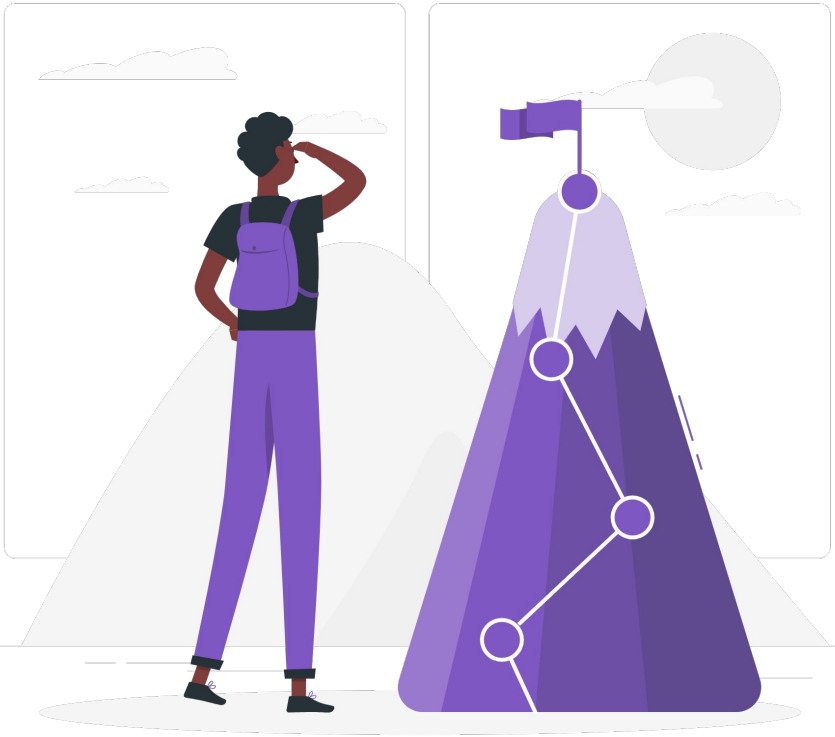

# BACKEND
# RESTFul API

DEV.F:
DESARROLLAMOS(PERSONAS);

Elaborado por: César Guerra
www.cesarguerra.mx



# Objetivos de la Sesión

- Aprender acerca de los protocolos relevantes con el que se logran comunicar las computadoras.
- Aprender acerca de los verbos HTTP y su uso.
- Comprender el concepto de REST y API y el papel que desempeñan en backends modernos.
- Entender qué son los códigos HTTP y para qué nos sirven.
- Aprender y aplicar las mejores prácticas para crear API REST.

DEV.F::

# Conceptos

- **TCP/IP** : Transfer control protocol/Internet Protocol

- **API** : Application Programming Interface es un conjunto de reglas que determinan cómo las aplicaciones o los dispositivos pueden conectarse y comunicarse entre sí. Es un mecanismo que permite a una aplicación o servicio acceder a un recurso dentro de otra aplicación o servicio.

- **REST**: Representational State Transfer es una interfaz para conectar varios sistemas basados en el protocolo HTTP (uno de los protocolos más antiguos) y nos sirve para obtener y generar datos y operaciones, devolviendo esos datos en formatos muy específicos.

- **API REST:** Una API REST es una API que se ajusta a los principios de diseño de REST. Las API REST son a veces denominadas API RESTful.

# Protocolos

Un protocolo es sistema de normas que regulan la comunicación entre dos o más sistemas que se transmiten información a través de diversos medios físicos

