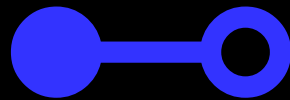


Tehnologii Web

servicii Web (II)



dezvoltarea de aplicații Web prin **REST**

“Concizia este sora talentului.”

Anton Cehov

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
cu focalizare asupra reprezentării datelor

teză de doctorat – Roy Fielding (2000)

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,
blog-ul unei persoane, fotografie, flux de știri, program,...

rest

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

reprezentare pe baza unui format de date

textual sau binar

exemple tipice: HTML, JSON, PNG, SVG, PDF, Atom etc.

rest

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

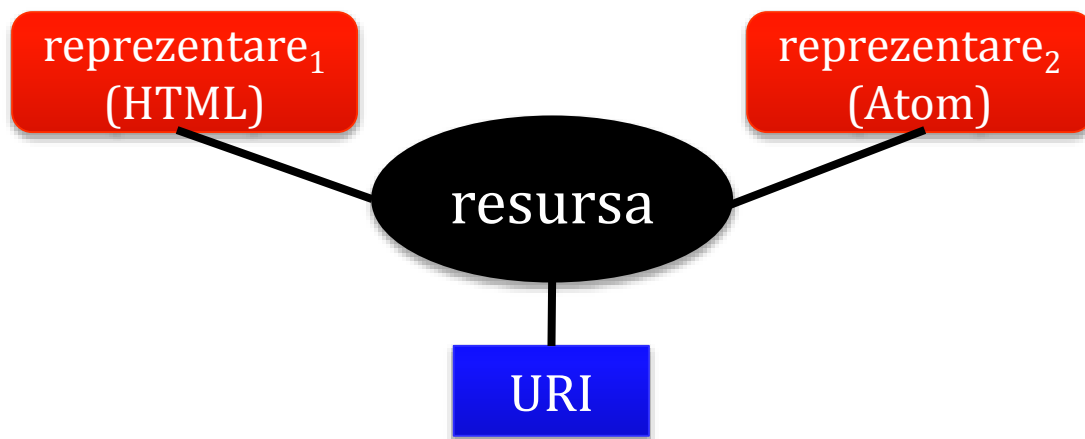
reprezentare pe baza unui format de date

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

rest

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

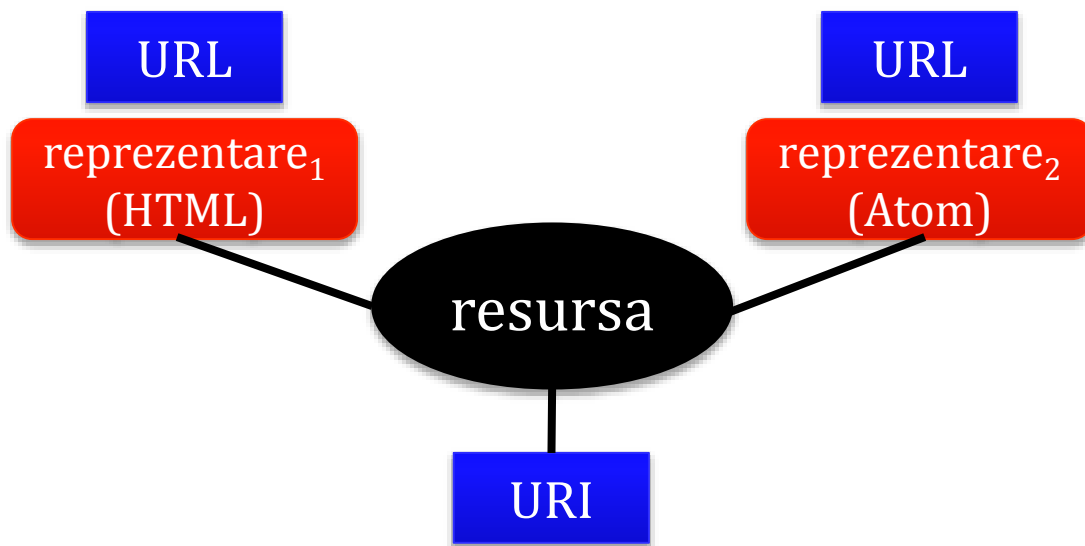
- reprezentările aceleiași resurse
- desemnate de un URI unic – pot fi multiple



rest

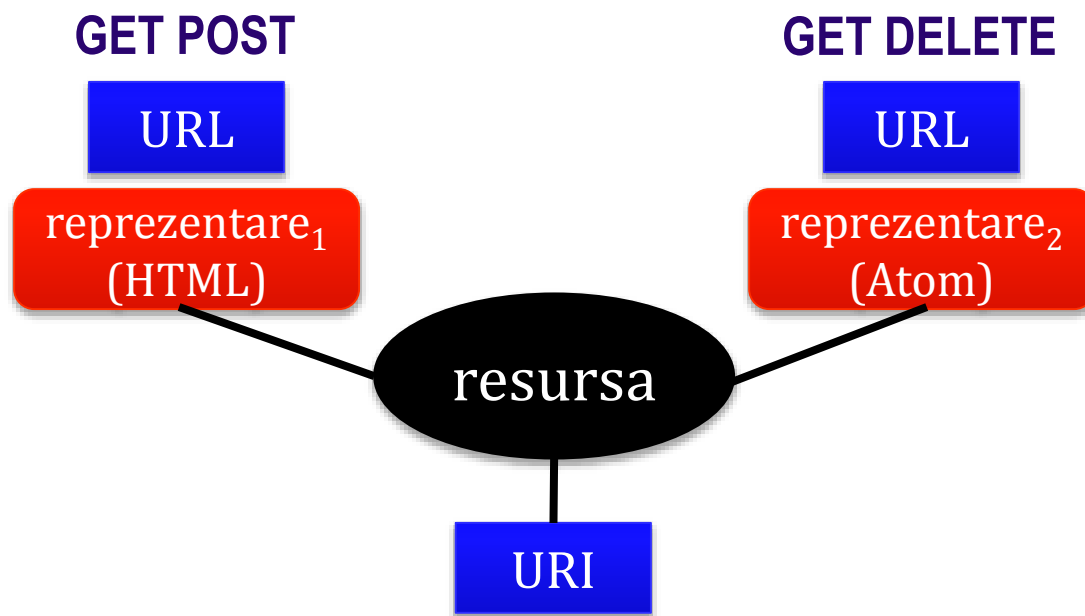
Rezultatul unei procesări conduce la obținerea unei **reprezentări** a unei resurse

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 protocolul HTTP

GET

scop: accesarea (citirea) unei reprezentări de resursă

nu conduce la modificarea stării serverului (*safe*)

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

rest

Verbele (acțiunile) sunt stipulate de protocolul HTTP

PUT

actualizează o reprezentare de resursă sau
eventual creează o resursă la nivel de server Web
uzual, returnează un identificator (URI) al resursei
nu e considerată *safe*, dar este idempotentă

rest

Verbele (acțiunile) sunt stipulate de protocolul HTTP

PATCH

actualizarea parțială a unei reprezentări de resursă

nu este *safe* și nici idempotentă

rest

Verbele (acțiunile) sunt stipulate de protocolul HTTP

POST

crează o resursă (uzual, subordonată altei resurse)

