Sesión 03

Estilo Arquitectónico REST

Instructor:

ERICK ARÓSTEGUI

earostegui@galaxy.edu.pe

GALAXY
TRAINING

ÍNDICE

GALAXY
TRAINING

01

Introducción a REST

No obtiene una API RESTful lista para usar solo porque usa el Minimal API de ASP.NET Core

No obtiene una API RESTful lista para usar solo porque usa el Minimal API de ASP.NET Core

Lo obtienes adhiriéndote a un conjunto de restricciones RESTful

Introducción a REST

REST es ….

Roy Fielding
https://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm

Introducción a REST

REpresentational **S**tate **T**ransfer pretende evocar una imagen de cómo se comporta una aplicación web bien diseñada:

Una red de páginas web (una máquina de estado virtual)...
... donde el usuario progresa a través de una aplicación seleccionando enlaces (transiciones de estado)...
... dando como resultado que la página siguiente (que representa el siguiente estado de la aplicación) se transfiera al usuario y se represente para su uso.

Roy Fielding
https://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm

¿Qué es REST?

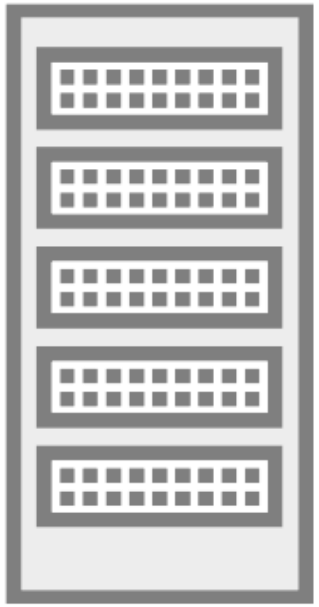
REST es un estilo arquitectónico

REST **NO** es un estándar por derecho propio

Los estándares se utilizan para implementar el estilo arquitectónico REST

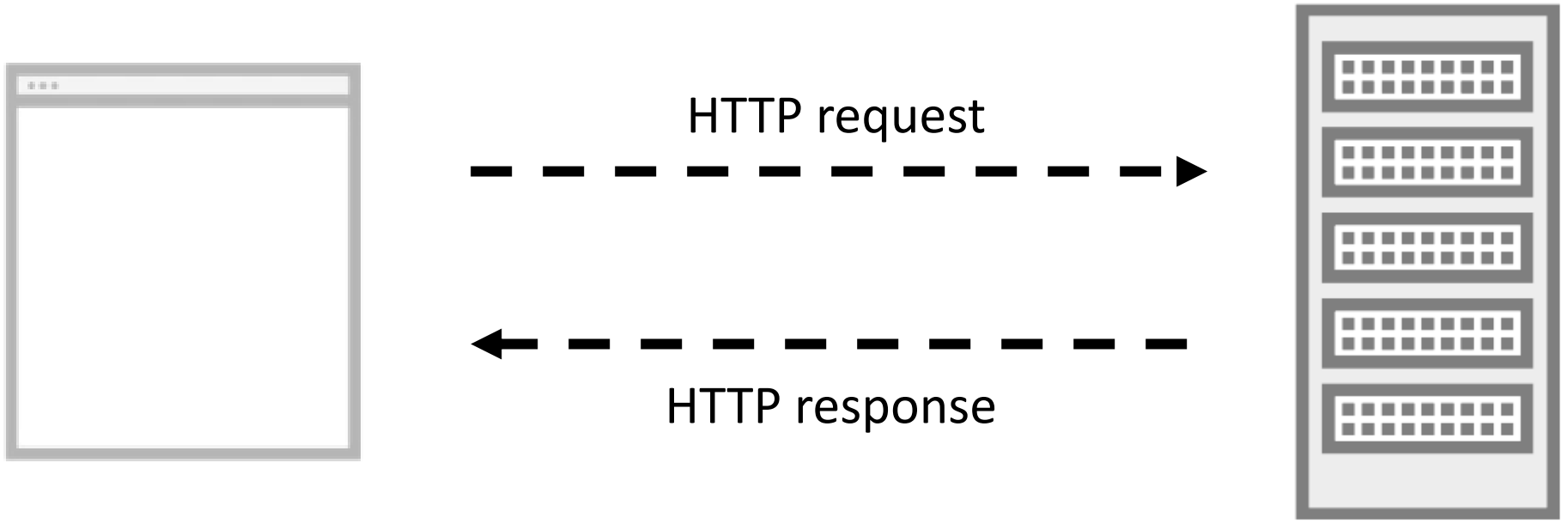
REST es, en principio, independiente del protocolo

