True REST design

Alexander Vinokurov

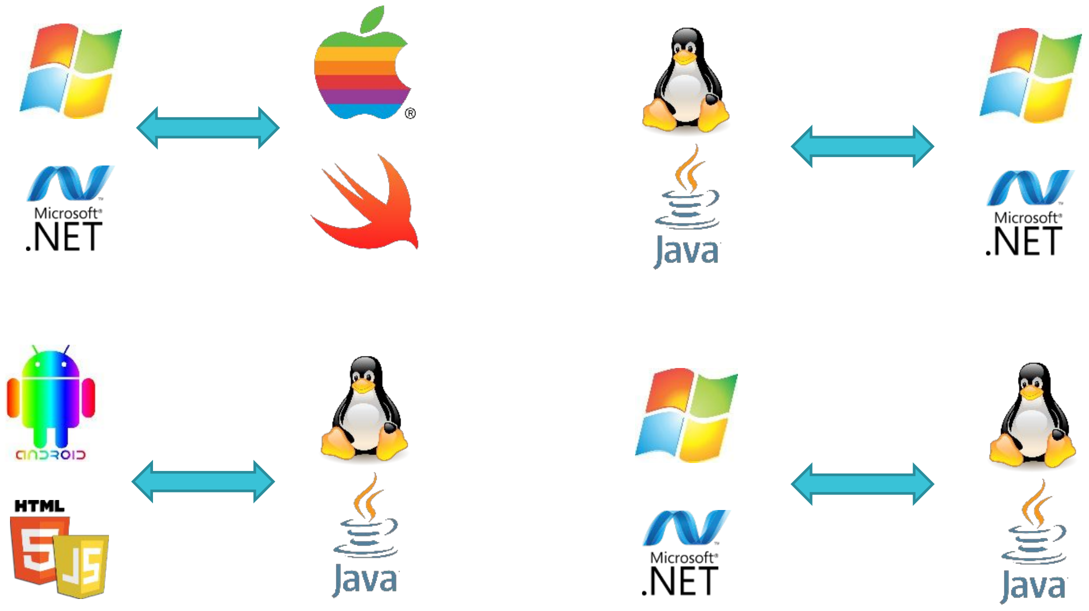
ALEXANDER VINOKUROV

Software Engineering Team Leader

- 13+ years in EPAM
- 10+ projects
- Full stack developer

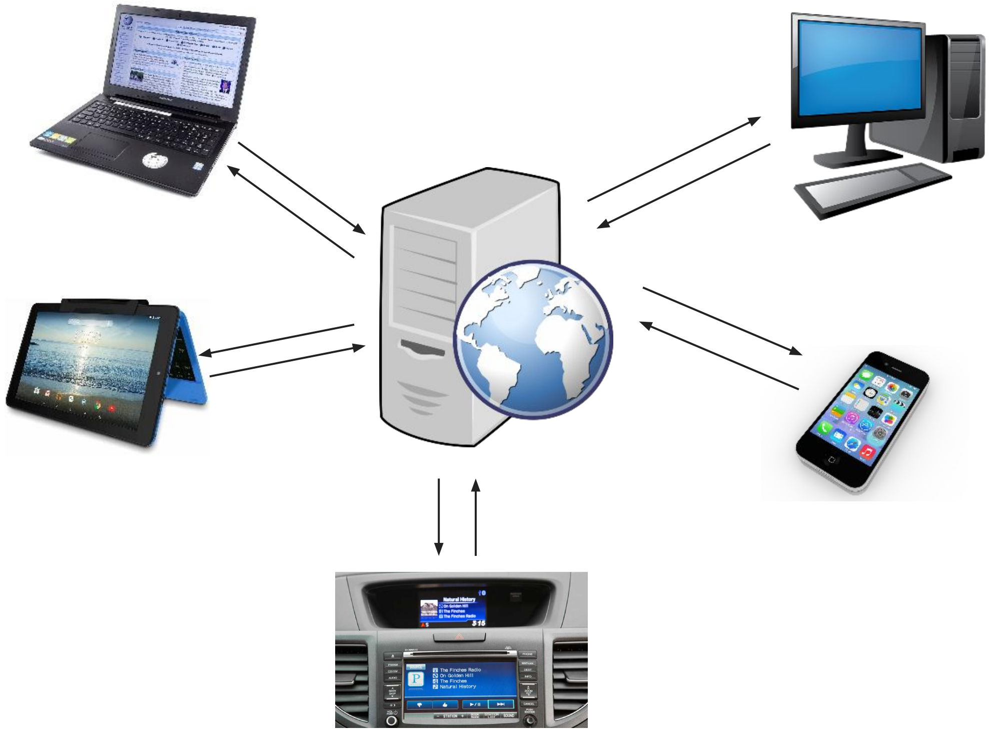
REST Drivers

HETEROGENOUS INTEROPERABILITY



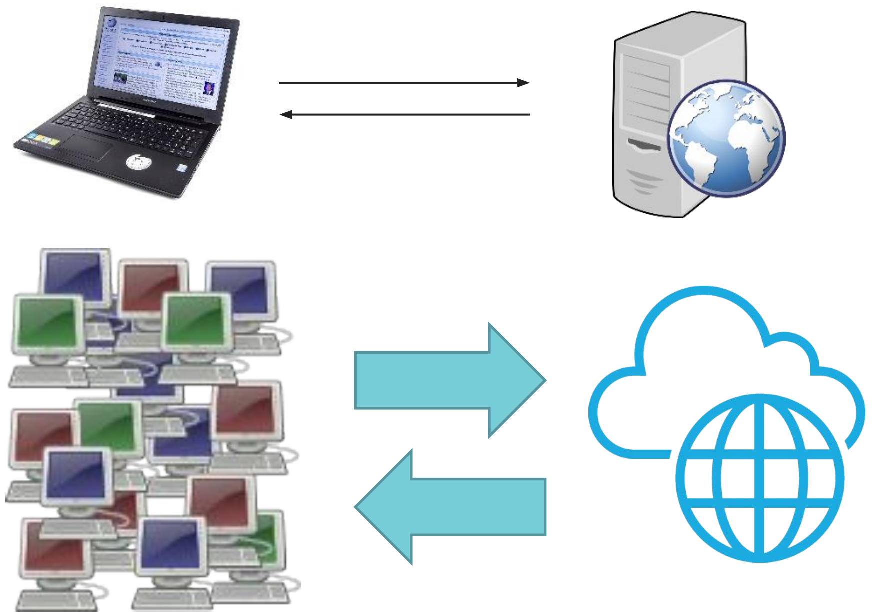
REST Drivers

DEVICES VARIETY



REST Drivers

CLOUD DISTRIBUTION



“It is critical to build a scalable architecture in order to take advantage of a scalable infrastructure”