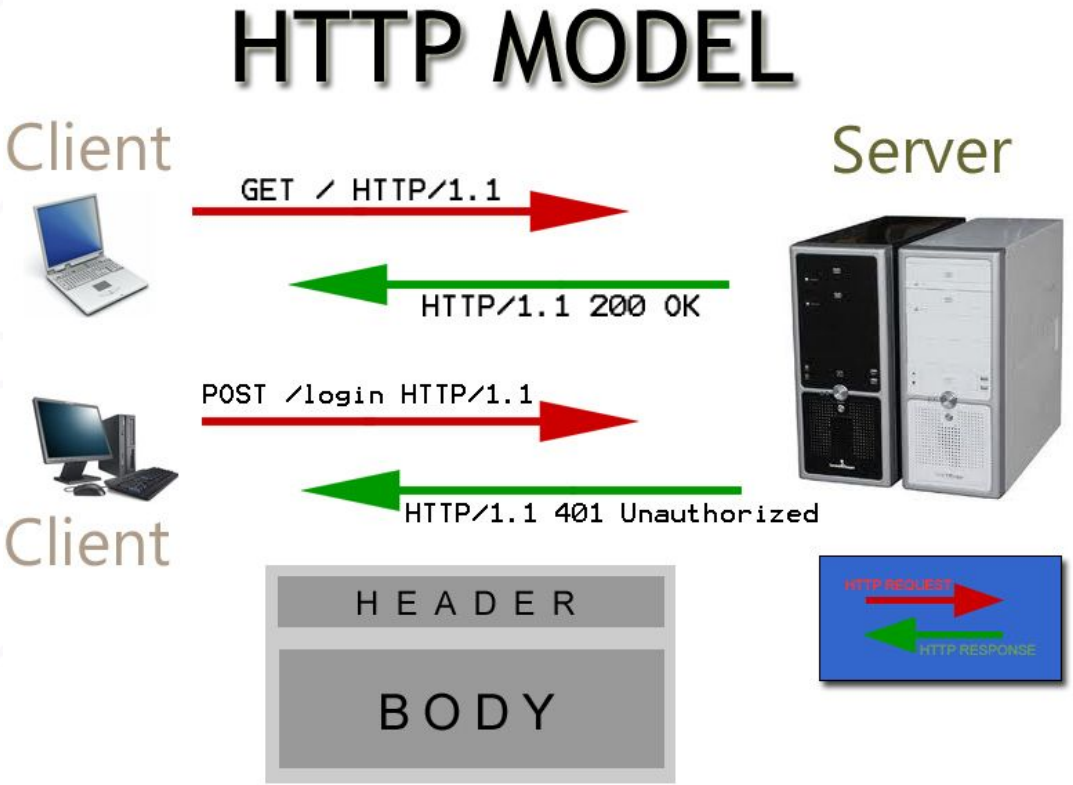
- **HTTP:** Se encarga de la comunicación entre un servidor web y un navegador web. HTTP se utiliza para enviar las peticiones de un cliente web (navegador) a un servidor web, volviendo contenido web (páginas web) desde el servidor al cliente.

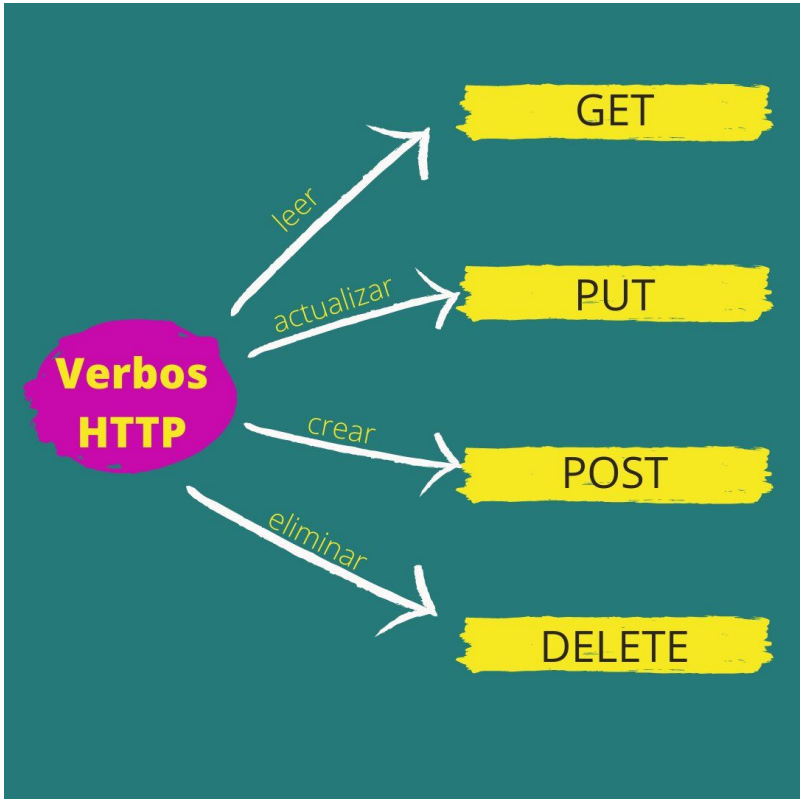
- **HTTPS:** Lo mismo pero más seguro. La información viaja de manera segura y encriptada mediante un certificado SSL/TLS

