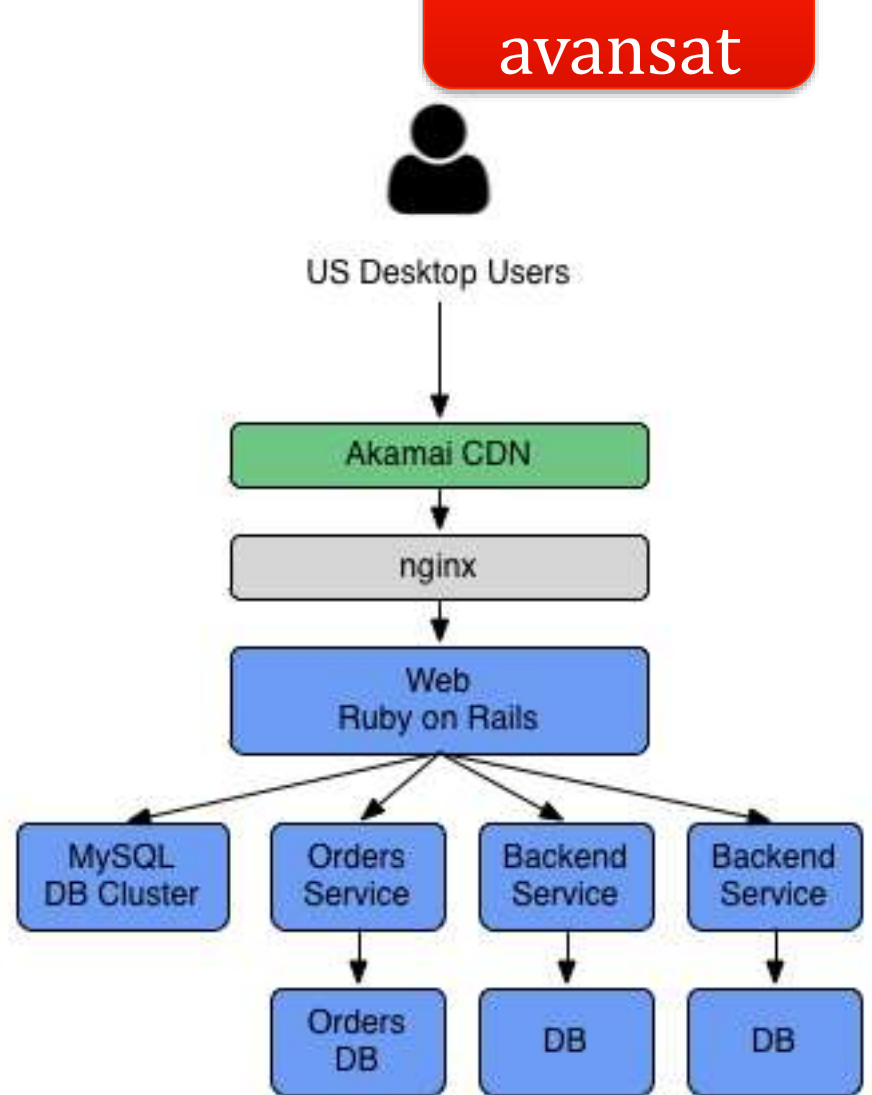
Scop: prezentarea de oferte de bunuri de consum