REST Definition

UNIVERSITY OF COLIFORNIA, IRVINE

Architectural Styles and the Design of Network-based Software Architectures

DISSERTATION

in Information and Computer Science

by

Roy Thomas Fielding

2000

REST Definition

REST IS NOT:

- **RPC**
  - ❑ REST is not a way to call methods over a network without the overhead of SOAP and WSDL
- **HTTP**
  - ❑ An architecture implemented on top of HTTP is not inherently RESTful
- **URIs**
  - ❑ Having clean URLs does not make the architecture automatically RESTful
  - ❑ Hyper-focus on URIs can actually make designs non-RESTful

REST Definition

REST IS:

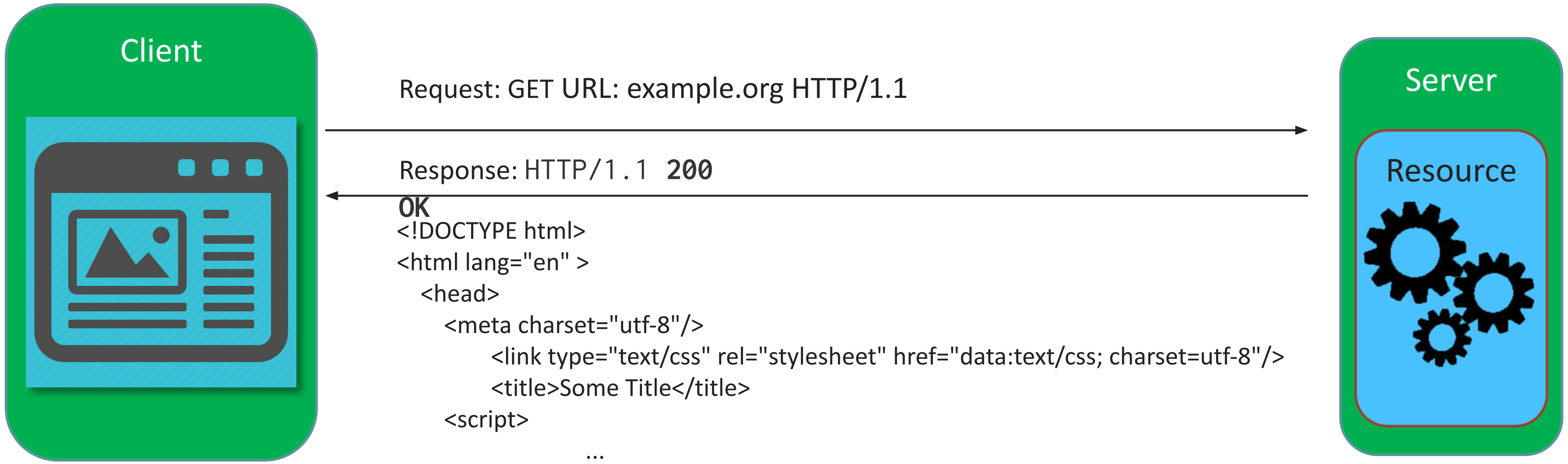
- **An Architectural Style (set of rules, constrains and recommendations)**
  - ❑ We use standards to implement it
  - ❑ Protocol Agnostic

- **Intended for long-lived network-based applications**

REST Definition

Передача репрезентативного состояния дает представление о том, как ведет себя хорошо спроектированное веб-приложение, где пользователь перемещается по приложению путем выбора ссылок, в результате чего следующая страница передается пользователю и отображается для дальнейшего использования.