Representational State Trasfer

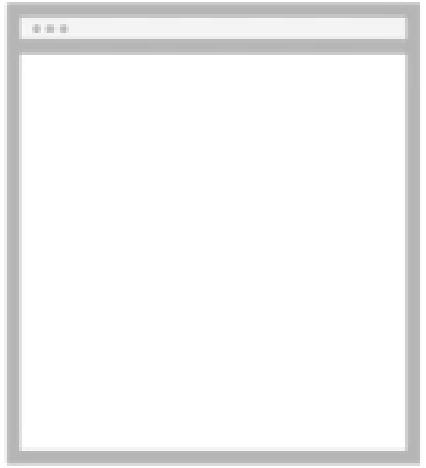


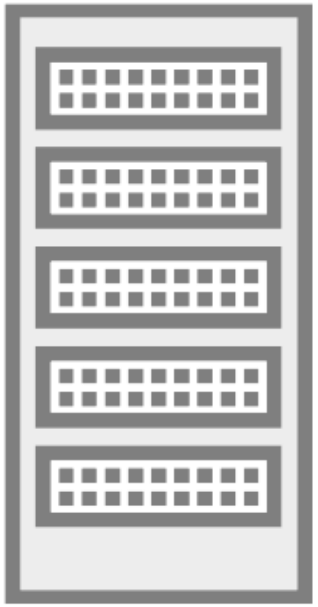
Representational State Trasfer

Representational State Trasfer



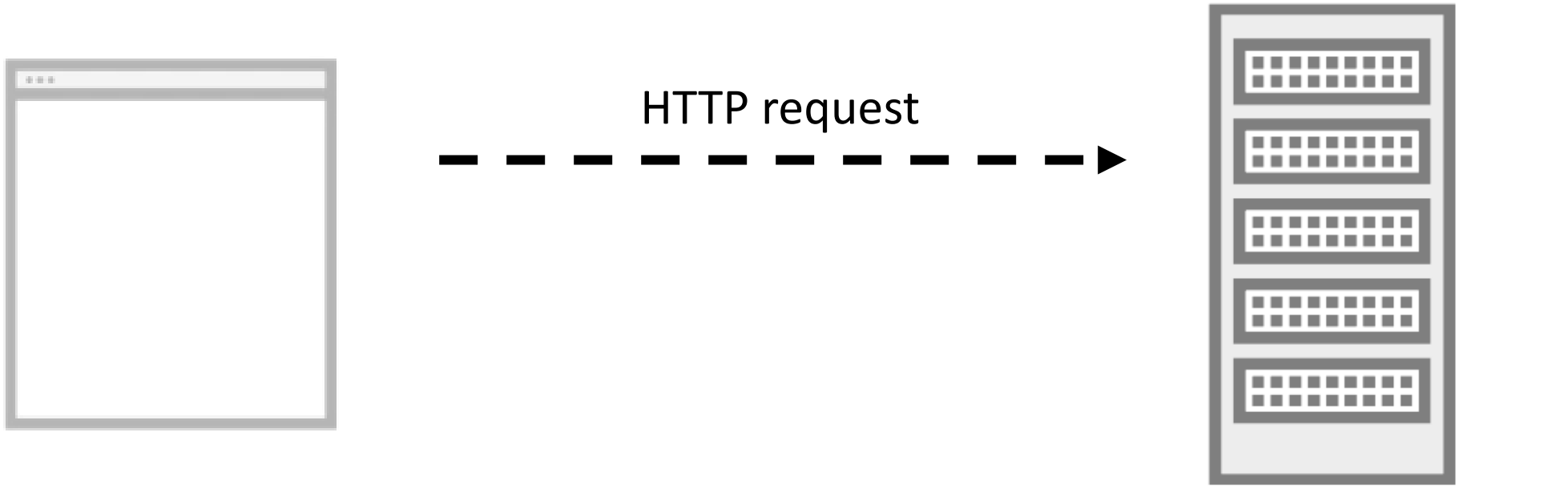
Representational State Trasfer



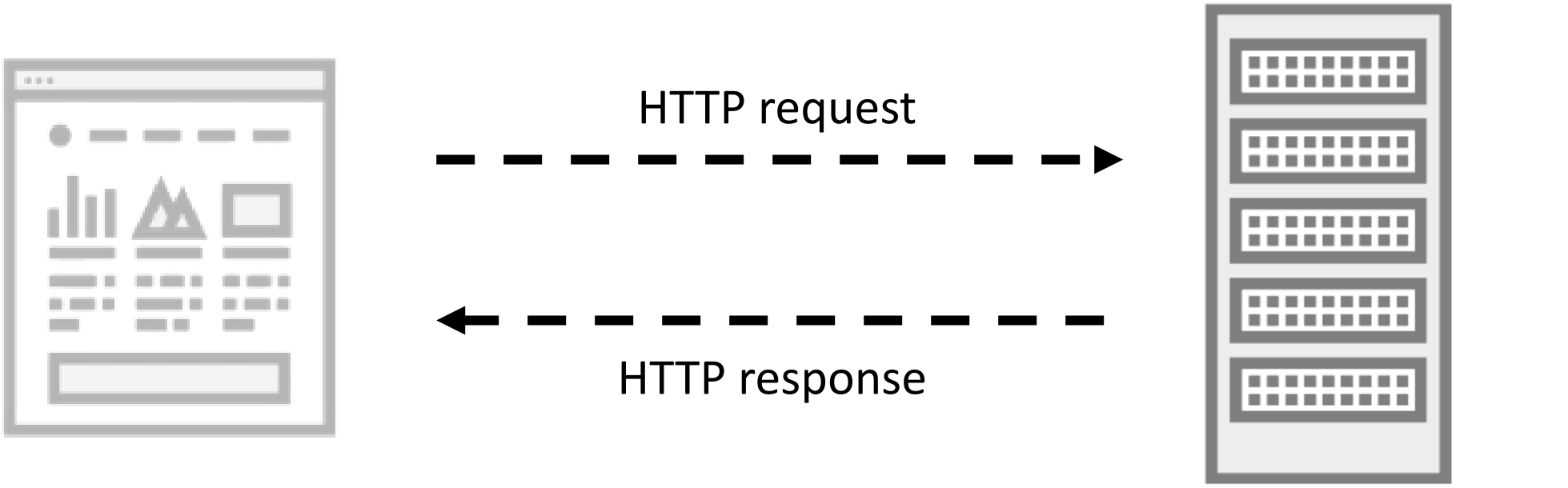


http://myappweb.com

Representational State Trasfer



Representational State Trasfer



REST necesita restricciones

REST es normado por 6 restricciones

- 5 restricciones obligatorias
- 1 restricción opcional

Una restricción es una decisión de diseño que puede tener un impacto positivo y negativo

REST necesita restricciones

Interfaz uniforme

Todos los recursos del servidor tienen un nombre en forma de URL o hipervínculo

REST necesita restricciones

Identificación de recursos

- Un recurso está separado conceptualmente de la representación.
- Media Types de representación:

  application/json, application/xml, custom, …

REST necesita restricciones

Manipulación de los recursos atreves de las representaciones

- La **representación + metadatos** deben ser suficientes para modificar o eliminar el recurso

REST necesita restricciones

Mensaje autodescriptivo

- Cada mensaje debe incluir información suficiente para describir cómo procesar el mensaje.

REST necesita restricciones

Hypermedia as the Engine of Application State (HATEOAS)