de la o arhitectură monolitică
la una adoptând servicii (API-uri) REST

engineering.groupon.com/2013/misc/i-tier-dismantling-the-monoliths/

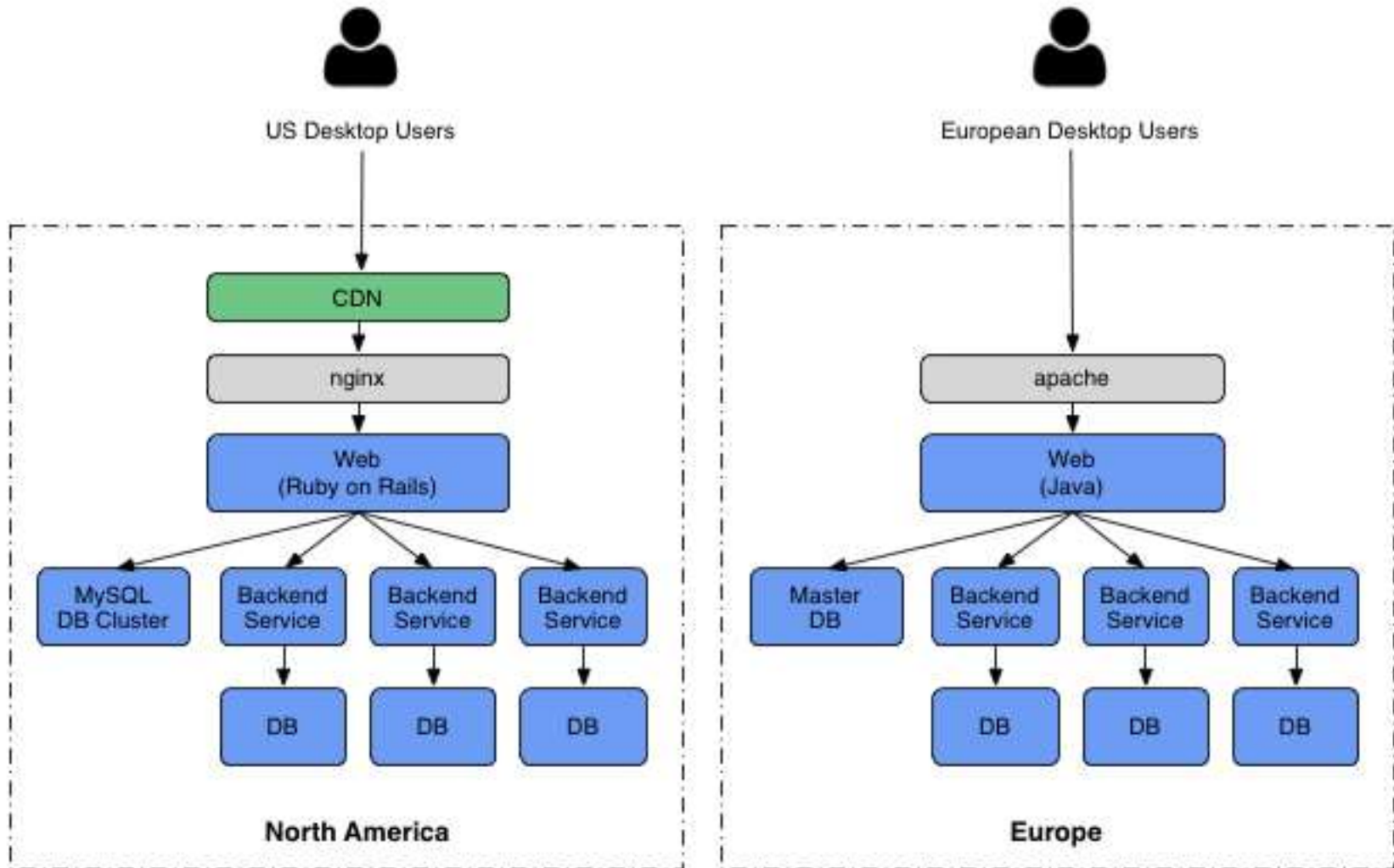