REST Definition



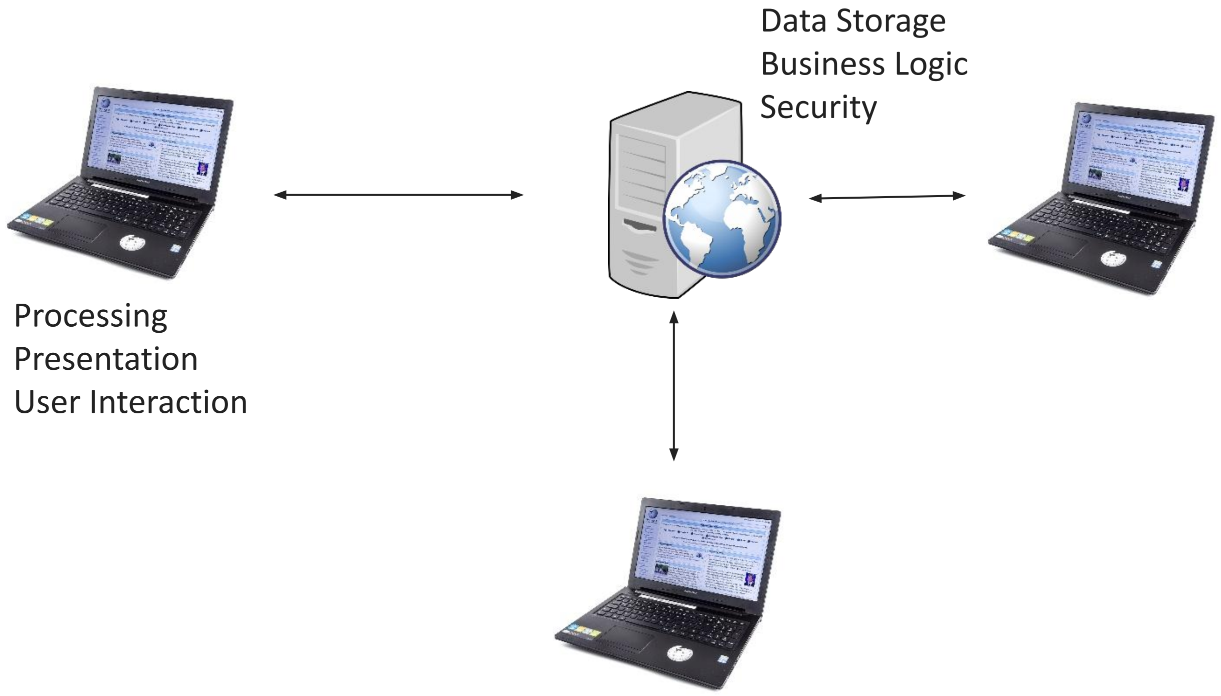
Request: GET URL: example.org HTTP/1.1

Response: HTTP/1.1 **200 OK**

```
<!DOCTYPE html>
<html lang="en" >
   <head>
      <meta charset="utf-8"/>
            <link type="text/css" rel="stylesheet" href="data:text/css; charset=utf-8"/>
            <title>Some Title</title>
      <script>
                        ...
      </script>
    <body>
   </body>
</html>
```

