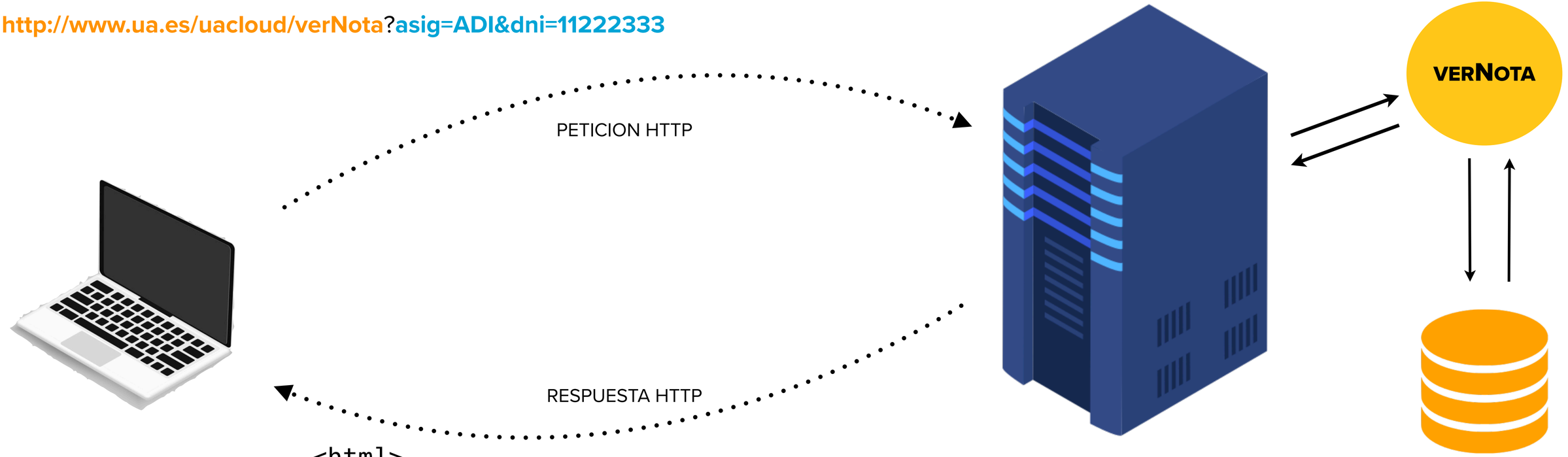
TEMA 1. INTRODUCCIÓN A LAS APIS WEB.
PARTE II: APIS RPC VS APIS REST

APP WEB "CLÁSICA"

PATH PARÁMETROS

http://www.ua.es/uacloud/verNota?asig=ADI&dni=11222333




```
<html>
<head>Tu nota de ADI</head>
<body>
  <p> Juan Ruiz: 10 </p>
</body>
</html>
```

APIs RPC

- Podemos ver la petición/respuesta en una app web clásica como una **llamada a un API remoto**

- El concepto central es la **operación** a realizar (`verNota`, `ponerNota`, `listarAlumnos`,...).

Por eso estos APIs se denominan (RPCs o Remote **Procedure** Call)

REST

REpresentational State Transfer

Es un estilo para el diseño de "sistemas hipermedia” como la web

Es un conjunto de reglas para diseñar cómo se interactúa con la aplicación, es decir, son **reglas generales para el diseño del API**

No es una tecnología, un lenguaje o una herramienta software

Roy T. Fielding
@fielding

En REST la idea central no es la operación sino el

RECURSO

RECURSO

Cualquier entidad/objeto
del modelo de dominio de
nuestra aplicación

Alumnos, profesores, asignaturas, notas,...

Productos, usuarios, pedidos,...

¿Pero entonces en REST no existen las operaciones? 🤔

Por supuesto, en el API se siguen haciendo **operaciones** sobre los **recursos,** pero en lugar de ser “ad-hoc” y propias del API (verNota, ponerNota, listarAlumnos) **son estándares y siempre las mismas (crear, leer, actualizar y borrar)**

PUNTOS IMPORTANTES EN UN API REST

- Las URLs (== los recursos)
- Los métodos HTTP (==las operaciones)
- El código de estado (==el resultado de la operación)
- El formato de los datos

1. LAS URLS

Cada recurso se representa con una URI distinta

Los APIS y URLs mostrados en esta transparencia son ficticios, cualquier parecido o semejanza con URLs reales o APIs pasados o presentes, es pura coincidencia