arhitectură inițială
MVC tradițional

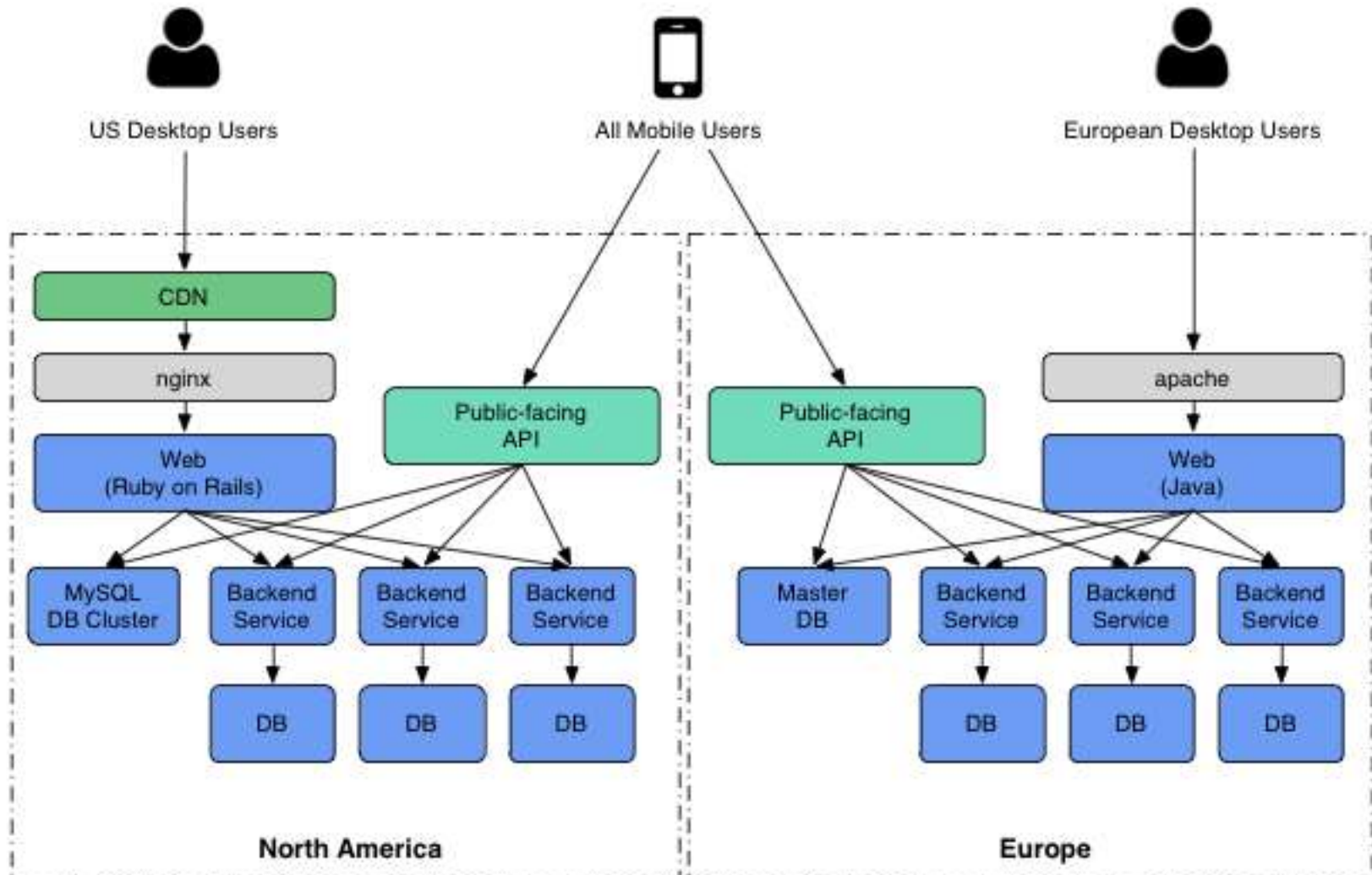


avansat

asigurarea performanței
scalabilitate cu CDN
(*Content Distribution Network*)
și servicii de acces la date