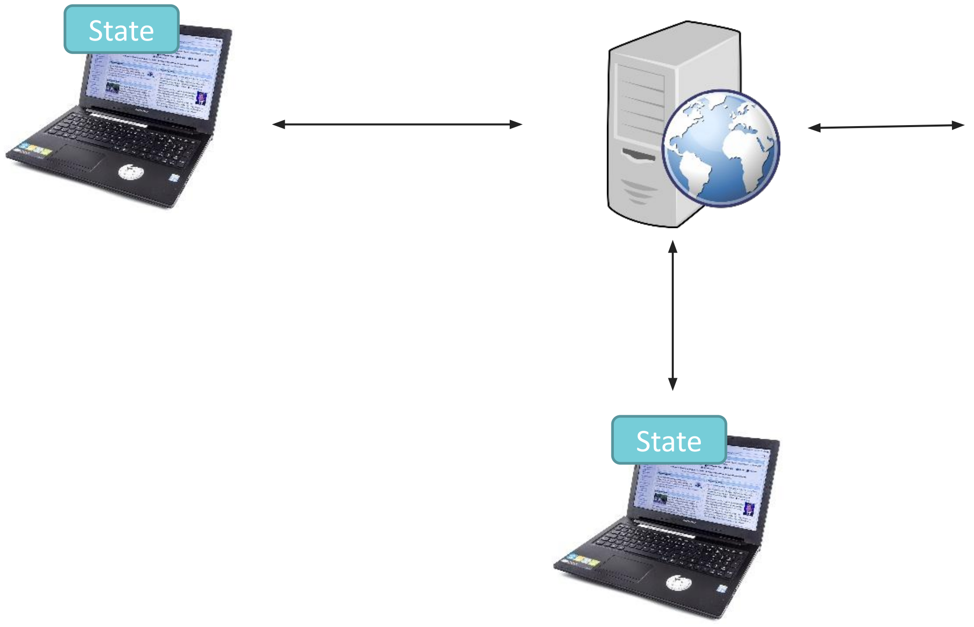
REST Constraints

1. Client-Server



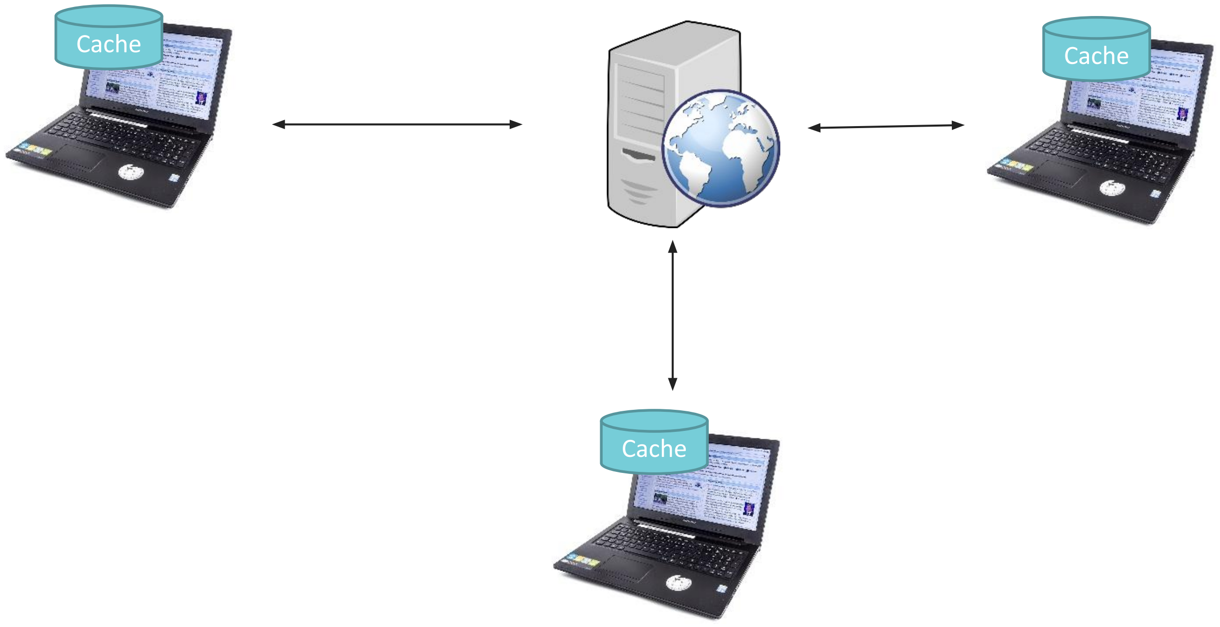
REST Constraints

1. Client-Server
2. **Stateless**



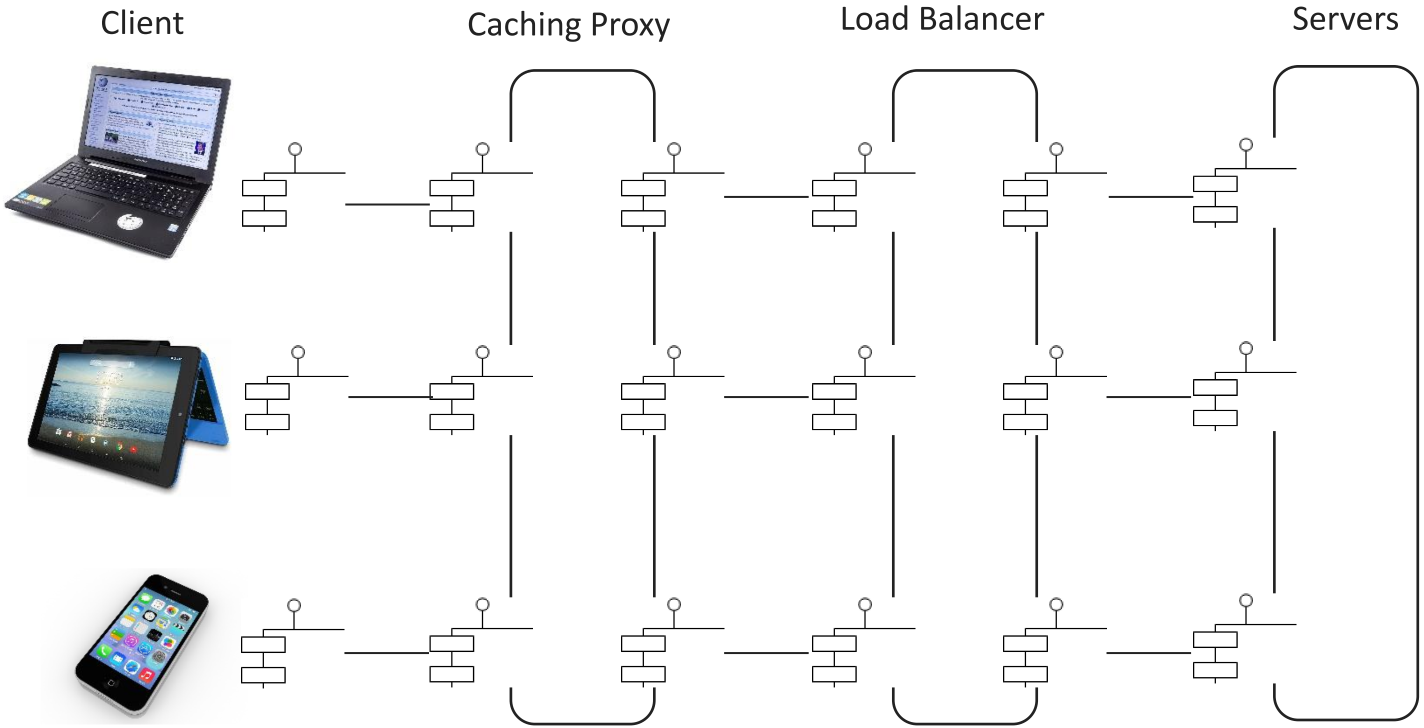
REST Constraints

1. Client-Server
2. Stateless
3. **Cache**



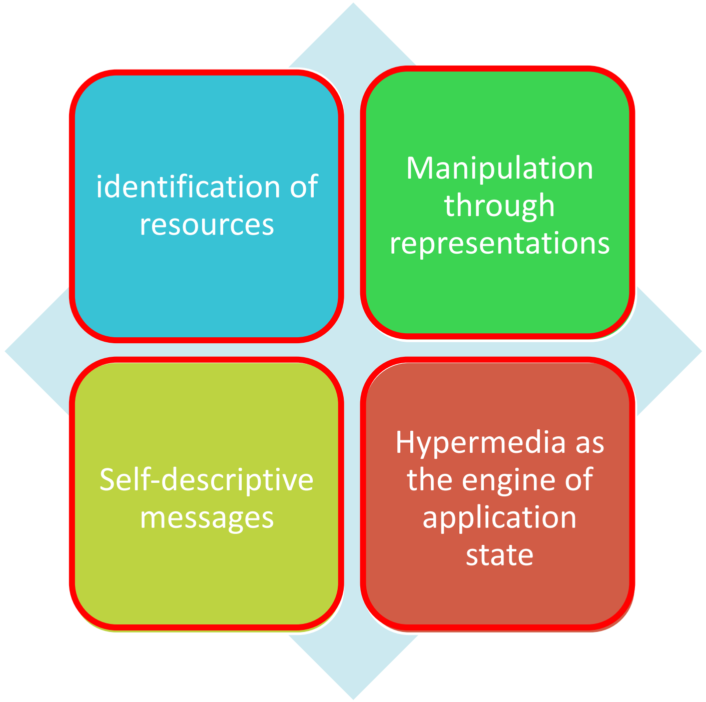
REST Constraints