Todos los estudios de grado

`http://api.ua.es/estudios/grados`

El grado en Ingeniería Informática

`http://api.ua.es/estudios/grados/GII`

Todos los alumnos

`http://api.ua.es/alumnos`

El alumno con DNI 47890123

`http://api.ua.es/alumnos/47890123`

Cada recurso se representa con una URI distinta

Todos los estudios de grado

`http://api.ua.es/estudios/grados`

El grado en Ingeniería Informática

`http://api.ua.es/estudios/grados/GII`

Todos los alumnos

`http://api.ua.es/alumnos`

El alumno con DNI 47890123

`http://api.ua.es/alumnos/47890123`

Cada recurso se representa con una URI distinta

Todos los estudios de grado

`http://api.ua.es/estudios/grados`

El grado en Ingeniería Informática

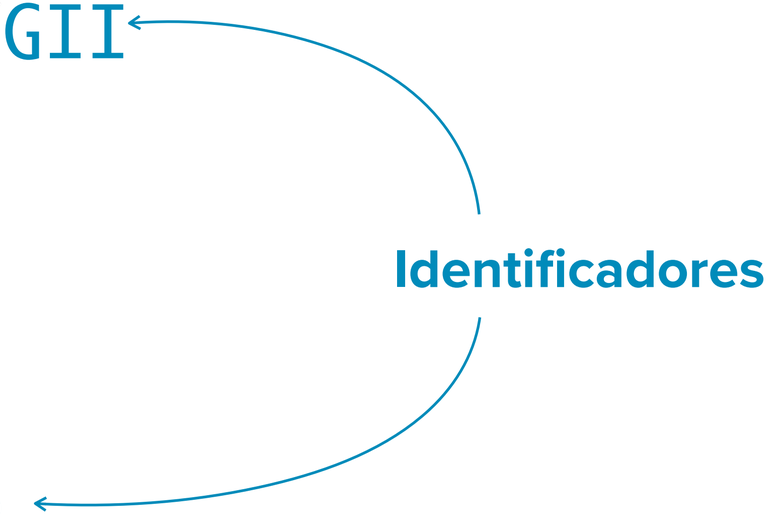
`http://api.ua.es/estudios/grados/GII`

Todos los alumnos del grado

`http://api.ua.es/alumnos`

El alumno con DNI 47890123

`http://api.ua.es/alumnos/47890123`



2. EL MÉTODO HTTP

CREATE / READ / UPDATE / DELETE

CRUD

POST GET PUT

CREATE / READ / UPDATE

DELETE

DELETE

CRUD

http://www.ua.es/uacloud/verNota?asig=ADI&dni=11222333

GET http://api.ua.es/asignaturas/ADI/notas/11222333

En la URI no aparecen verbos

El método HTTP es importante

PUT http://api.ua.es/asignaturas/ADI/notas/11222333

En la URI no aparecen verbos

El método HTTP es importante: representa la operación

3. EL CÓDIGO DE ESTADO

En REST el código de estado devuelto por el servidor es **importante**, ya que indica qué ha pasado con la operación

```
int main() {
   ...
   return 0;  //En web esto es 200 OK
}
```

https://httpstatuses.com/

4. EL FORMATO DE DATOS

HTML no es muy apropiado para **datos** en general, ya que está diseñado para representar **documentos**

```
<html>
<head>Tu nota de ADI</head>
<body>
  <h1>Tu nota:</h1>
  <p>Juan Ruiz:10</p>
</body>
</html>
```

XML

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<nota>
  <alumno>
     Juan Ruiz
  </alumno>
  <valor>
     10
  </valor>
</nota>
```

bueno...😑

JSON

```
{
  “alumno”: “Juan Ruiz”,
  “nota”: 10
}
```

APIS WEB RPC

- Las URIs son **VERBOS** y representan **OPERACIONES**
- El método HTTP (GET, POST,...)“no importa demasiado”, ni tampoco el código de estado
- El formato de datos no está estandarizado

APIS REST

- Las URIs son **NOMBRES** y representan **RECURSOS**
- El método HTTP (GET,POST, PUT, DELETE) define la **OPERACIÓN**
- El código de estado representa el **RESULTADO** de la operación
- El servidor devuelve **JSON**

Todo esto no son más que convenciones

Para diseñar un buen API para los servicios necesitamos usar algo que la gente conozca. Así que, **aunque no hay nada superior desde el punto de vista técnico en REST** y JSON con respecto a usar RPC con un protocolo de más bajo nivel, **usar algo que la gente comprenda bien [...] ayuda mucho en el diseño del API**

Jay Kreps, Lessons from Building and Scaling LinkedIn, QCon NY 2013