- Hypermedia es una generalización de Hypertext (links)
  - Indica como consumir y utilizar la API
  - Permite la auto-documentación de la API

REST necesita restricciones

Interfaz uniforme

Todos los recursos del servidor tienen un nombre en forma de URL o hipervínculo

REST necesita restricciones

Interfaz uniforme

Todos los recursos del servidor tienen un nombre en forma de URL o hipervínculo

Cliente-Servidor

El cliente y servidor están separados

(cliente y servidor pueden evolucionar por separado)

GALAXY TRAINING

REST necesita restricciones

Interfaz uniforme

Todos los recursos del servidor tienen un nombre en forma de URL o hipervínculo

Cliente-Servidor

El cliente y servidor están separados

(cliente y servidor pueden evolucionar por separado)

Statetlessness

El estado está contenido dentro de la solicitud

REST necesita restricciones

REST necesita restricciones

Sistema en capas

El cliente no puede determinar a que capa esta conectada

REST necesita restricciones

Sistema en capas

El cliente no puede determinar a que capa esta conectada

Cacheable

Cada mensaje de respuesta debe indicar explícitamente si se puede almacenar en caché o no.

REST necesita restricciones

Sistema en capas

El cliente no puede determinar a que capa esta conectada

Cacheable

Cada mensaje de respuesta debe indicar explícitamente si se puede almacenar en caché o no.