1. Client-Server
2. Stateless
3. Cache
4. **Uniform Interface**



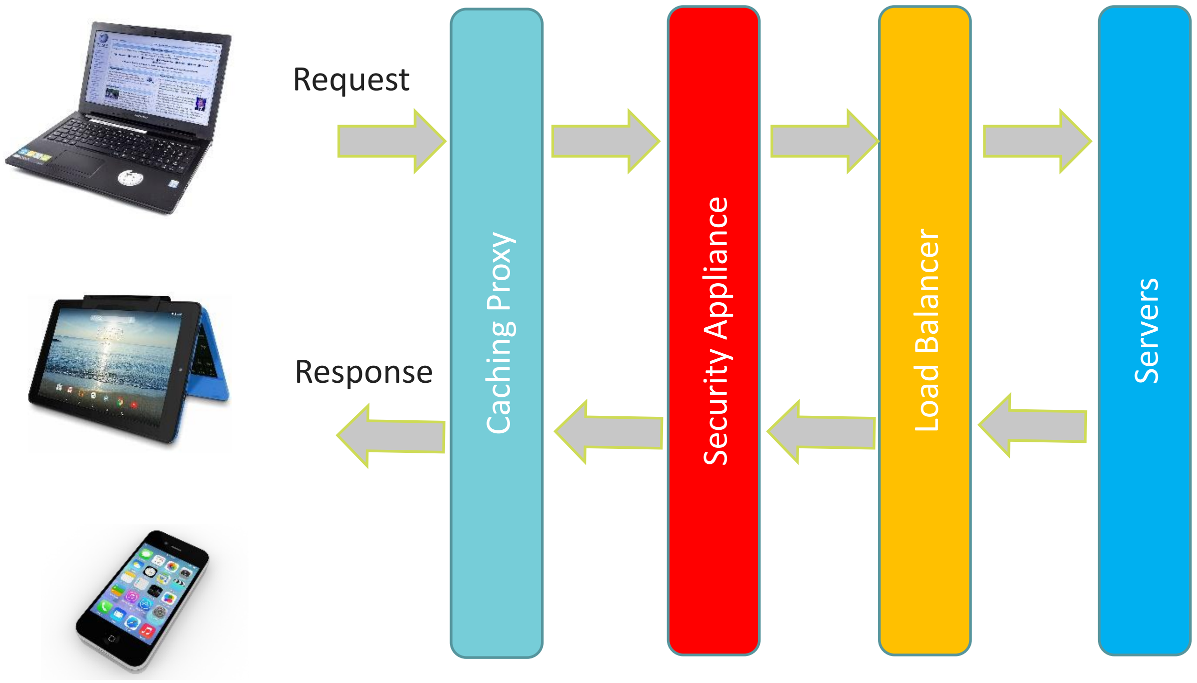
REST Constraints

1. Client-Server
2. Stateless
3. Cache
4. **Uniform Interface**


identification of resources
Manipulation through representations
Self-descriptive messages
Hypermedia as the engine of application state