- **FTP**: File transfer protocol, como su nombre lo indica es un protocolo para transferencia de archivos.

- **SMTP**: Simple Mail Transfer Protocol
- **IMAP** - Internet Message Access Protocol
- **POP** - Post Office Protocol
- **SSL** - Secure Sockets Layer
- **TLS** - Transport Layer Security

# Modelo HTTP


HTTP MODEL
Client
Server
GET / HTTP/1.1
HTTP/1.1 200 OK
POST /login HTTP/1.1
HTTP/1.1 401 Unauthorized
Client
HEADER
BODY
HTTP REQUEST
HTTP RESPONSE




Referencia de Imagen:
https://twitter.com/wwcodeguatemala/status/1527826609260183554

# Verbos HTTP

**GET**: Verbo exclusivo para obtener recursos del servidor

**POST**: Verbo exclusivo para crear nuevos recursos en el servidor

**PUT**: Verbo reemplaza un recurso por completo en el servidor

**PATCH**: Verbo que modifica parcialmente el el recurso.

**DELETE**: Verbo que elimina física o lógicamente el recurso

DEV.F::

# ¿Qué es REST ?

**REST is acronym for REpresentational State Transfer. It is architectural style for distributed hypermedia systems**

# Principios de REST

**Existen seis principios de diseño de REST, también conocidos como restricciones de arquitectura:**

1. **Client–server -** En el diseño de API REST, las aplicaciones de cliente y de servidor deben ser completamente independientes entre sí. La única información que la aplicación de cliente debe conocer es el URI del recurso solicitado.
2. **Stateless** – Cada solicitud debe incluir toda la información necesaria para procesarla. Las aplicaciones de servidor no pueden almacenar ningún dato relacionado con una solicitud de cliente.
3. **Cacheable** – La respuesta debería poder almacenarse en la memoria caché en caso de que la respuesta no haya cambiado con el tiempo.El objetivo es mejorar el rendimiento en el lado del cliente, al mismo tiempo que aumenta la escalabilidad en el lado del servidor.

# Principios de REST

**Existen seis principios de diseño de REST, también conocidos como restricciones de arquitectura:**

**4. Uniform interface** – Todas las solicitudes de API para el mismo recurso deben ser iguales, independientemente de la procedencia de la solicitud. Deben tener una interfaz común para cualquier tipo de información (JSON)

**5. Layered system** – En las API REST, las llamadas y respuestas pasan por diferentes capas. Deben diseñarse para que ni el cliente ni el servidor puedan notar si se comunican con la aplicación final o con un intermediario.

**6. Code on demand (optional)** – Generalmente, las API REST envían recursos estáticos, pero en algunos casos, las respuestas también pueden contener un código ejecutable. En estos casos, el código solo debería ejecutarse bajo demanda.