Código bajo demanda (opcional)

el servidor puede extender la funcionalidad del cliente

Un sistema solo se considera RESTful cuando se adhiere a todas las restricciones requeridas

La mayoría de las API "RESTful" **no son realmente RESTful** ...

... pero eso no los hace malos APIs, siempre y cuando se entienda las posibles compensaciones

02

Modelo de madurez de Richardson

GALAXY
TRAINING

Modelo de madurez de Richardson

```
POST (informacion)
http://host/myapi

POST (para crear un autor)
http://localhost/authors
```

Nivel 0 (Swamp of POX)

El protocolo HTTP se utiliza para la interacción remota.

... el resto del protocolo no se usa como debería ser.

Implementaciones de estilo RPC (SOAP, a menudo vistas cuando se usa WCF)

Modelo de madurez de Richardson

POST
http://localhost/api/authors

POST
http://localhost/api/authors/**{id}**

Nivel 1 (Recursos)

Cada recurso se asigna a un URI

Los métodos HTTP no se usan como deberían ser

Resultados en complejidad reducida

Modelo de madurez de Richardson

```
GET
http://host/api/authors
200 Ok (authors)


POST (representación de author)
http://localhost/api/authors
201 Created (author)
```

Nivel 2 (Verbos)

Se utilizan correctamente los verbos HTTP.

Se utilizan correctamente los códigos de estado.

Remueve las variaciones innecesarias.

Modelo de madurez de Richardson

```
GET
http://host/api/authors
200 Ok (authors + links que
controlan el estado de la
aplicación)
```

Nivel 3 (Hypermedia)

La API tiene soporte de Hypermedia as the Engine of Application State (HATEOAS)

Autodocumentacion

03

Diseño de un servicio REST

Diseño del contrato

Identificador de recursos
http://localhost/api/authors

Método HTTP
https://datatracker.ietf.org/doc/html/rfc9110

Payload
(Representación: media types)

Diseño del contrato

Identificador de recursos
http://localhost/api/authors

Método HTTP
https://datatracker.ietf.org/doc/html/rfc9110

Payload
(Representación: media types)

Lineamientos para el nombramiento de recursos

Sustantivos: cosas, no acciones.

- ~~`api/getauthors`~~
- **`GET`** `api/authors`
- **`GET`** `api/authors/`**`{authorId}`**

Transmitir significado al elegir sustantivos

Lineamientos para el nombramiento de recursos

Seguir este principio para la predictibilidad

- ~~api/something/somethingelse/authors~~
- api/authors
- ~~api/id/authors~~
- api/authors/**{authorId}**

Lineamientos para el nombramiento de recursos

Representar la jerarquía al nombrar recursos

- `api/authors/{authorId}/books`
- `api/authors/{authorId}/books/{bookId}`

Lineamientos para el nombramiento de recursos

Filters, sorting orders, … no son recurso

- ~~api/authors/orderby/name~~
- api/authors**?orderby=name**

Lineamientos para el nombramiento de recursos

A veces, las llamadas al estilo RPC no se asignan fácilmente a nombres de recursos pluralizados

- ~~api/authors/**{authorId}/pagetotals**~~
- ~~api/**authorpagetotals/{id}**~~
- api/authors/**{authorId}/totalamountofpages**

Diseño del contrato

Identificador de recursos
http://localhost/api/authors

Método HTTP
https://datatracker.ietf.org/doc/html/rfc9110

Payload
(Representación: media types)

GALAXY TRAINING

Interactuar con recursos a través de métodos HTTP

Método HTTP	Contenido del Request	Ejemplo de URI	Contenido de la respuesta
GET	-	/api/authors /api/authors/**{authorId}**	Colección de authors Un único author
POST	Un único author	/api/authors	Un único author
PUT	Un único author	/api/authors/**{authorId}**	Un único autor o vacío
PATCH	JsonPatchDocument del autor	/api/authors/**{authorId}**	Un único autor o vacío
DELETE	-	/api/authors/**{authorId}**	-
HEAD	-	/api/authors /api/authors/**{authorId}**	-
OPTIONS	-	/api/..	-

La importancia de los códigos de estado

Los códigos de estado indican al consumidor :

- Si la solicitud funcionó o no como se esperaba
- ¿Quién es responsable de una solicitud fallida?