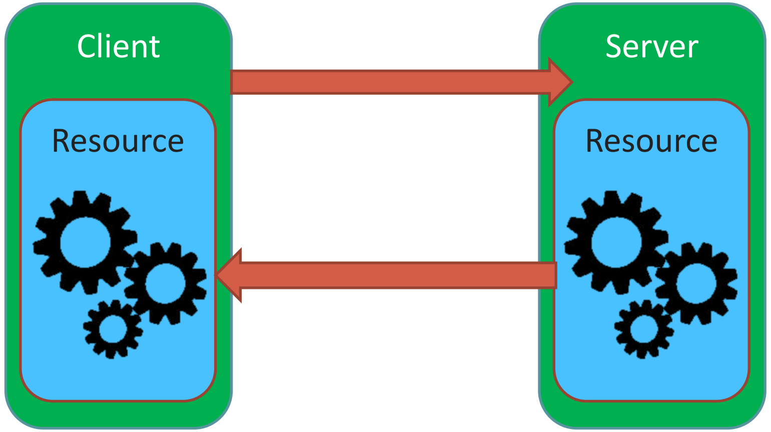
REST Constraints

1. Client-Server
2. Stateless
3. Cache
4. Uniform Interface
5. **Layered System**

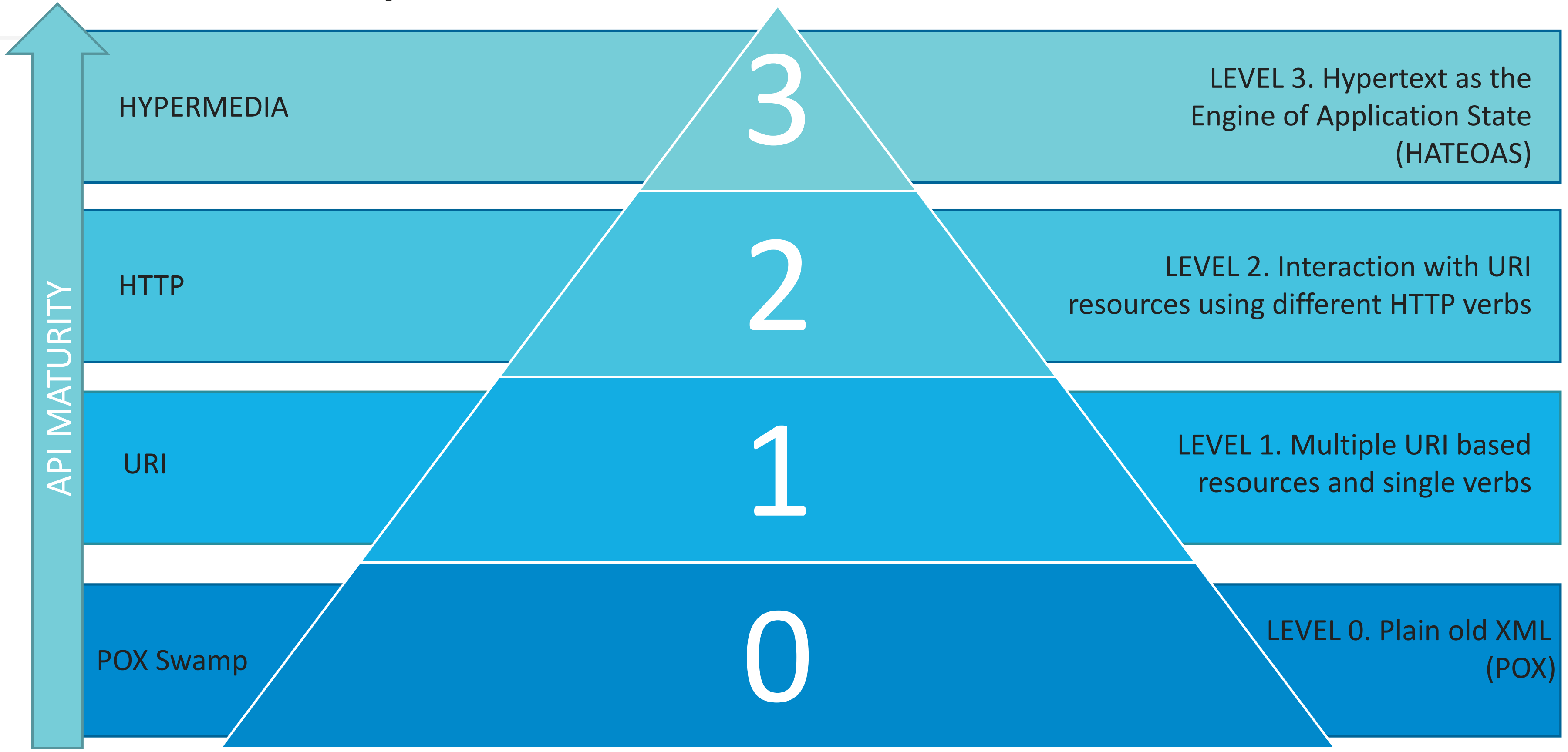


REST Constraints

1. Client-Server
2. Stateless
3. Cache
4. Uniform Interface
5. Layered System
6. **Code-On-Demand** **(optional)**



Richardson Maturity Model



API MATURITY
HYPERMEDIA
HTTP
URI
POX Swamp
3
2
1
0
LEVEL 3. Hypertext as the Engine of Application State (HATEOAS)
LEVEL 2. Interaction with URI resources using different HTTP verbs
LEVEL 1. Multiple URI based resources and single verbs
LEVEL 0. Plain old XML (POX)

Richardson Maturity Model. Level 0

```
POST /appointmentService HTTP/1.1

<openSlotRequest date = "2010-01-04" doctor =
"mjones"/>
```

```
HTTP/1.1 200 OK

<openSlotList>
     <slot start = "1400" end =
"1450">
        <doctor id = "mjones"/>
     </slot>
     <slot start = "1600" end =
"1650">
        <doctor id = "mjones"/>
     </slot>
 </openSlotList>
```

```
POST /appointmentService HTTP/1.1

<appointmentRequest>
  <slot doctor = "mjones" start = "1400" end =
"1450"/>
  <patient id = "jsmith"/>
</appointmentRequest>
```

```
HTTP/1.1 200 OK

<appointment>
  <slot doctor = "mjones" start = "1400" end =
"1450"/>
  <patient id = "jsmith"/>
</appointment>
```

```
HTTP/1.1 200 OK

<appointmentRequestFailure>
  <slot doctor = "mjones" start = "1400" end
= "1450"/>
  <patient id = "jsmith"/>
  <reason>Slot not available</reason>
</appointmentRequestFailure>
```

Richardson Maturity Model. Level 1

RESOURCES

```
POST /doctors/mjones HTTP/1.1

<openSlotRequest date = "2010-01-04"/>
```

```
HTTP/1.1 200 OK

<openSlotList>
  <slot id = "1234" doctor = "mjones" start = "1400" end = "1450"/>
  <slot id = "5678" doctor = "mjones" start = "1600" end = "1650"/>
</openSlotList>
```

```
POST /slots/1234 HTTP/1.1

<appointmentRequest>
  <patient id = "jsmith"/>
</appointmentRequest>
```

```
HTTP/1.1 200 OK

<appointment>
  <slot id = "1234" doctor = "mjones" start = "1400" end = "1450"/>
  <patient id = "jsmith"/>
</appointment>
```

Richardson Maturity Model. Level 2

HTTP VERBS

```
GET /doctors/mjones/slots?date=20100104&status=open
HTTP/1.1
```

```
HTTP/1.1 200 OK

<openSlotList>
  <slot id = "1234" doctor = "mjones" start = "1400" end = "1450"/>
  <slot id = "5678" doctor = "mjones" start = "1600" end = "1650"/>
</openSlotList>
```

```
POST /slots/1234 HTTP/1.1

<appointmentRequest>
  <patient id = "jsmith"/>
</appointmentRequest>
```

```
HTTP/1.1 201 Created

<appointment>
  <slot id = "1234" doctor = "mjones" start = "1400" end = "1450"/>
  <patient id = "jsmith"/>
</appointment>
```

```
HTTP/1.1 409 Conflict

<openSlotList>
  <slot id = "5678" doctor = "mjones" start = "1600" end = "1650"/>
</openSlotList>
```

