REST - BACKEND

DEV.F:
DESARROLLAMOS(PERSONAS);

Conceptos

- **TCP/IP** : Transfer control protocol/Internet Protocol
- **REST**: Representational State Transfer
- **API** : Application Programming Interface
- **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

Otros Protocolos

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

SMTP: Simple Mail Transfer Protocol

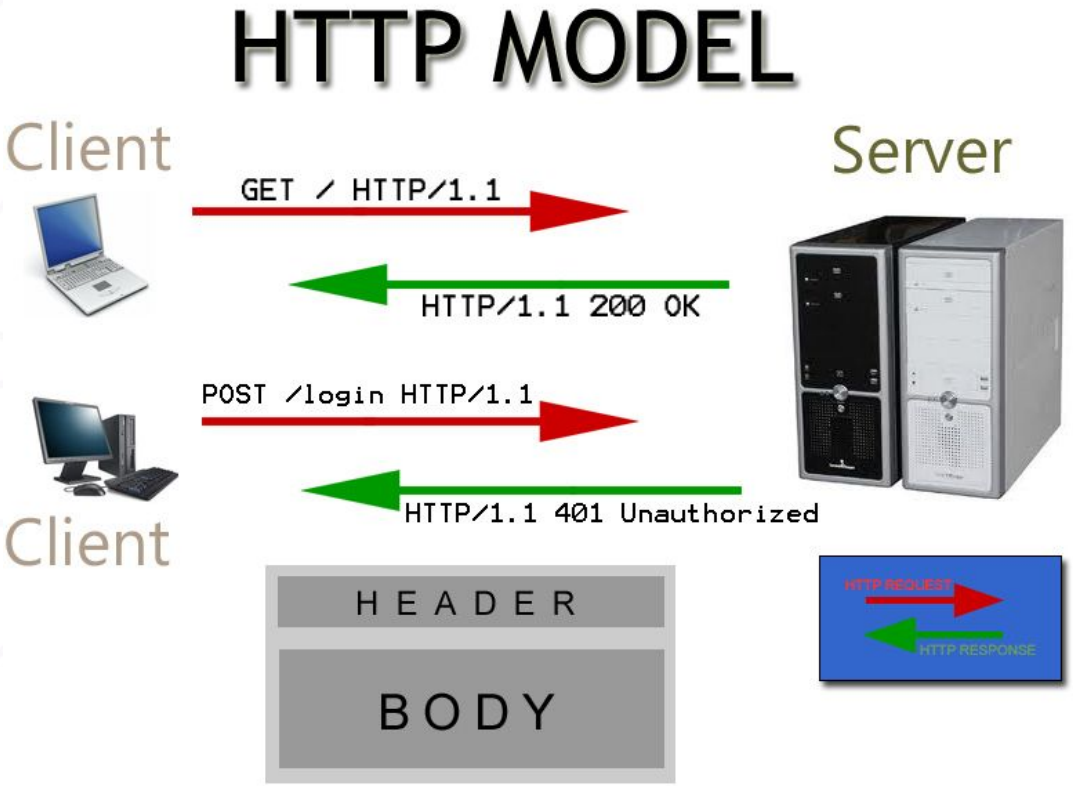
IMAP - Internet Message Access Protocol

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

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

¿Qué es REST ?

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

Principios de REST

1. **Client–server -** The architecture is divided into server-client.
2. **Stateless** – The activity is stateless since it does not retain any type of information.
3. **Cacheable** – The response should be cacheable in case the response has not changed over time.
4. **Uniform interface** – It must have a uniform interface for any kind of information.(JSON)
5. **Layered system** – It must follow a layered type of system in which the client only interacts with the point of contact.(MVC)
6. **Code on demand (optional)** – can return “executable code”

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.

REST - STATUS CODES

HTTP status ranges in a nutshell:

1xx: hold on

2xx: here you go

3xx: go away

4xx: you fucked up

5xx: I fucked up

-via @abt_programming

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 Filtrado,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.
  - Ejemplo

```
{
  "id": 711,
  "manufacturer": "bmw",
  "model": "X5",
  "seats": 5,
  "drivers": [
    {
     "id": "23",
     "name": "Stefan Jauker",
     "links": [
      {
       "rel": "self",
       "href": "/api/v1/drivers/23"
      }
     ]
    }
  ]
}
```