La importancia de los códigos de estado

Ser lo más específico posible

- Los consumidores de API suelen ser no humanos

Ser especialmente específico en lo que respecta a informar quién/qué es responsable de un error

GALAXY TRAINING

La importancia de los códigos de estado

Nivel 100 Informativo	Nivel 200 Éxito	Nivel 300 Informativo
En su mayoría no se usan	200 – Ok 201 – Created 204 – No content	En su mayoría no se usan

La importancia de los códigos de estado

400 – Bad request 401 – Unauthorized 403 – Forbidden 404 – Not found	405 – Method not allowed 406 – Not acceptable 409 - Conflict 415 – Unsupported media type 422 – Unprocessable entity	En su mayoría no se usan
Nivel 400 Errores del cliente	**Nivel 400 Errores del cliente**	**Nivel 500 Errores del servidor**

Diseño del contrato

Identificador de recursos
http://localhost/api/authors

Método HTTP
https://datatracker.ietf.org/doc/html/rfc9110

Payload
(Representación: media types)

Formateadores y Negociación de Contenido

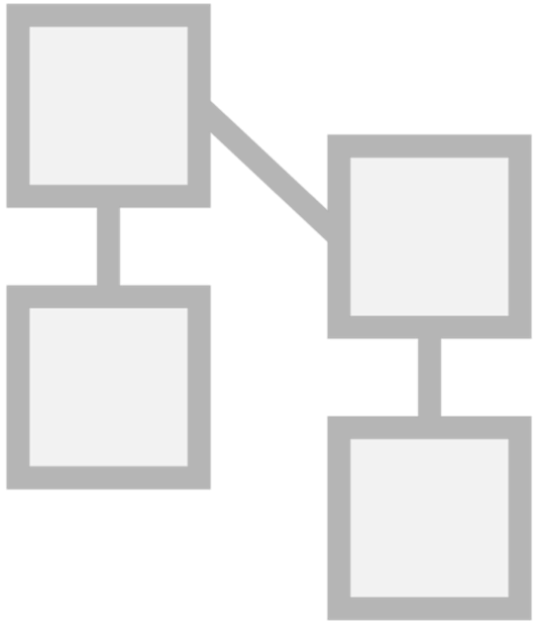
Output formatter

Se ocupa del tipo de salida
Media type: Accept header

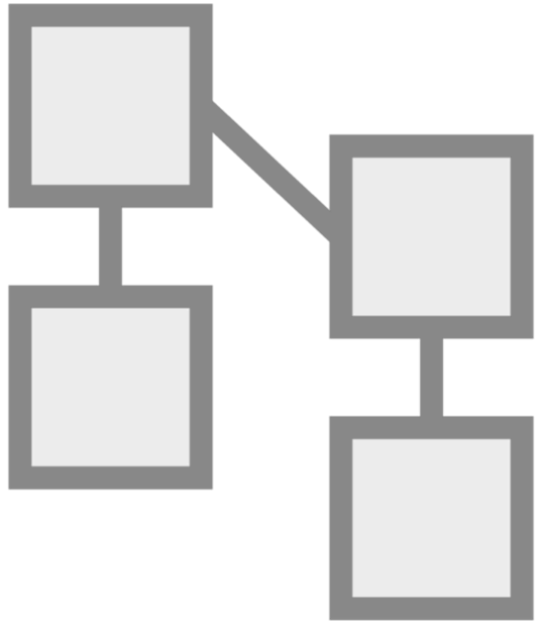
Input formatter

Se ocupa del tipo de entrada
Media type: Content-type header

Modelo externo frente a modelo de entidad



El modelo de entidad representa las filas de la base de datos como objetos



El modelo exterior representa lo que se envía a través del servicio

Modelo externo frente a modelo de entidad

Modelo orientado al exterior (AuthorDto)

```
Guid Id
string FirstName
string LastName
int Age
```

Modelo de entidad (Autor)

```
Guid Id
string FirstName
string LastName
DateTimeOffset DateOfBirth
```

Modelo externo frente a modelo de entidad

Modelo orientado al exterior (AuthorDto)

```
Guid Id
string Name
int Age
```

Modelo de entidad (Autor)

```
Guid Id
string FirstName
string LastName
DateTimeOffset DateOfBirth
```