Richardson Maturity Model. Level 3

Hypermedia Controls

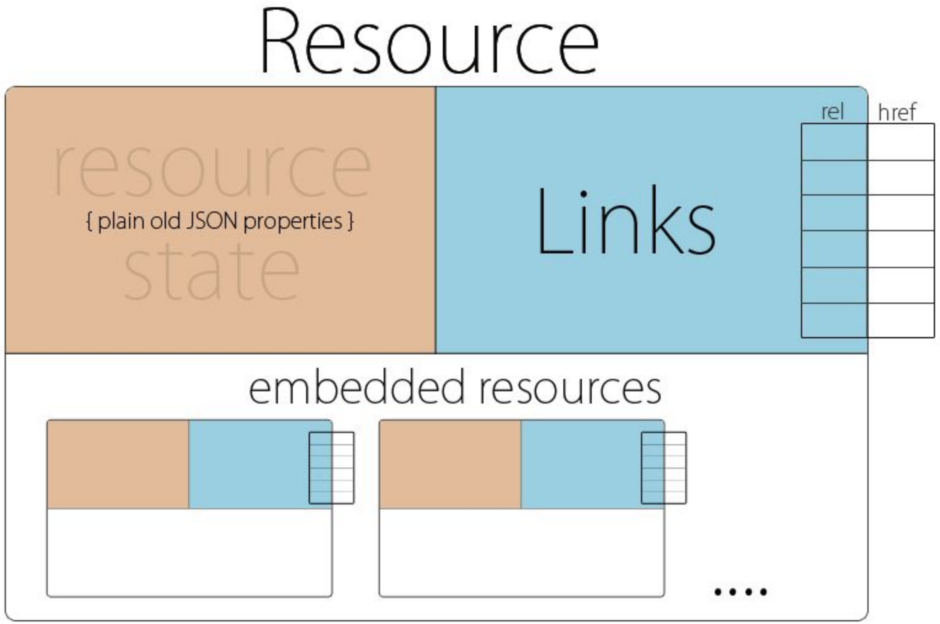
```
GET /doctors/mjones/slots?date=20190104&status=open
HTTP/1.1
```

```
HTTP/1.1 200 OK

<openSlotList>
  <slot id = "1234" doctor = "mjones" start = "1400" end = "1450">
     <link rel = "book" uri = "/slots/1234“ method=“post”/>
  </slot>
</openSlotList>
```

```
POST /slots/1234 HTTP/1.1

<appointmentRequest>
  <patient id = "jsmith"/>
</appointmentRequest>
```

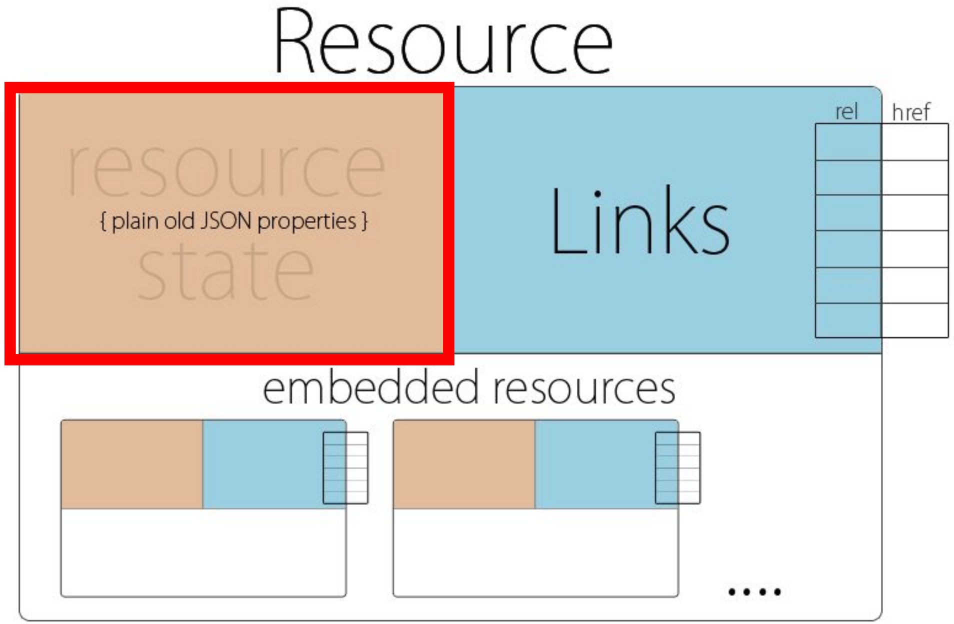
```
HTTP/1.1 201 Created

<appointment>
  <slot id = "1234" doctor = "mjones" start = "1400" end = "1450"/>
  <patient id = "jsmith"/>
  <link rel = "self " uri = "/slots/1234/appointment"/>
  <link rel = "cancel“ uri = "/slots/1234/appointment/cancel “/>
</appointment>
```

Hypermedia Application Language (HAL)



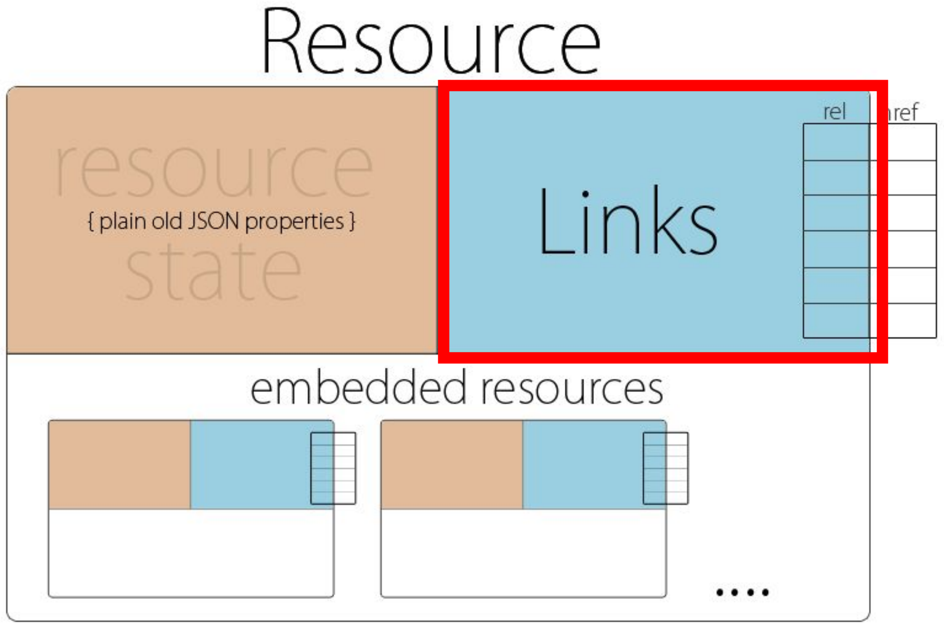
```
{
   "_links": {
      "self": { "href": "/orders" },
      "curies": [{ "name": "ea", "href": "http://example.com/docs/rels/{rel}",
"templated": true }],
      "next": { "href": "/orders?page=2" },
      "ea:find": {
         "href": "/orders{?id}",
         "templated": true
      },
      "ea:admin": [{
         "href": "/admins/5",
         "title": "Kate"
      }]
   },
   "currentlyProcessing": 14,
   "shippedToday": 20,
   "_embedded": {
      "ea:order": [{
         "_links": {
            "self": { "href": "/orders/123" },
            "ea:basket": { "href": "/baskets/98712" },
            "ea:customer": { "href": "/customers/7809" }
         },
         "total": 30.00,
         "currency": "USD",
         "status": "shipped"
      }]
   }
}
```