nu este nici *safe*, nici idempotentă

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

rest

Verbele (acțiunile) sunt stipulate de protocolul HTTP

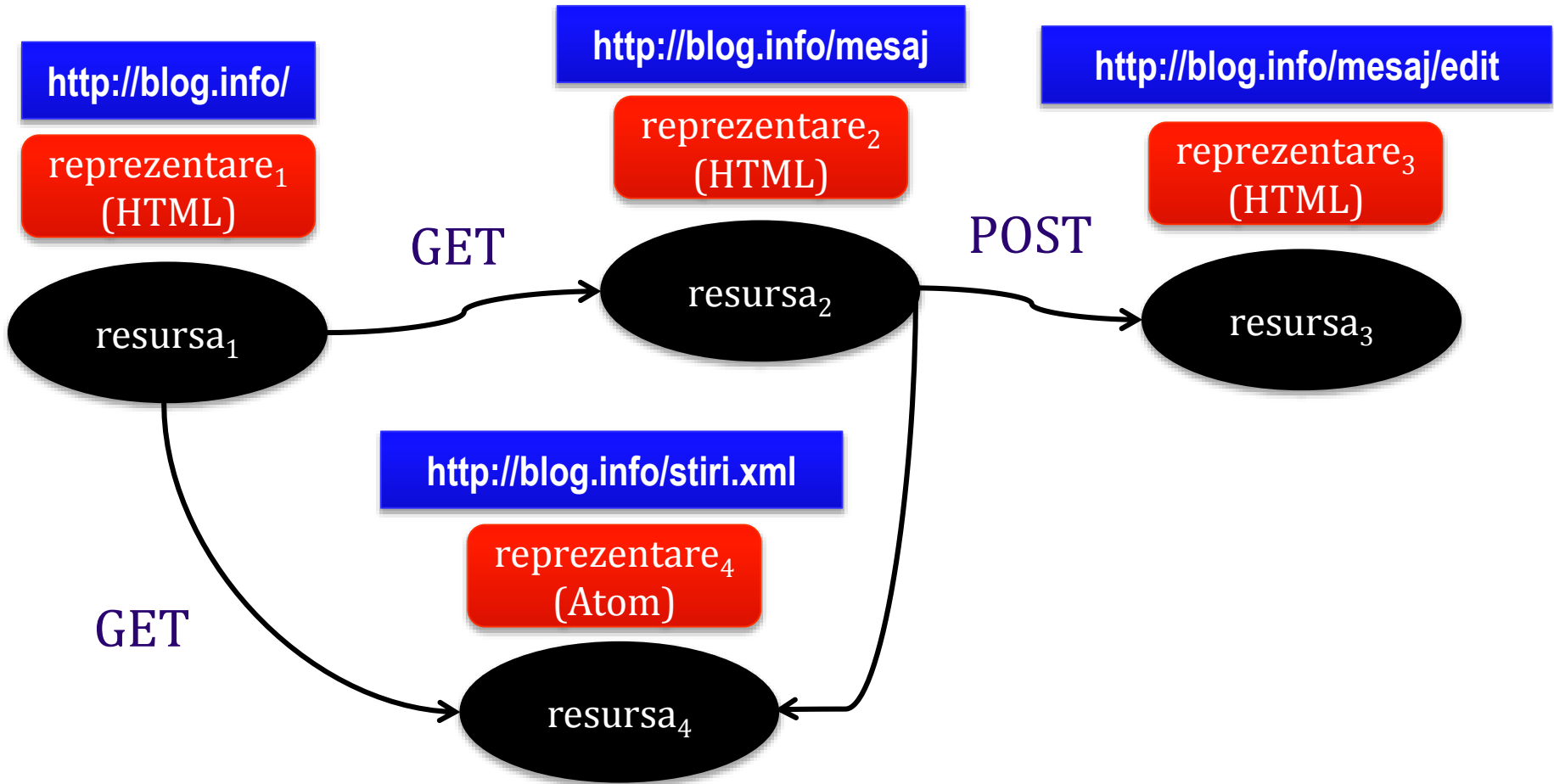
DELETE

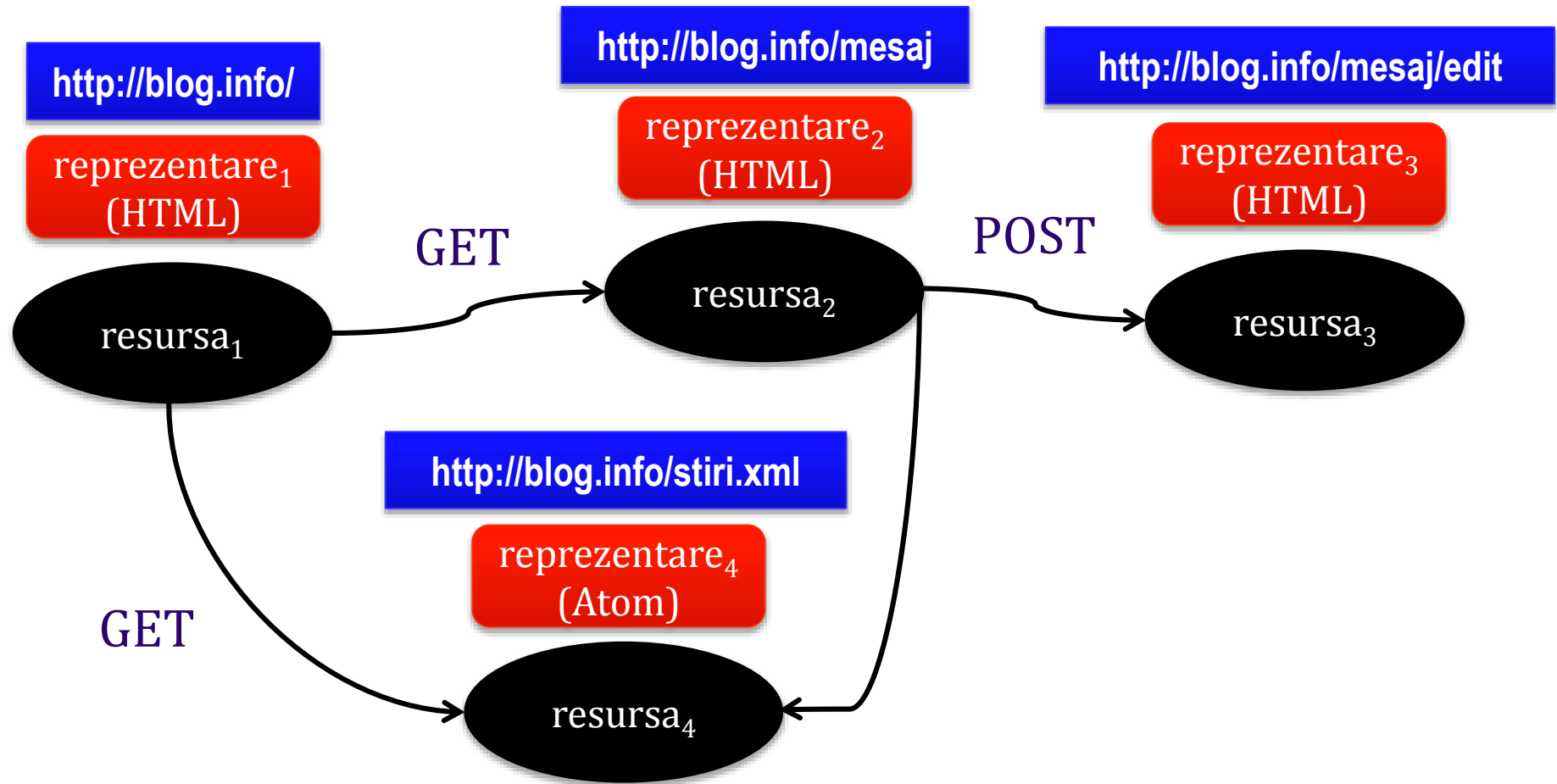
șterge (elimină) o resursă desemnată de un URI

este idempotentă

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*)

rest

Transferul se realizează prin protocolul **HTTP**

Reprezentarea este modelată conform unui format

– *e.g.*, **XML** sau **JSON** – și indicată prin tipuri **MIME**

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 considerație contextul

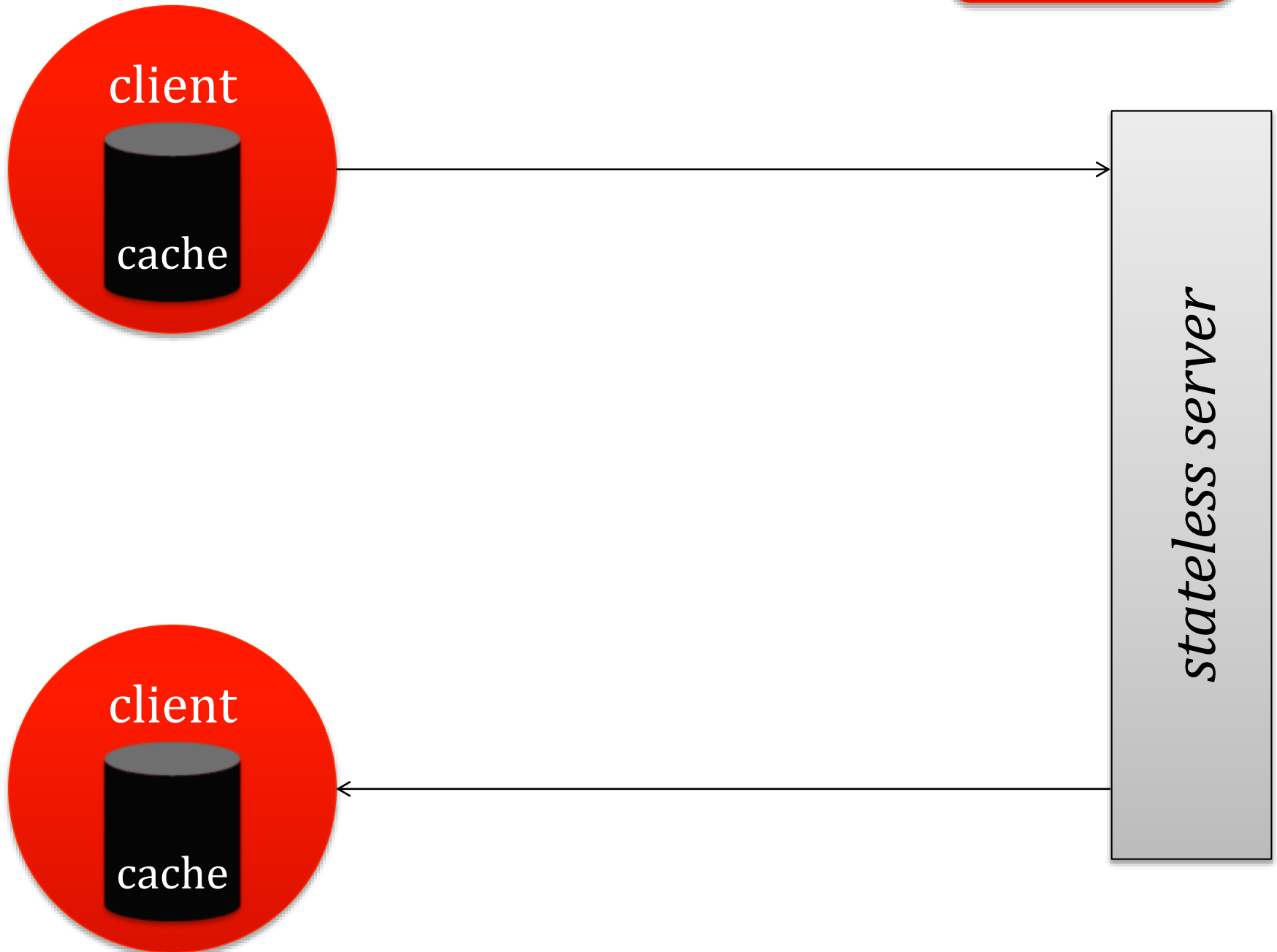
stateless server

rest

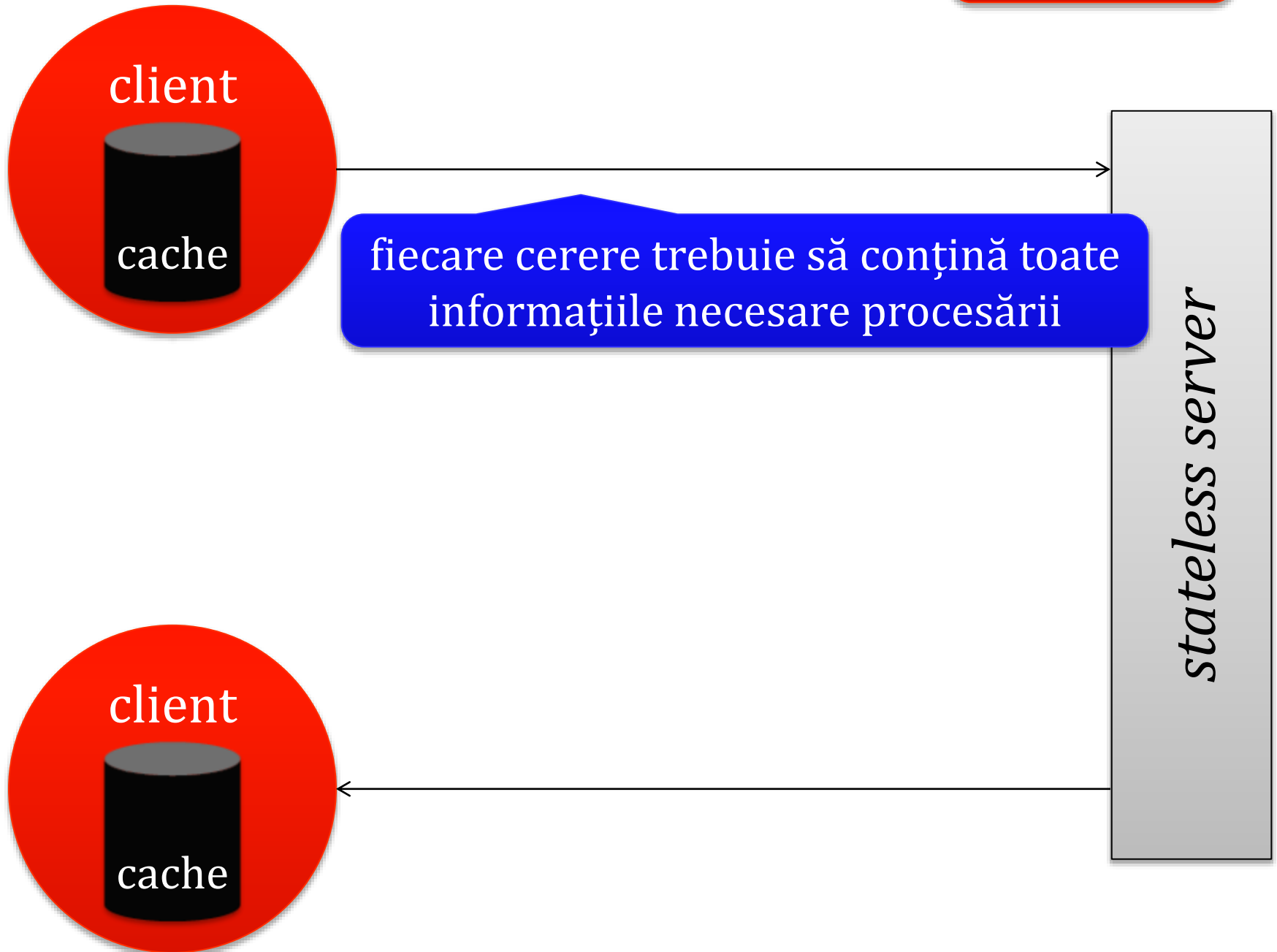
Reprezentările de resurse pot fi stocate temporar