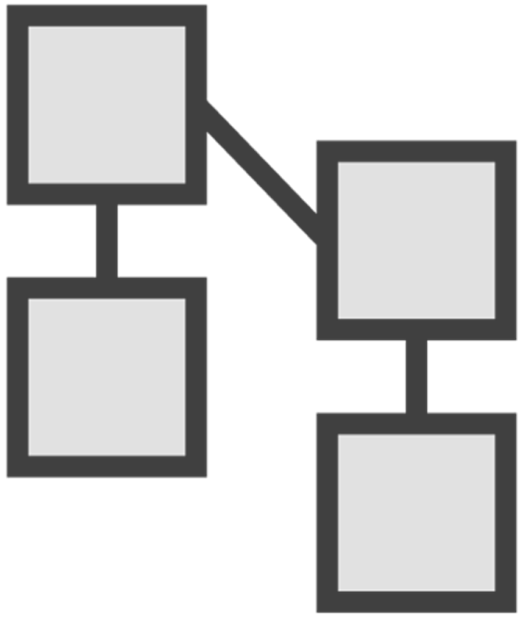
La separación de los modelos externos y de entidad conduce a un código más robusto, confiable y evolutivo

04

Implementado un servicio REST

GALAXY
TRAINING

Enlace de modelo con atributos de origen



[FromBody]

- Cuerpo de solicitud

[FromForm]

- Datos del formulario en el cuerpo de la solicitud

[FromHeader]

- Encabezado de solicitud

[FromQuery]

- Parametros por Query string

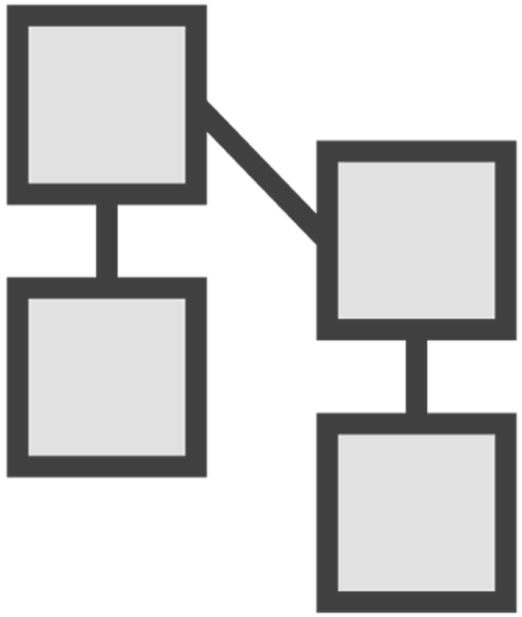
[FromRoute]

- Datos a traves de rutas de la solicitud actual

[FromService]

- El servicio inyectado como parámetro de acción

PUT Vs PATCH



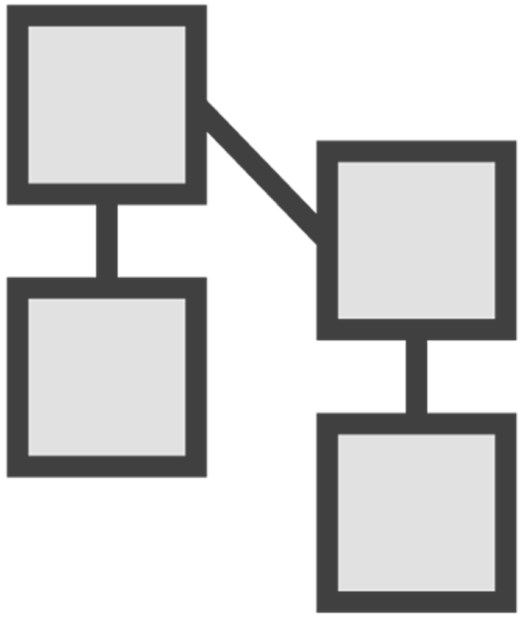
PUT es para actualizaciones completas

- Todos los campos de recursos se sobrescriben o se establecen en sus valores predeterminados