Hypermedia Application Language (HAL)



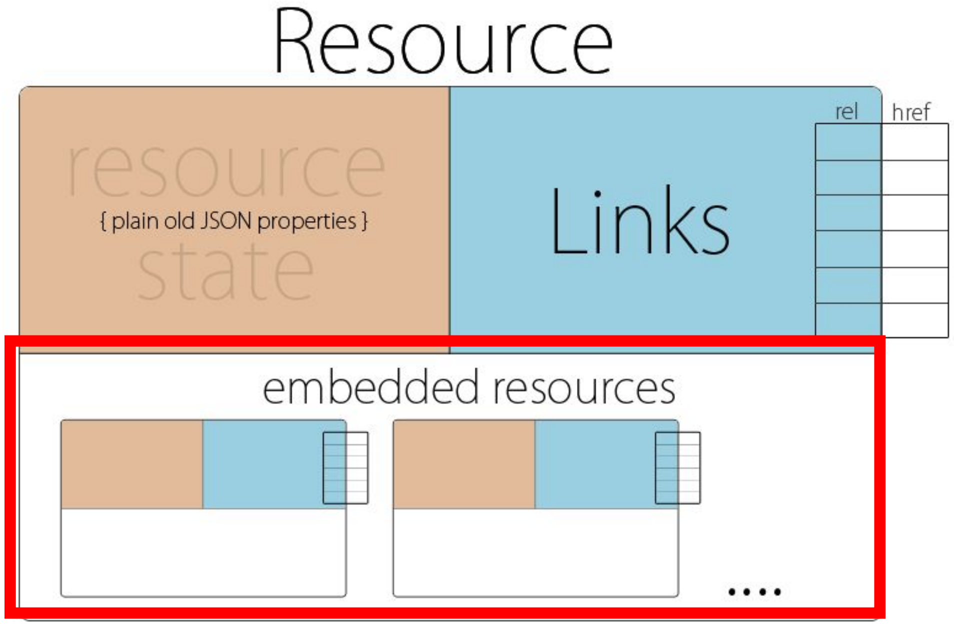
```
{
   "_links": {
      "self": { "href": "/orders" },
      "curies": [{ "name": "ea", "href": "http://example.com/docs/rels/{rel}",
"templated": true }],
      "next": { "href": "/orders?page=2" },
      "ea:find": {
         "href": "/orders{?id}",
         "templated": true
      },
      "ea:admin": [{
         "href": "/admins/5",
         "title": "Kate"
      }]
   },
   "currentlyProcessing": 14,
   "shippedToday": 20,
   "_embedded": {
      "ea:order": [{
         "_links": {
            "self": { "href": "/orders/123" },
            "ea:basket": { "href": "/baskets/98712" },
            "ea:customer": { "href": "/customers/7809" }
         },
         "total": 30.00,
         "currency": "USD",
         "status": "shipped"
      }]
   }
}
```

Hypermedia Application Language (HAL)



```
{
   "_links": {
      "self": { "href": "/orders" },
      "curies": [{ "name": "ea", "href": "http://example.com/docs/rels/{rel}",
"templated": true }],
      "next": { "href": "/orders?page=2" },
      "ea:find": {
         "href": "/orders{?id}",
         "templated": true
      },
      "ea:admin": [{
         "href": "/admins/5",
         "title": "Kate"
      }]
   },
   "currentlyProcessing": 14,
   "shippedToday": 20,
   "_embedded": {
      "ea:order": [{
         "_links": {
            "self": { "href": "/orders/123" },
            "ea:basket": { "href": "/baskets/98712" },
            "ea:customer": { "href": "/customers/7809" }
         },
         "total": 30.00,
         "currency": "USD",
         "status": "shipped"
      }]
   }
}
```

Hypermedia Application Language (HAL)



```
{
   "_links": {
      "self": { "href": "/orders" },
      "curies": [{ "name": "ea", "href": "http://example.com/docs/rels/{rel}",
"templated": true }],
      "next": { "href": "/orders?page=2" },
      "ea:find": {
         "href": "/orders{?id}",
         "templated": true
      },
      "ea:admin": [{
         "href": "/admins/5",
         "title": "Kate"
      }]
   },
   "currentlyProcessing": 14,
   "shippedToday": 20,
   "_embedded": {
      "ea:order": [{
         "_links": {
            "self": { "href": "/orders/123" },
            "ea:basket": { "href": "/baskets/98712" },
            "ea:customer": { "href": "/customers/7809" }
         },
         "total": 30.00,
         "currency": "USD",
         "status": "shipped"
      }]
   }
}
```

True REST API

1. Github
2. Facebook for Developers (Graph API)
3. Google Drive REST API
4. API Яндекс Диска
5. Paypal

References

1. Fielding, Roy Thomas. Architectural Styles and the Design of Networkbased Software Architectures. Doctoral dissertation, University of California, Irvine, 2000.
   https://www.ics.uci.edu/~fielding/pubs/dissertation/fielding_dissertation.pdf
2. REST APIs must be hypertext-driven
   https://roy.gbiv.com/untangled/2008/rest-apis-must-be-hypertext-driven
3. Richardson Maturity Model
   https://martinfowler.com/articles/richardsonMaturityModel.html
4. HAL - Hypertext Application Language
   http://stateless.co/hal_specification.html
5. Dylan Beattie — Real World REST and Hands-On Hypermedia
   https://youtu.be/kPrTMj-BK14

QUESTIONS?