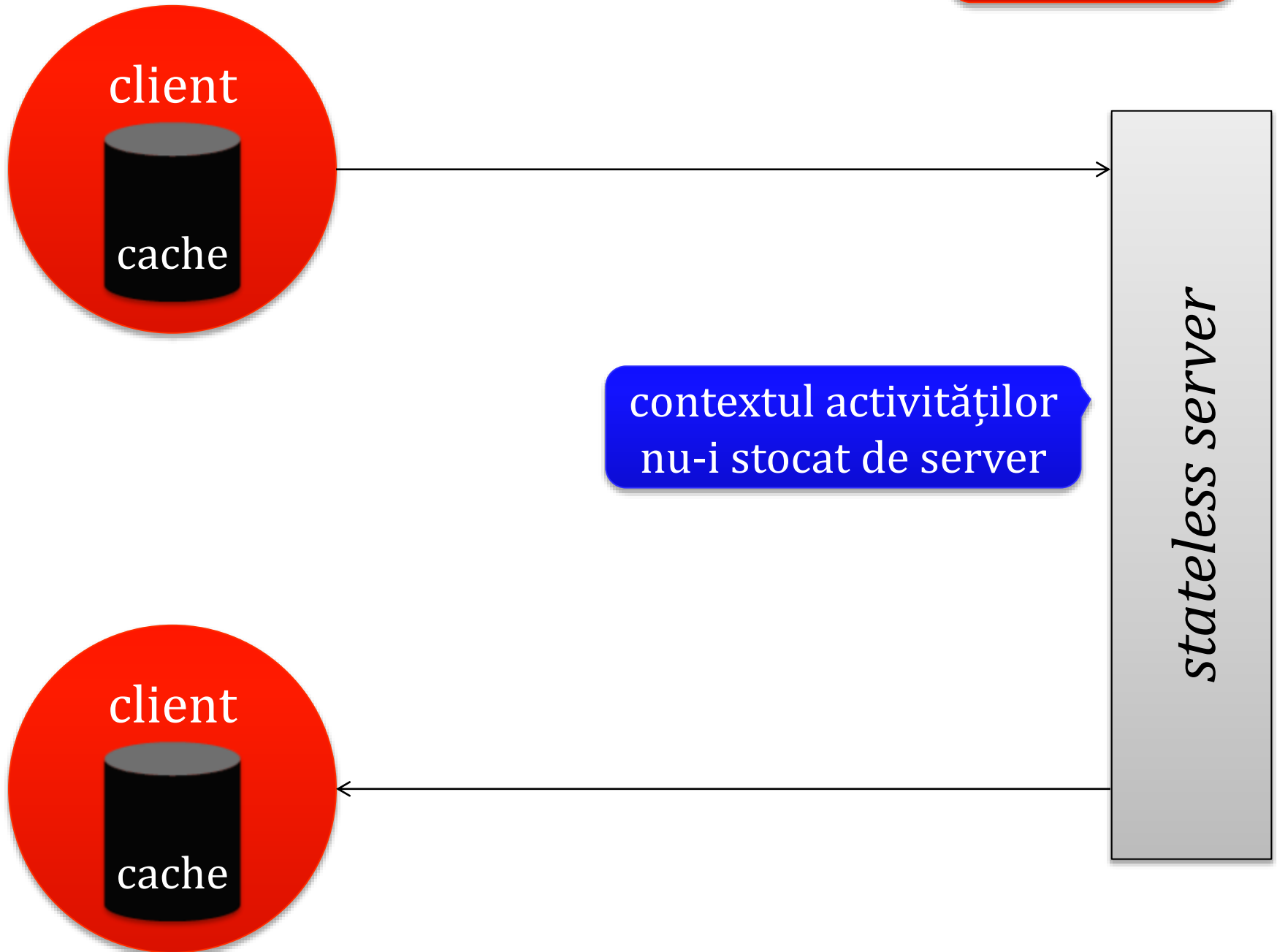
caching

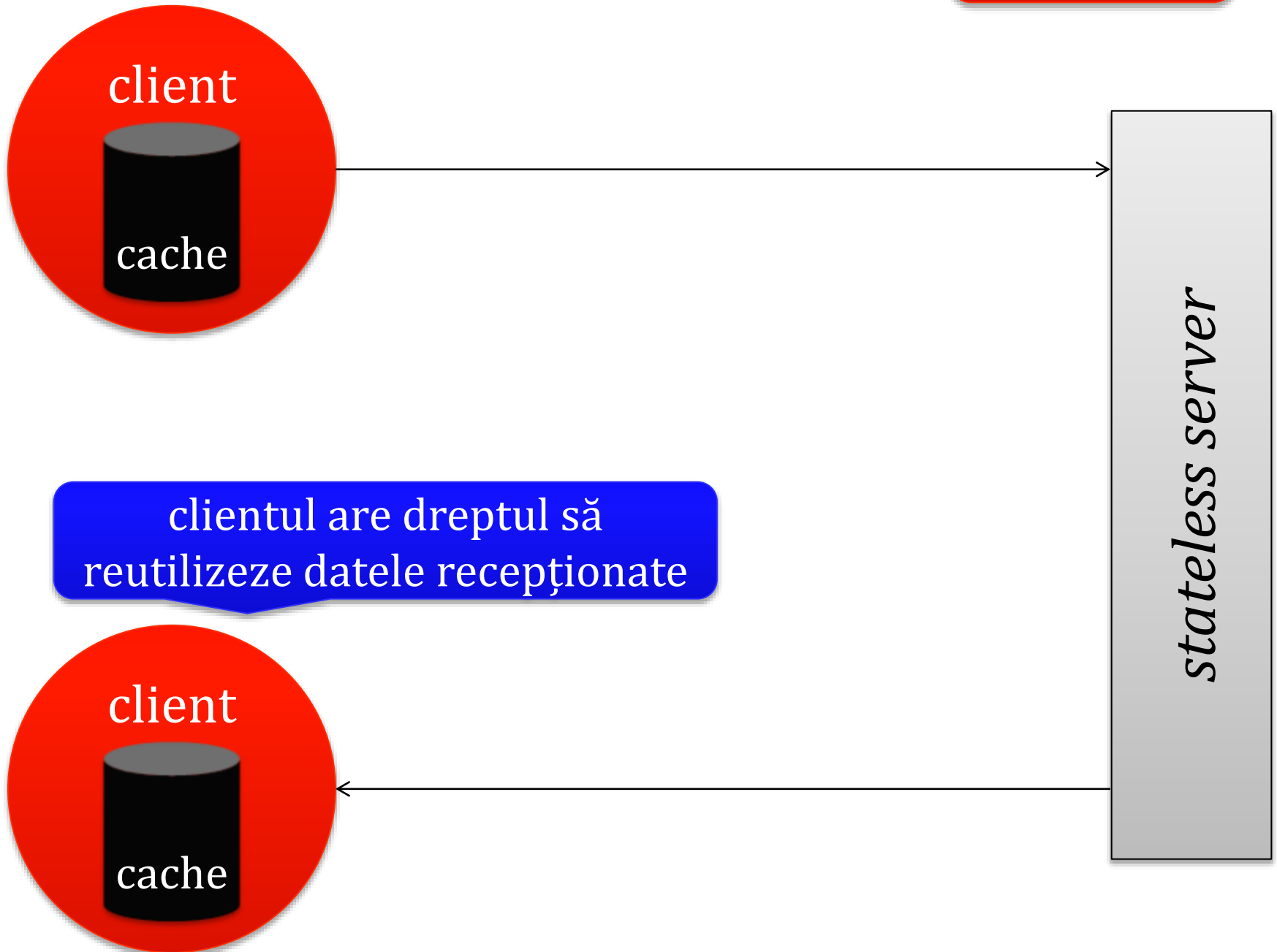
avansat



adaptare după B. Mulloy (2012)