PATCH es para actualizaciones parciales

- Permite enviar conjuntos de cambios a través de
- JsonPatchDocument

PUT Vs PATCH



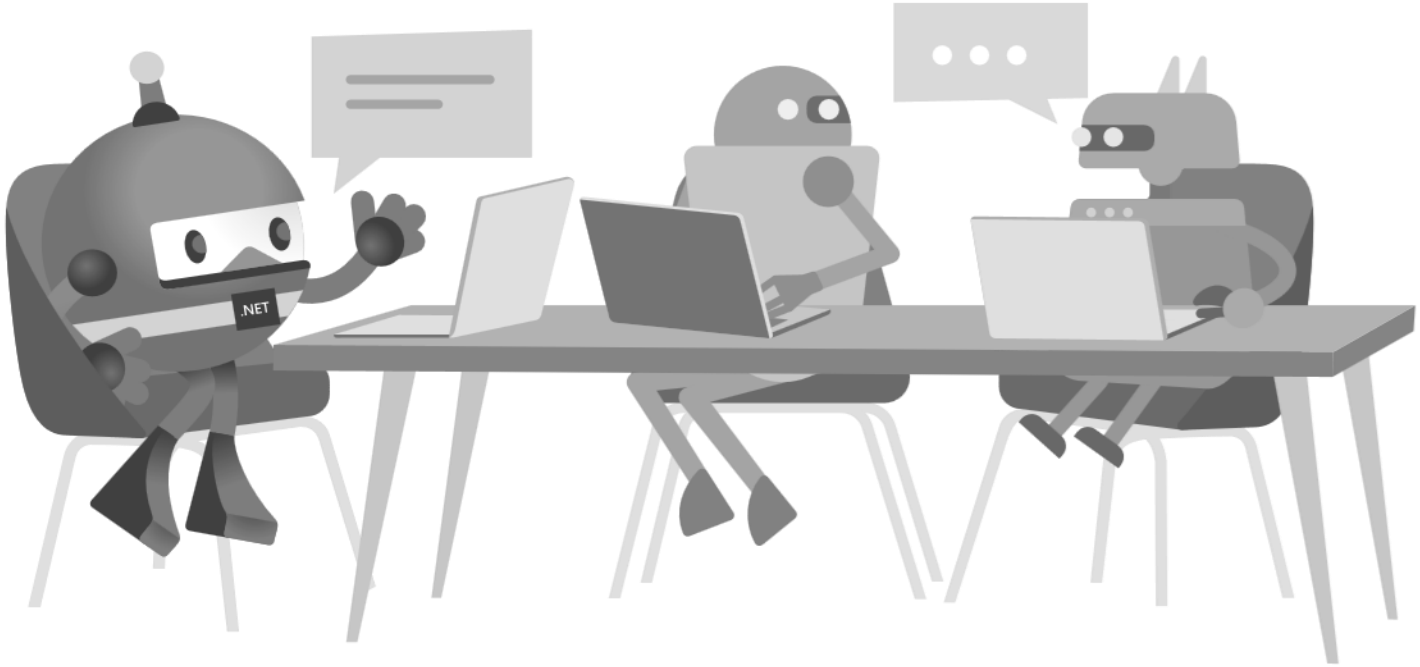
HTTP PATCH is for partial updates

- El cuerpo de la solicitud de un PATCH se describe en RFC 6902 (JSON patch) https://tools.ietf.org/html/rfc6902

Las solicitudes PATCH deben enviarse con el media type "**application/json-patch+json**"