arhitectură eterogenă
implementări distincte în funcție de zona geografică

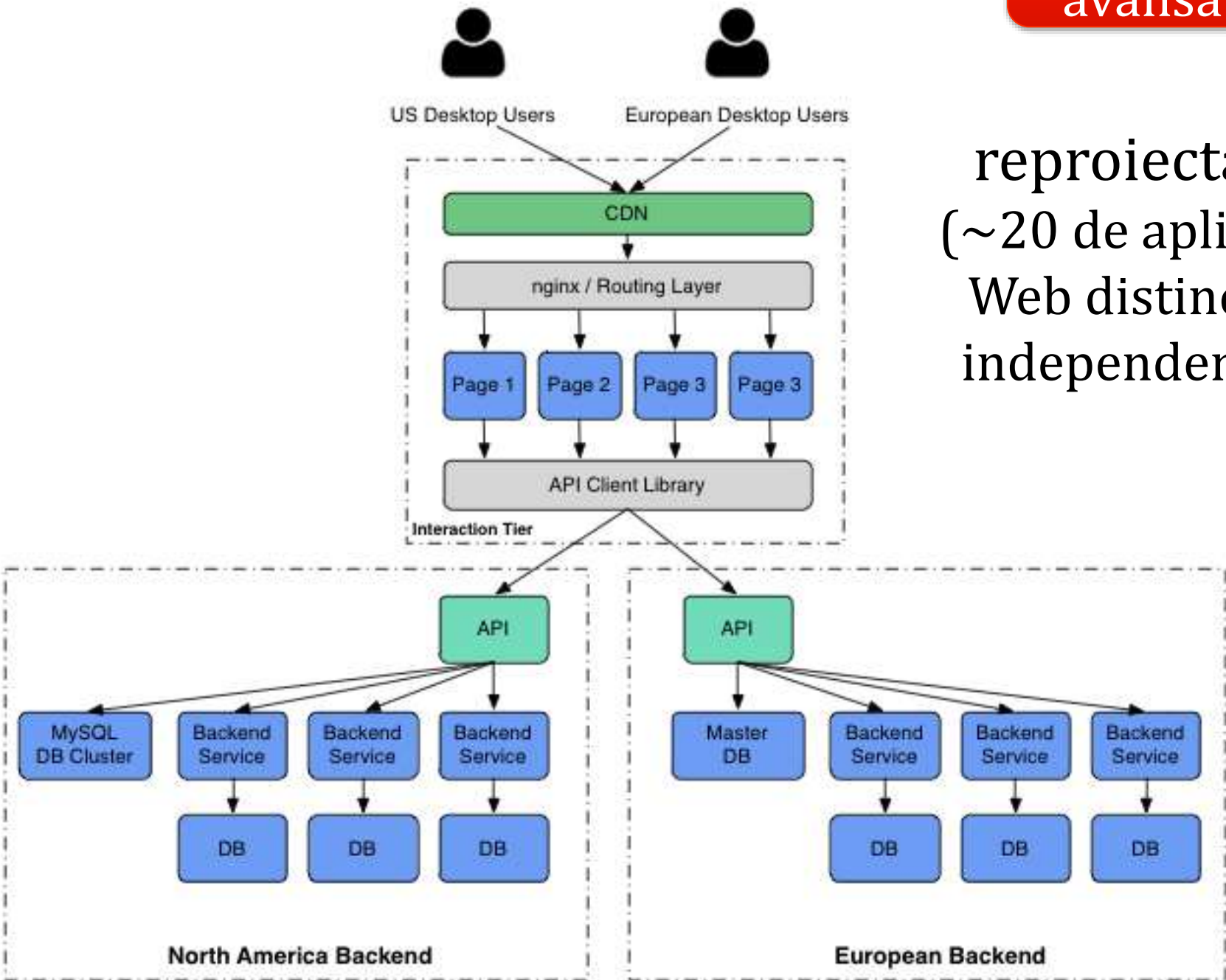


oferirea suportului

pentru interacțiuni cu dispozitive mobile

API dedicat, accesul depinzând de localizarea utilizatorului

reproiectare
(~20 de aplicații
Web distincte,
independente)