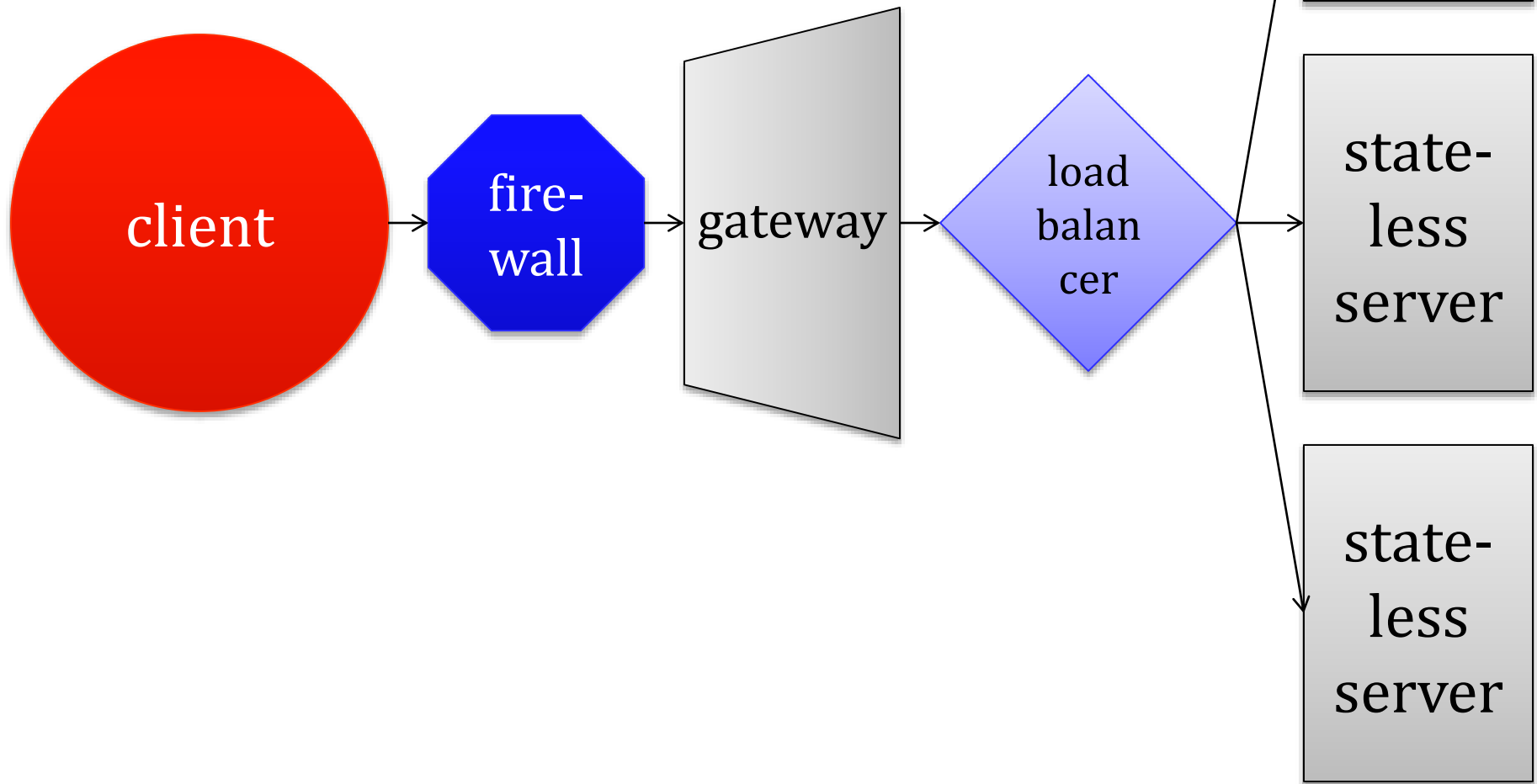


rest

Aplicația Web dezvoltată va fi stratificată

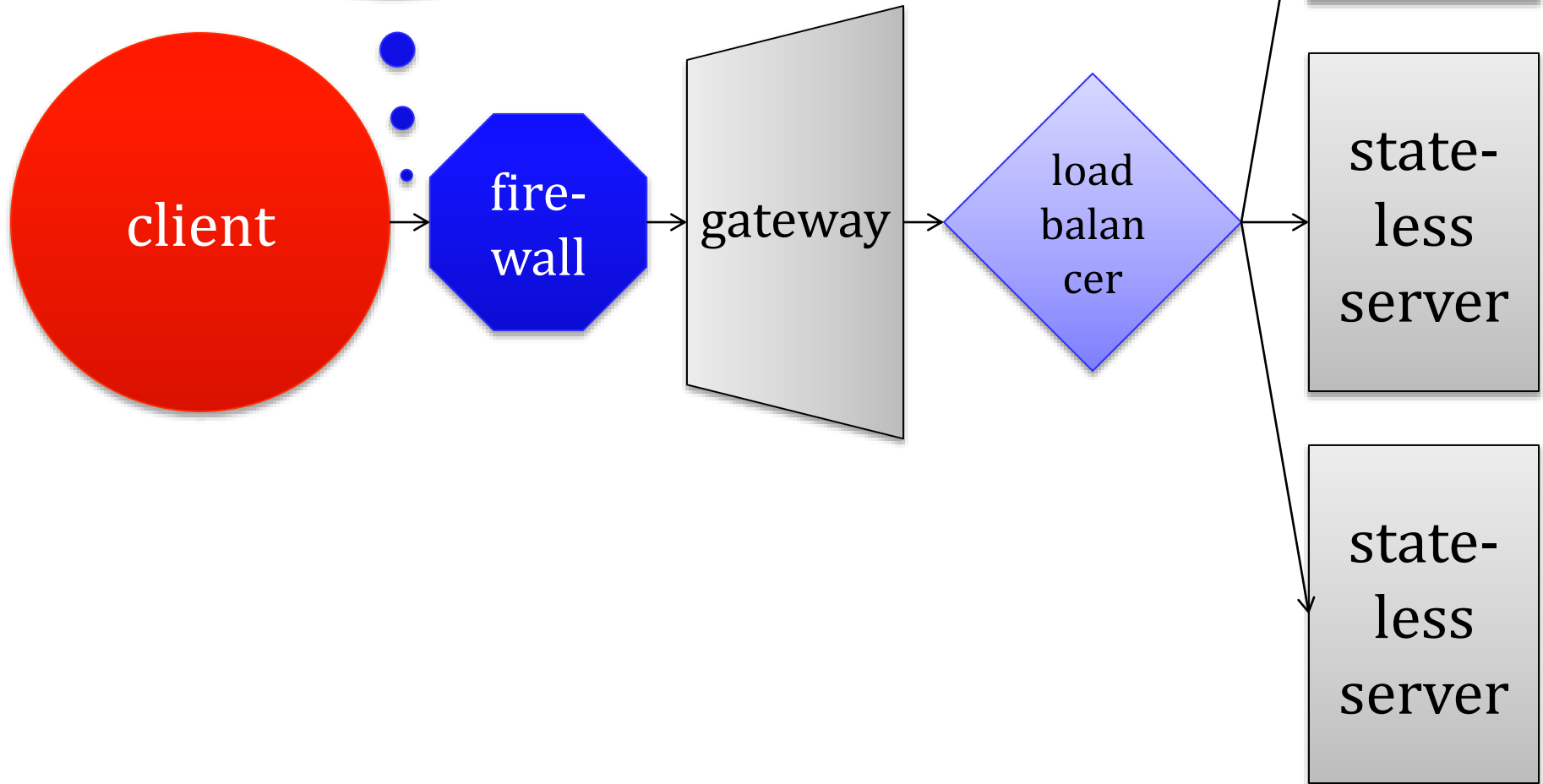
layered system

avansat

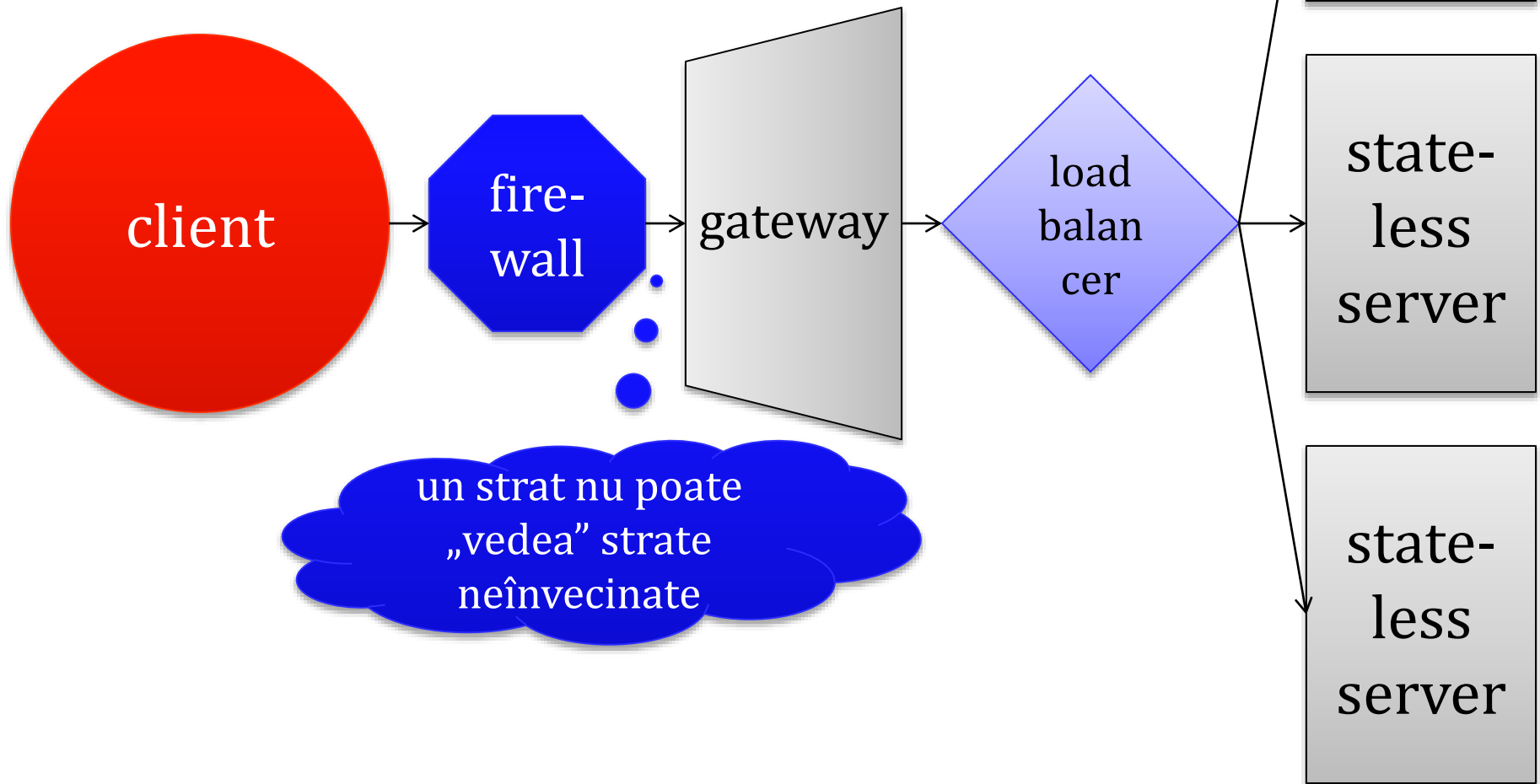


avansat

fiecare strat oferă
servicii stratelor
vecine

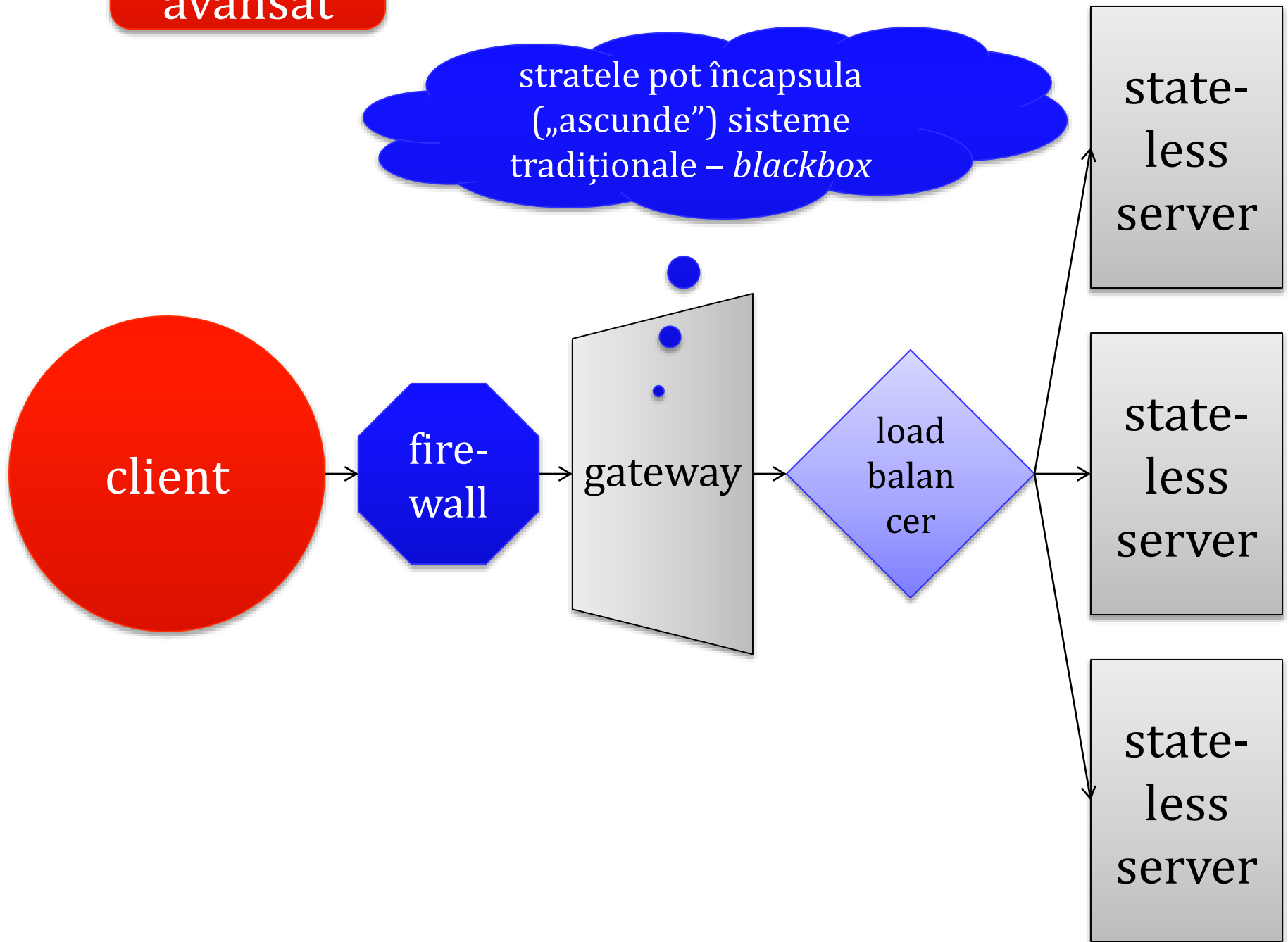


avansat

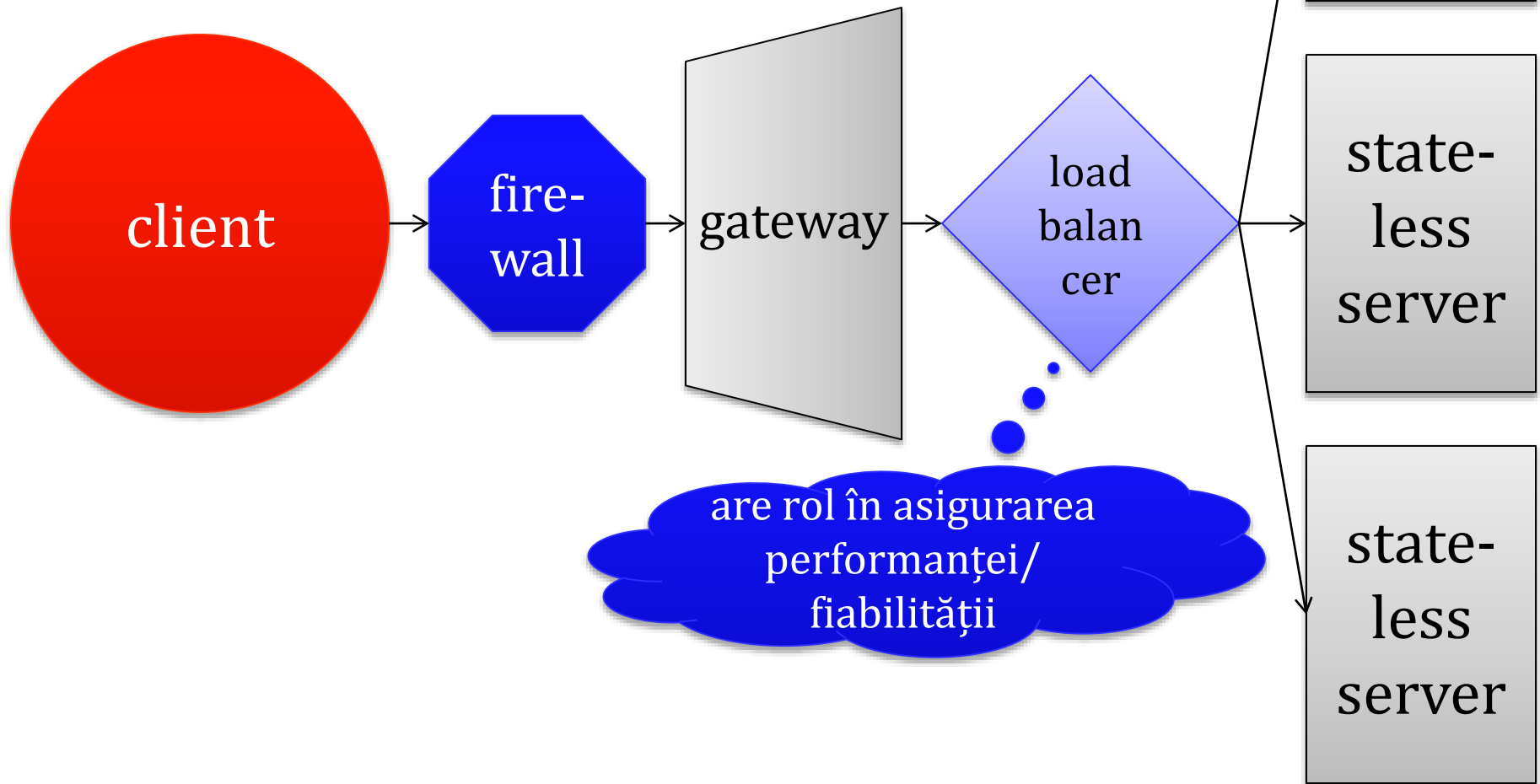


avansat

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



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:

furnizeazaSortim(), adaugaSortim(), listeazaSortim(), cautaSortim()

operații ce vizează managementul utilizatorilor:

furnizeazaUtiliz(), adaugaUtiliz(), stergeUtiliz(), cautaUtiliz(),...

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/portocala>

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/portocala/albastra>

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

Resursa	URL	Metoda	Reprezentare
<i>Bookmark</i>	/bookmarks/{md5}	GET	application/bookmark+xml
<i>Bookmark</i>	/bookmarks/{md5}	PUT	application/bookmark+xml
<i>Bookmark</i>	/bookmarks/{md5}	DELETE	
Lista de adrese	/bookmarks	GET	application/atom+xml
Lista de utilizatori	/users	GET	application/atom+xml
Lista de tag-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/a211528fb5108cddaa4b0d3aecbdbdcf"/>

<summary>

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

</summary>

</entry>

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

</feed>

digest MD5

obținerea
bookmark-urilor

GET /bookmarks/a211528fb5108cddaa4b0d3aecdbdcf

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/a211528fb5108cddaa4b0d3aeccdbdcf

PUT /bookmarks/a211528fb5108cddaa4b0d3aeccdbdcf

Content-type: application/bookmark+xml

...

200 OK

actualizarea
unui *bookmark*

rest

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

Reprezentările sunt interconectate prin URL-uri

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

rest

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

Reprezentările sunt interconectate prin URL-uri

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

Transferul de date poate fi și asincron – stil Ajax/Comet



vezi cursul viitor

rest

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

operații **CRUD** – *Create, Retrieve, Update, Delete*

Putem adopta o metodologie vizând dezvoltarea de servicii Web via REST?

rest: metodologie

Divizarea în resurse a setului de date
ale problemei

clase tipice de resurse:

Utilizator

Document – alternative: **Fotografie, Prodos, Software,...**

Metadata – *e.g.*, **Comentariu, Format, Locatie, Platforma** etc.

rest: metodologie

„Numirea” prin URI a fiecărei resurse

exemplificări:

<http://aplicatie.info/Utilizator/tux>

<http://aplicatie.info/Document/pinguini-cu-mere-albastre>

rest: metodologie

„Numirea” prin URI a fiecărei resurse

cazuri concrete:

accesarea datelor despre o producție cinematografică

<http://www.imdb.com/title/tt0401383/>

acces la prezentările Slideshare ale utilizatorului **busaco**

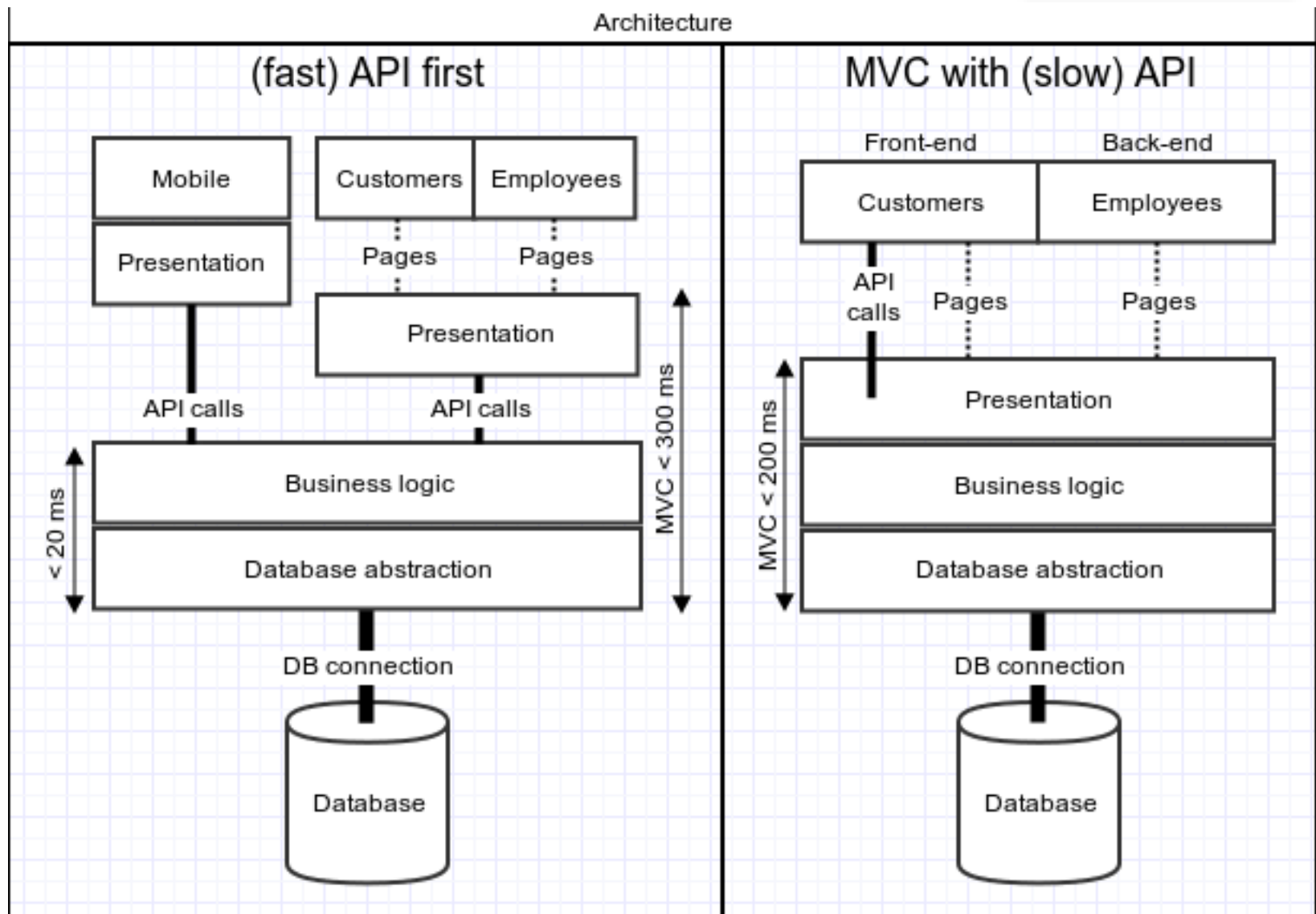
<http://www.slideshare.net/busaco/presentations>

obținerea listei utilizatorilor ce urmăresc o persoană

<http://twitter.com/followers>

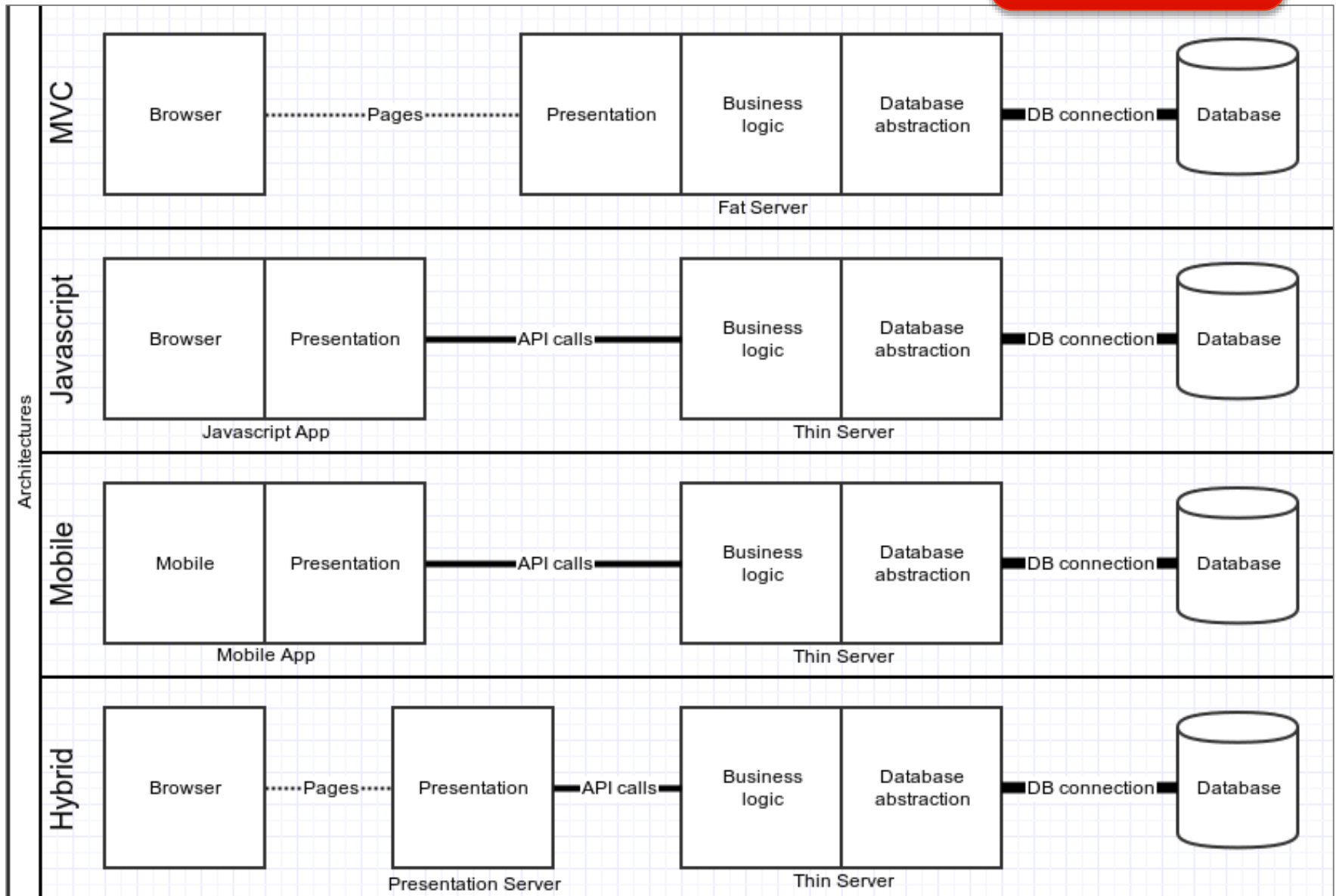
rest: metodologie

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



strategia *API first* adoptată de Twitter (van der Schee, 2013)

www.leaseweblabs.com/2013/10/api-first-architecture-fat-vs-thin-server-debate/



diverse abordări vizând arhitectura aplicațiilor Web

rest: metodologie

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

de considerat formate standard – *e.g.*, HTML, Atom, JSON

rest: metodologie

Integrarea resurselor
via legături hipertext + formulare

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

DELETE /Users/{id}/accessTokens

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 – aici Users

<http://myapi-strongdemo.rhcloud.com/explorer/>

POST /Users

Create a new instance of the model

avansat

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": ""
}
```

Response Content Type application/json

Parameters

Parameter	Value	Description	Parameter Type	Data Type
data	<div><div></div><div>Parameter content type: application/json application/json 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!

StrongLoop API
testarea interactivă a API-ului
vezi <http://strongloop.com/strongblog/node-js-rest-api-openshift-redhat/>

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”

colecții de resurse (uzual, la plural)

/students

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

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

Tratarea erorilor

folosirea codurilor de stare HTTP

200 – *OK*

303 – *See Other*

400 – *Bad Request*, 404 – *Not Found*

500 – *Internal Server Error*

2xx success

avansat

200

standard response for successful HTTP requests

201

request has been fulfilled; new resource created

202

request accepted, processing pending

203

request processed, information may be from another source

204

request processed, no content returned

205

request processed, no content returned, reset document view

206

partial resource return due to request header

207

XMLI, can contain multiple separate responses

208

results previously returned

226

request fulfilled, reponse is instance-manipulations

<http://httpstatus.es/>

3xx redirection

300

multiple options for the resource delivered

301

this and all future requests directed to the given URI

302

temporary response to request found via alternative URI

303

permanent response to request found via alternative URI

304

resource has not been modified since last requested

305

content located elsewhere, retrieve from there

rest: privire pragmatică

Tratarea erorilor

mesajele returnate trebuie să includă informații utile

exemplu: **Twilio** – cod de stare HTTP întors: 401

```
{ "status":    "401",  
  "message":  "Authenticate",  
  "code":     20003,  
  "more info": "http://www.twilio.com/docs/errors/20003"  
}
```


rest: privire pragmatică

Controlul versiunilor API-ului dezvoltat

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

specificarea versiunii
în antetul HTTP vs. în cadrul URL-ului

<http://feeds.delicious.com/v2/{format}/{username}>

Facebook – **?v=1.0**

rest: privire pragmatică

Paginarea și oferirea de răspunsuri parțiale

uzual, 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

rest: privire pragmatică

Paginarea și oferirea de răspunsuri parțiale

exemplificări reale:

Delicious – [/v1/posts/recent?count=30&tag=web](#)

Twitter – [api.twitter.com/1.1/search/tweets.json?q=...&since_id=24012619984051000&max_id=250126199840518145&result_type=mixed&count=4](#)

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

avansat

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	<input type="text"/>		

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*
<http://developer.nytimes.com/>

Response Headers [Select content](#)

```
X-Mashery-Responder: prod-j-worker-atl-01.mashery.com
Server: nginx/1.4.1
Date: Tue, 30 Sep 2014 12:23:20 GMT
Content-Type: application/json; charset=UTF-8
Content-Length: 34238
X-Powered-By: PHP/5.5.10
Last-Modified: Tue, 16 Sep 2014 22:34:59 GMT
Etag: "Tue, 16 Sep 2014 22:34:59 GMT"
Cache-Control: max-age=7200
Accept-Ranges: bytes
X-Varnish: 1713323640
Age: 0
Via: 1.1 varnish
X-Cache: MISS
```

diverse (meta-)date
oferite de serverul Web

Response Body [Select content](#)