# REST - Independiente del formato

No se hace:

/contacto/tarea.pdf

Si se hace

/contactos/tareas

# REST - Mantienen jerarquía lógica

No se hace:

/tarea/4/contactos/2

Si se hace:

/contactos/2/tareas/4

# REST - Filtrado y otras operaciones

No se hace:

/tareas/fecha-desde/2007/pagina/3

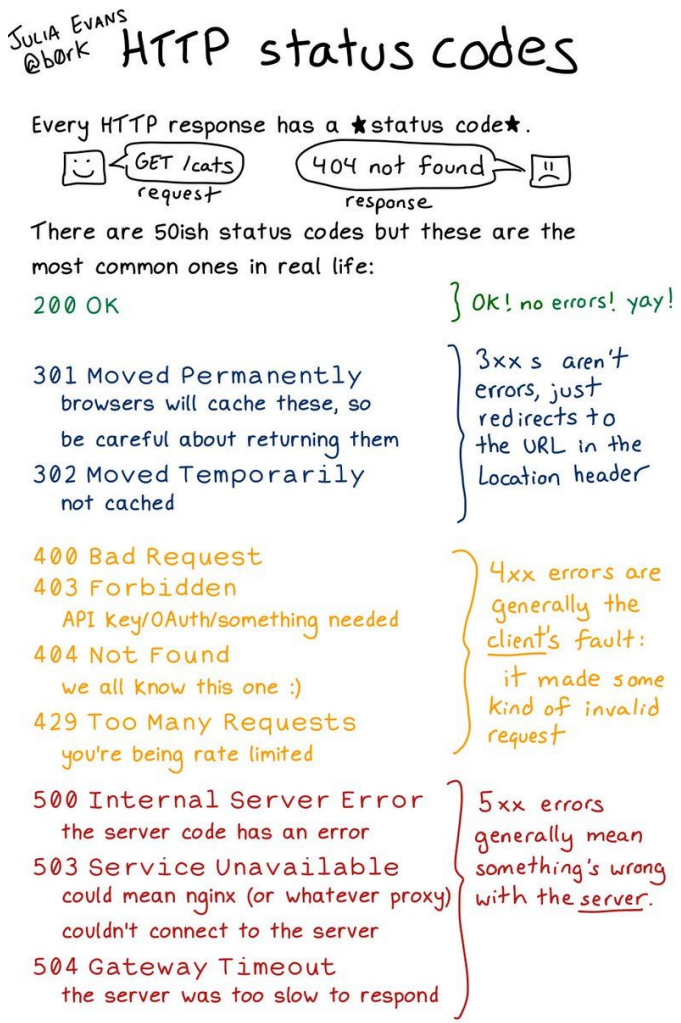
Si se hace:

/tareas?fecha-desde=2007&pagina=3

# REST - CRUD

## CREATE - READ - UPDATE - DELETE

| API Name | HTTP Method | Path | Status Code | Description |
|---|---|---|---|---|
| GET Employees | GET | `/api/v1/employees` | 200 (OK) | All Employee resources are fetched. |
| POST Employee | POST | `/api/v1/employees` | 201 (Created) | A new Employee resource is created. |
| GET Employee | GET | `/api/v1/employees/{id}` | 200 (OK) | One Employee resource is fetched. |
| PUT Employee | PUT | `/api/v1/employees/{id}` | 200 (OK) | Employee resource is updated. |
| DELETE Employee | DELETE | `/api/v1/employees/{id}` | 204 (No Content) | Employee resource is deleted. |