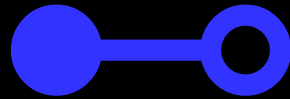


soap vs. rest

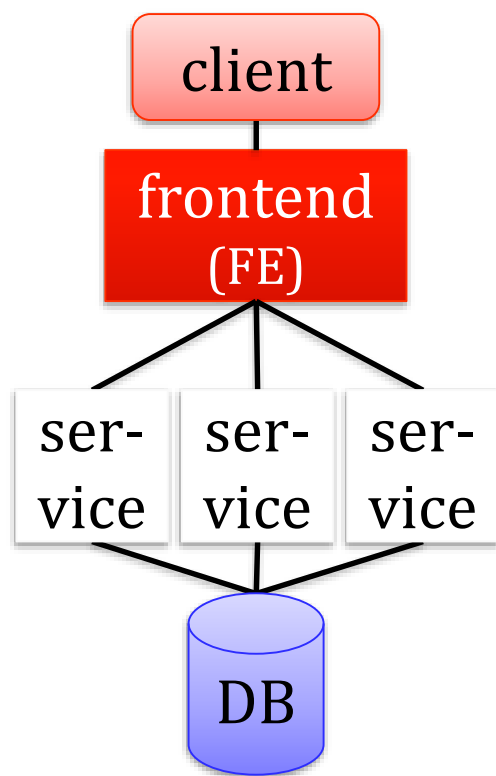
SOAP	REST
Acțiuni arbitrare (verbe)	Acțiuni fixe – HTTP: GET, POST,...
Structuri de date oricât de complexe – inclusiv validare	Operează asupra reprezentărilor de resurse – XML, JSON, HTML
Descriere complexă a serviciului (pe baza WSDL)	Scalabil (mai ușor de extins)
Suport pentru <i>XML messaging</i>	Bazat pe URI + hipermedia
Dezvoltare sofisticată: securitate, intermediari, specificații WS-*, interoperabilitate,...	Uzual, mai facil de programat (+disponibilitatea API-urilor)
Specific mediului <i>enterprise</i> (infrastructuri complexe)	Abordare pragmatică aplicații sociale <i>et al.</i> (Web 2.0)

rezumat

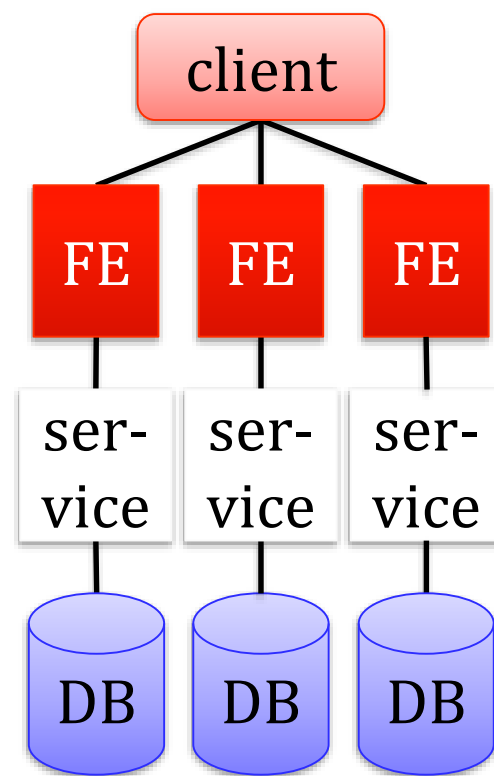
dezvoltare de servicii Web via REST



de la HTTP la metodologii, instrumente și exemple



arhitectură bazată
pe servicii Web



arhitectură recurgând
la microservicii

episodul viitor:

dezvoltarea de aplicații Web complexe:

**specificarea API-urilor, microservicii, autentificare,
autorizare, acces la date via GraphQL**