```
{, {
  "nyt_movie_id": 467021,
  "display_title": "12 Years a Slave",
  "sort_name": "12 Years a Slave",
  "mpaa_rating": "R",
  "critics_pick": 1,
  "thousand_best": "0",
  "byline": "Manohla Dargis",
  "headline": "The Blood and Tears, Not the Magnolias",
  "capsule_review": "",
  "summary_short": "Steve McQueen&rsquo;s &ldquo;12 Years a Slave,&u201d based on the true story of
Solomon Northup, drives straight to the heart of the cruelties of oppression.",
  "publication_date": "2013-10-18",
  "opening_date": "2013-10-18",
  "dvd_release_date": "2014-03-04",
  "date_updated": "2013-10-17 08:00:09",
  "seo_name": "12-Years-a-Slave",
  "link": {
    "type": "article",
    "url": "http://movies.nytimes.com/2013/10/18/movies/12-years-a-slave-holds-nothing-
back-in-show-of-suffering.html",
    "suggested_link_text": "Read the New York Times Review of 12 Years a Slave"
  },
  "related_urls": [{
```

... răspuns în
format JSON

rest: privire pragmatică

Paginarea și oferirea de răspunsuri parțiale

exemplificări reale:

Delicious – [/v1/posts/recent?count=30&tag=web](#)

Twitter – [api.twitter.com/1.1/search/tweets.json?q=...&since_id=24012619984051000&max_id=250126199840518145&result_type=mixed&count=4](#)

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

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

Apache HttpComponents (Java): <http://hc.apache.org/>

hackage (Haskell): <http://hackage.haskell.org/package/HTTP>

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

httplib (Python 2) + **http.client** (Python 3)

rest: implementare

Biblioteci/API-uri implementând HTTP

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

neon (bibliotecă C): <http://www.webdav.org/neon/>

rest (Node.js): <https://www.npmjs.org/package/rest>

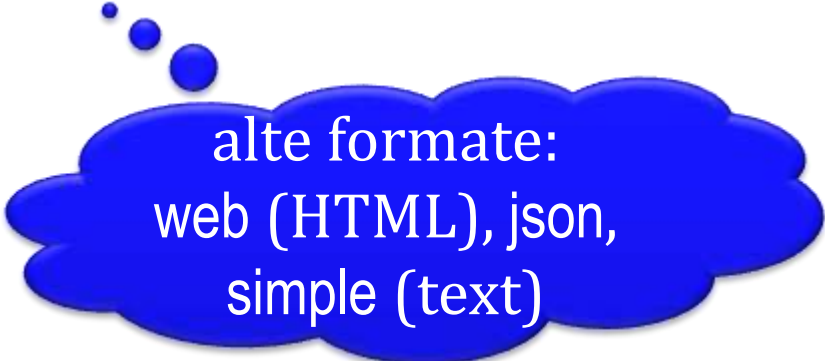
RestKit (pentru iOS): <http://restkit.org/>

RestSharp (pentru .NET): <http://restsharp.org/>

rest: implementare – exemplul 1

Studiu de caz: invocarea unui serviciu Web
de prescurtare de URL-uri – <http://is.gd/>

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



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

Cererea HTTP ce invocă serviciul Web prin REST:

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

Cererea HTTP ce invocă serviciul Web prin REST:

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: Fri, 08 May 2015 12:27:00 GMT

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

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

<output>

<shorturl>http://is.gd/DBmG2L</shorturl>

</output>



reprezentare POX
(Plain Old XML)

apelarea serviciului Web REST via PHP

```
// inițializăm cURL
$c = curl_init ();
// stabilim URL-ul serviciului Web invocat
curl_setopt ($c, CURLOPT_URL,
    'http://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, 1);
// preluăm resursa oferită de server (aici, un document XML)
$res = curl_exec ($c);
// închidem conexiunea cURL
curl_close ($c);
// procesăm rezultatul via DOM
$doc = new DOMDocument ();
$doc->loadXML ($res);
// preluăm conținutul elementului <shorturl>
$urls = $doc->getElementsByTagName ('shorturl');
foreach ($urls as $url) {
    echo '<p>Adresa prescurtată este: ' . $url->nodeValue . '</p>';
}
```

rest: implementare – exemplul 1



rest: implementare – exemplul 2

Studiu de caz: acces la Wikipedia via REST

folosim URL-ul

[http://en.wikipedia.org/w/api.php?action=query&
prop=categories&titles=Category:Arts&format=format](http://en.wikipedia.org/w/api.php?action=query&prop=categories&titles=Category:Arts&format=format)

pentru a obține date despre categoriile asociate artelor
în diverse formate (JSON, XML, text obișnuit,...)

detalii la www.mediawiki.org/wiki/API

HTTP/1.1 200 OK
Server: nginx/1.1.19
Date: Thu, 07 May 2015 17:01:59 GMT
Content-Type: **text/xml**; charset=utf-8

```
<api>  
  <query>  
    <pages>  
      <page pageid="4892515" ns="14" title="Category:Arts">  
        <categories>  
          <cl ns="14" title="Category:Creativity"></cl>  
          <cl ns="14" title="Category:Culture"></cl>  
          <cl ns="14" title="Category:Humanities"></cl>  
          <cl ns="14" title="Category:Main topic classifications"></cl>  
        </categories>  
      </page>  
    </pages>  
  </query>  
</api>
```

răspuns XML furnizat de serviciul Web
implementat de Wikipedia

```
{
  query: {
    pages: {
      4892515: {
        pageid: 4892515,
        ns: 14,
        title: "Category:Arts",
        categories: [
          {
            ns: 14,
            title: "Category:Creativity"
          },
          {
            ns: 14,
            title: "Category:Culture"
          },
          {
            ns: 14,
            title: "Category:Humanities"
          },
          {
            ns: 14,
            title: "Category:Main topic classifications"
          }
        ]
      }
    }
  }
}
```

reprezentarea
JSON
echivalentă

rest: implementare – exemplul 3

Studiu de caz: accesarea datelor publice
oferite de *Science Museum* (Londra)

se recurge la URL-ul

<http://api.sciencemuseum.org.uk/exhibitions/?output=json>
pentru a obține mesajele publice în format JSON

detalii la <http://api.sciencemuseum.org.uk/>

HTTP/1.1 200 OK

avansat

Date: Fri, 08 May 2015 12:29:47 GMT

Content-Type: application/json

Server: Microsoft-IIS/6.0