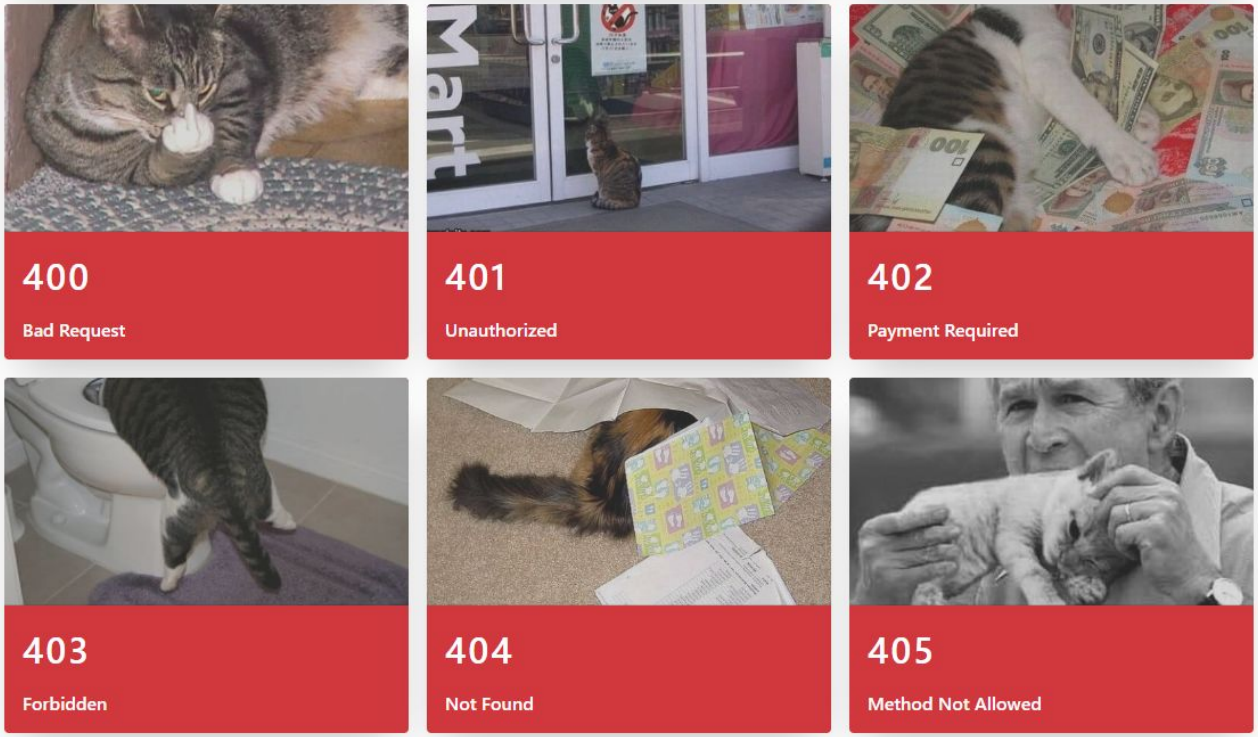

https://twitter.com/b0rk/status/1145824140462608387

# REST - STATUS CODES

Códigos más comunes.

DEV.F::

# Http Cats




https://http.cat/

# REST - Best Practices

- REST siempre debe enviar y aceptar **JSON** como **Content-Type**
- Usa sustantivos en la rutas en vez de verbos
  - Ejemplo
    - Si ***/employees/2323423***
    - No ***/getOneEmployee***
- Usa sustantivos en **plural** en vez de **singular**
  - Ejemplo
    - Si ***/cars***
    - No ***/car***

# REST - Best Practices

- Utiliza sub-resources para relaciones
  - Ejemplo
    - ***/employees/711/addresses*** #Obtinen todas las direcciones del empleado 711
    - ***/employees/711/addresses/2*** #Obtinen la dirección 2 del empleado 711

- Siempre utiliza el Header ***Content-Type*** para especificar el tipo de contenido del request y response

# REST - Best Practices

- Provee Filter, Sorting, Field Selection y Pagination para las colecciones
- Versiona tu API
- Usa Status Codes de HTTP para capturar errores
- Usa mensajes de error descriptivos


# REST - Best Practices

Usa **HATEOAS**, conocidas como ***Hypermedia as the Engine of Application State***, hace mas facil la navegación de la api.

**HATEOAS** es un término inglés que traducido significa “Estoy desesperado buscando un acrónimo con gancho y no lo consigo encontrar”. Pero detrás de esta palabra encontrarás lo que durante veinte años ha sido el core de la web: el hipertexto, los enlaces.

DEV.F::


*Solo es complicado si tú lo complicas. O bueno, si es complicado de por sí.*

*¡ Tengan en cuenta las buenas prácticas !*

DEV.F::


¡ GRACIAS !
CésarGuerra
www.cesarguerra.mx