Implementado un servicio REST

DEMO



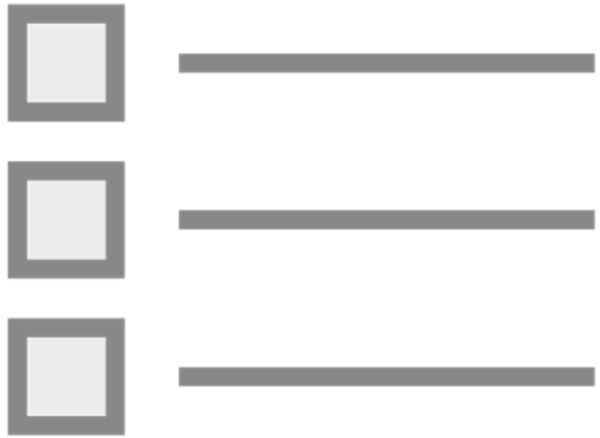
GALAXY
TRAINING

05

Validación de parámetros de entrada

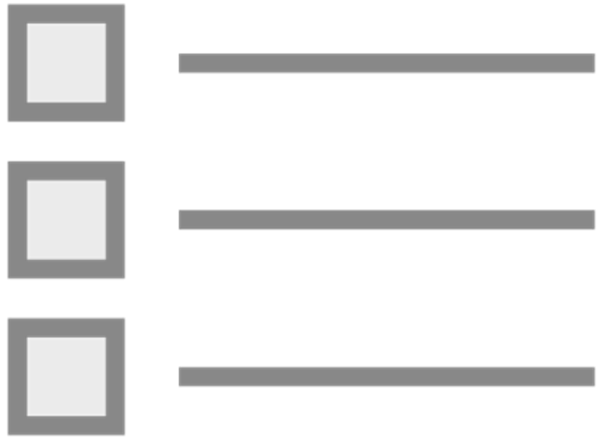
GALAXY
TRAINING

Diseño del contrato



Definición de reglas de validación

Diseño del contrato

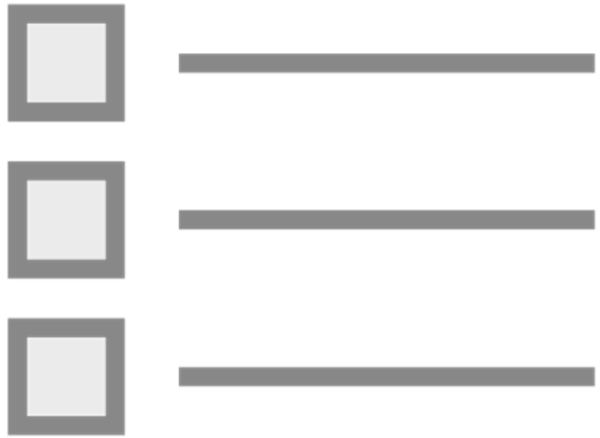


Definición de reglas de validación

Comprobación de reglas de validación

Validación de parámetros de entrada

Diseño del contrato



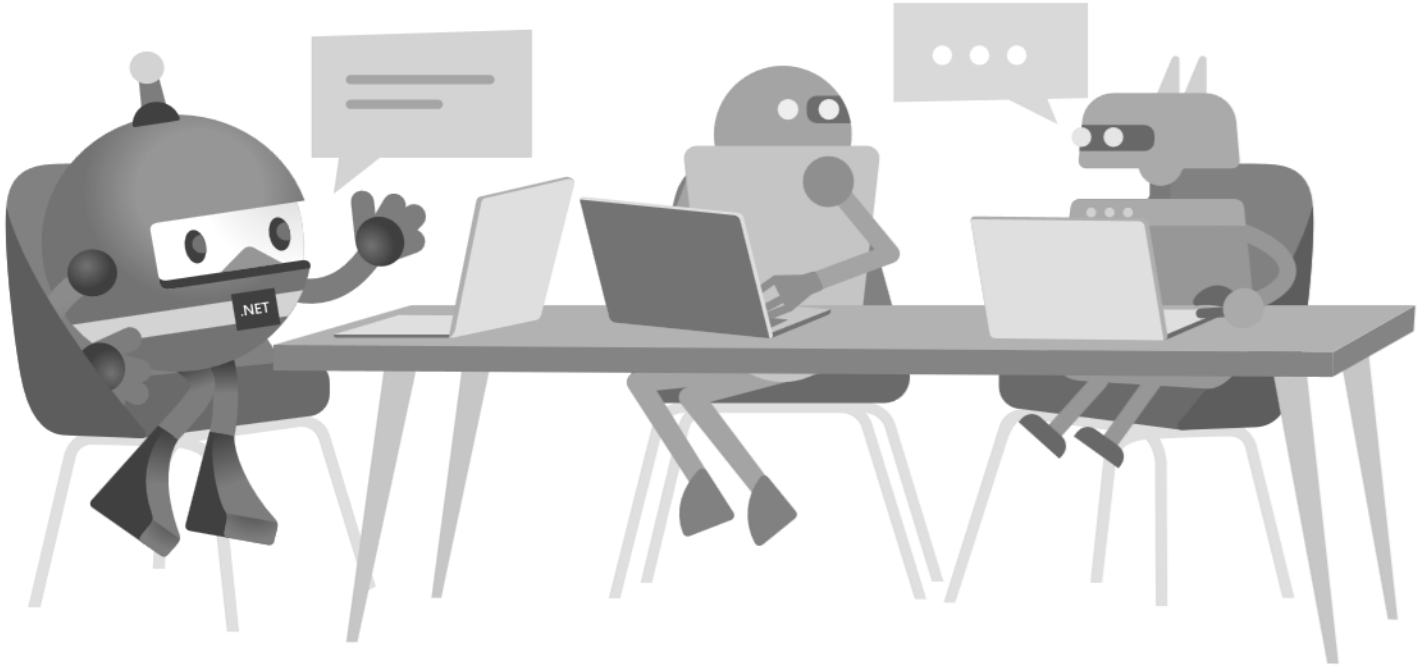
Definición de reglas de validación

Comprobación de reglas de validación

Notificación de errores de validación

Validación de parámetros de entrada

DEMO



GALAXY
TRAINING

GRACIAS

POR SU PREFERENCIA

GALAXY
TRAINING