```
{ "exhibitions": [  
  {  
    "id" : "1031",  
    "name" : "Dan Dare & the Birth of Hi-tech Britain",  
    "admission_fee" : false,  
    "opened_on" : "2008-04-30",  
    "closed_on" : "2009-10-24",  
    "closed" : null,  
    "website" : "http://www.sciencemuseum.org.uk/..."  
  },  
  ...  
],  
  "maximum" : 1000,  
  "returned" : 91  
}
```

Cum putem valida
corectitudinea
răspunsului JSON obținut?

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

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 o interfață de programare (API) specifică
disponibilitate pe orice platformă

suport pentru REST via obiectul **XMLHttpRequest** (Ajax)
sau folosind WebSocket-uri (HTML5)

vezi cursul viitor

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
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)

rest: dezvoltare

ASP.NET MVC + Web API (C# *et al.*): www.asp.net/web-api

cujoJS, Express, Restify (Node.js)
<https://nodejsmodules.org/tags/rest>

JAX-RS – *Java Architecture for RESTful web Services*
<https://jax-rs-spec.java.net/>

Catalyst, Jifty, Mojolicious, REST::Client (Perl)

rest: dezvoltare

Restlet (Java): <http://restlet.org/>

Bottle, Cornice, Django, Eve, Flask, Pecan (Python)

Grape, RESTRack, Ruby on Rails (Ruby)
www.ruby-toolbox.com/categories/API_Builders

Epiphany, Fat-Free, Flight, FRAPI, Slim
(*micro-framework-uri* PHP)

...și multe altele

rest: dezvoltare

Servicii publice ce pot fi consumate via REST – exemple:
500px, AIDSInfo, Amazon, Basecamp, Blip.tv, DBpedia,
eBay, Ericsson, Facebook, GitHub, Google, LinkedIn,
Mastercard, Nodejitsu, Pipl, Quora, SlideShare, Tumblr,...



5722 (aprilie 2013), 3986 (mai 2012)

<http://tinyurl.com/2ssfc2>

rest: dezvoltare

Servicii publice ce pot fi consumate via REST – exemple:
500px, AIDSInfo, Amazon, Basecamp, Blip.tv, DBpedia,
eBay, Ericsson, Facebook, GitHub, Google, LinkedIn,
Mastercard, Nodejitsu, Pipl, Quora, SlideShare, Tumblr,...



inclusiv API publice disponibile pentru C++, C#, Java,
JavaScript, PHP, Python, Objective-C, Ruby,...

În cazul aplicațiilor Web sociale,
putea utiliza servicii Web
pentru autorizare și autentificare?

rest: dezvoltare

Pași uzuali de urmat pentru implementarea unei aplicații ce va invoca un serviciu Web pe baza unui API public:

(1) înregistrarea aplicației concepute
via situl entității furnizoare a serviciului

► cheie de acces – *API key, consumer key, developer key*

rest: dezvoltare

Pași uzuali de urmat pentru implementarea unei aplicații ce va invoca un serviciu Web pe baza unui API public:

- (2) pe baza acestei chei, aplicația se va putea autentifica pentru a putea fi autorizată să acceseze serviciul

rest: dezvoltare

Pași uzuali de urmat pentru implementarea unei aplicații ce va invoca un serviciu Web pe baza unui API public:

(2) pe baza acestei chei, aplicația se va putea autentifica pentru a putea fi autorizată să acceseze serviciul

pot fi impuse diverse politici de acces (*permissions*): doar consultare (*read*), posibilitatea editării etc.

rest: dezvoltare

Pași uzuali de urmat pentru implementarea unei aplicații ce va invoca un serviciu Web pe baza unui API public:

- (3) autentificarea și autorizarea aplicației
au loc cu acordul utilizatorului

rest: dezvoltare

Pași uzuali de urmat pentru implementarea unei aplicații ce va invoca un serviciu Web pe baza unui API public:

(3) autentificarea și autorizarea aplicației
au loc cu acordul utilizatorului

dacă utilizatorul nu este autentificat, i se vor solicita informațiile de autentificare (*e.g.*, nume + parola), apoi va putea autoriza aplicația să aibă acces la date via serviciul Web furnizat

rest: dezvoltare

Pași uzuali de urmat pentru implementarea unei aplicații ce va invoca un serviciu Web pe baza unui API public:

- (4) aplicația apelează funcționalitățile oferite de serviciu pentru preluarea/modificarea datelor de interes, conform politicilor de acces

rest: dezvoltare

Pași uzuali de urmat pentru implementarea unei aplicații ce va invoca un serviciu Web pe baza unui API public:

(4) aplicația apelează funcționalitățile oferite de serviciu

sesiunea curentă va fi stabilită și menținută pe baza unor **informații de autentificare** (*auth tokens*)

rest: dezvoltare – oauth

Autorizarea securizată a unei aplicații să acceseze date private într-un mod standardizat se poate realiza via **OAuth**

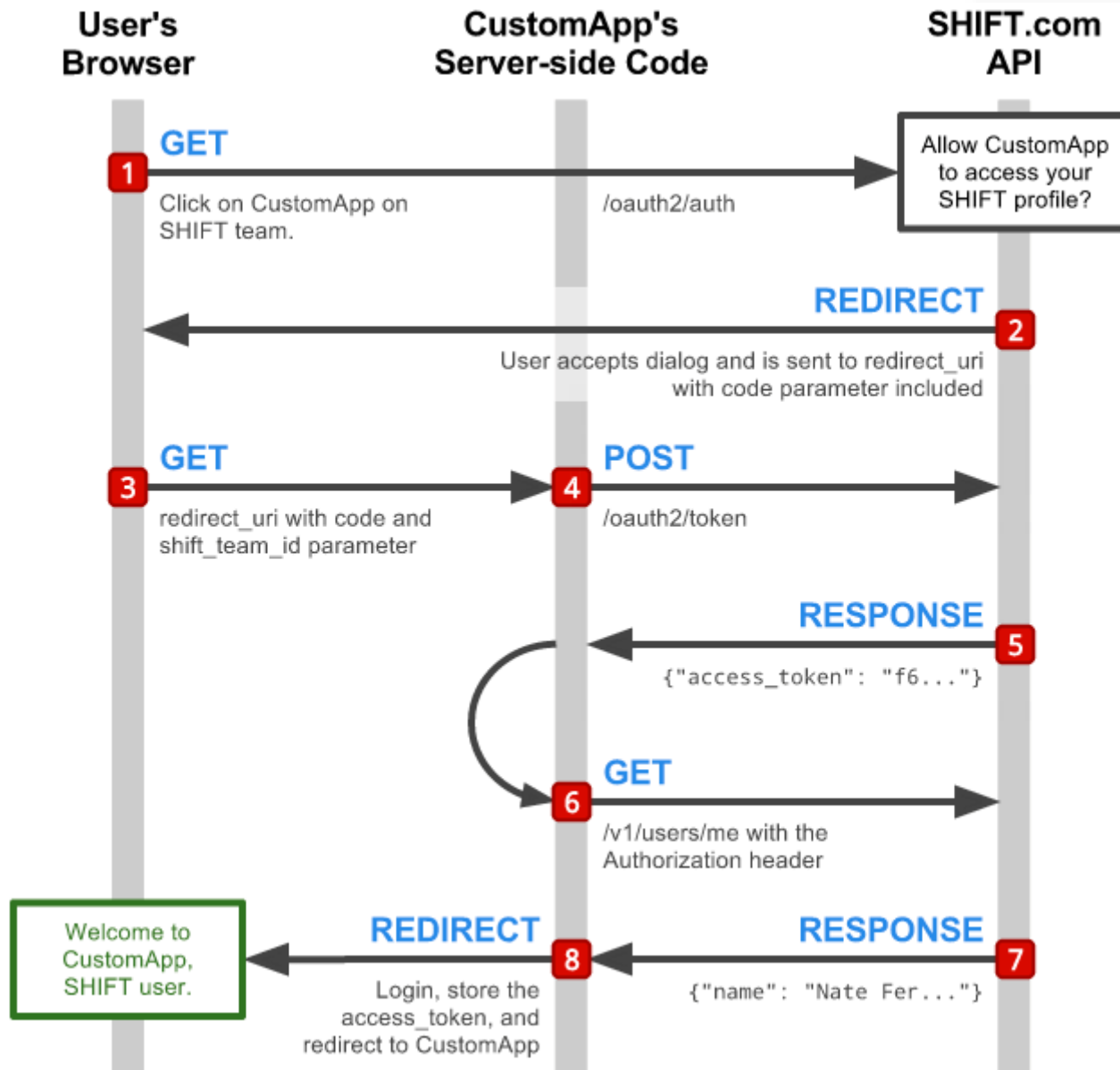
rest: dezvoltare – oauth

Autorizarea securizată a unei aplicații să acceseze date private într-un mod standardizat se poate realiza via **OAuth**

protocol deschis

OAuth 1.0 (2010), OAuth 2.0 (2012)

<http://oauth.net/2/>



utilizarea OAuth pentru o aplicație Web – <http://tinyurl.com/mm4ur9u>



exemplu concret – Facebook:

autorizare cu diverse permisiuni – *e.g.*, `age_range`,
`email` (acces la adresa de *e-mail* a unui utilizator),
`public_profile`, `read_mailbox`, `user_birthday`, `user_friends`,
`user_likes`, `user_photos`, `user_status`

<https://developers.facebook.com/docs/facebook-login/permissions/v2.0>

rest: dezvoltare – oauth

Dezvoltare Web:

diverse biblioteci disponibile pentru C#, Go, Erlang, Java, JavaScript, Objective-C, Perl, PHP, Python, Ruby,...

<http://oauth.net/code/>

avansat

adodson.com/hello.js/

www.webappers.com/2014/09/29/hello-js-javascript-sdk-authenticating-web-services/

rest: dezvoltare – openid

OpenID

mecanism descentralizat de autentificare a utilizatorului la nivel de Web pe baza paradigmei SSO – *Single Sign On*

utilizatorul poate demonstra că deține un URL specific menit a-l identifica *on-line* via un ofertant (serviciu) de identitate digitală (*identity provider*)
e.g., folosind o aplicație Web socială

<http://openid.net/get-an-openid/>



Fiecare identitate e desemnată de un URL (stabilit de *identity provider*):

nume.wordpress.com
google.com/profiles/me
etc.

pentru a-și confirma identitatea, utilizatorul va trebui să se autentifice:
nume de cont + parolă,
smart card,
date biometrice,

...

Please login to use API Designer

[I forgot my password](#)

Log in

Or log in with:



Twitter



Facebook



Google



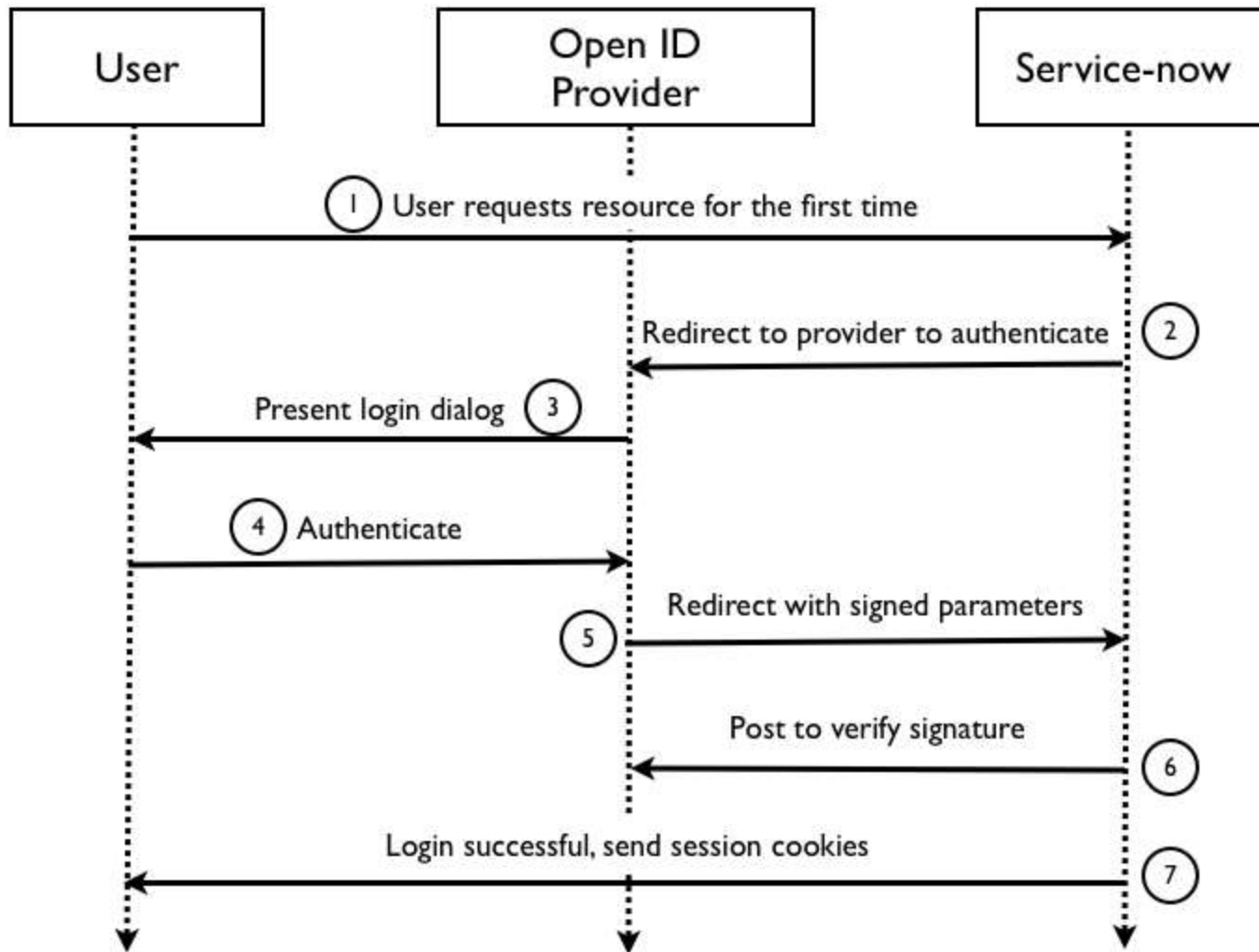
GitHub



OpenID

[Not registered?](#)

Create an account in seconds



rest: dezvoltare – openid

Biblioteci *open source* disponibile
pentru C, C#, Java, JavaScript, PHP, Python, Ruby,...

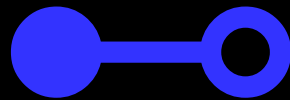
<http://openid.net/developers/libraries/>

o soluție alternativă: Mozilla Persona

<https://developer.mozilla.org/Persona>

Tehnologii Web

dezvoltare de servicii Web via REST



considerații privind autentificarea & autorizarea

episodul viitor: **suita de tehnologii Ajax** **aplicații Web hibride (*mash-ups*)**

