José Montoya Guzmán

- Tech Lead Front End en BBVA.
- Git Evangelist.
- Experiencia en Java y SQL.
- Sensei por hobby.

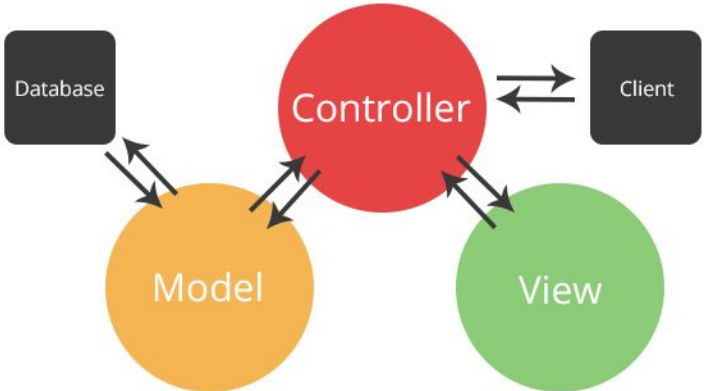
https://github.com/montoyaguzman

KATA
JavaScript Avanzado

DEV.F::
DESARROLLAMOS(PERSONAS);

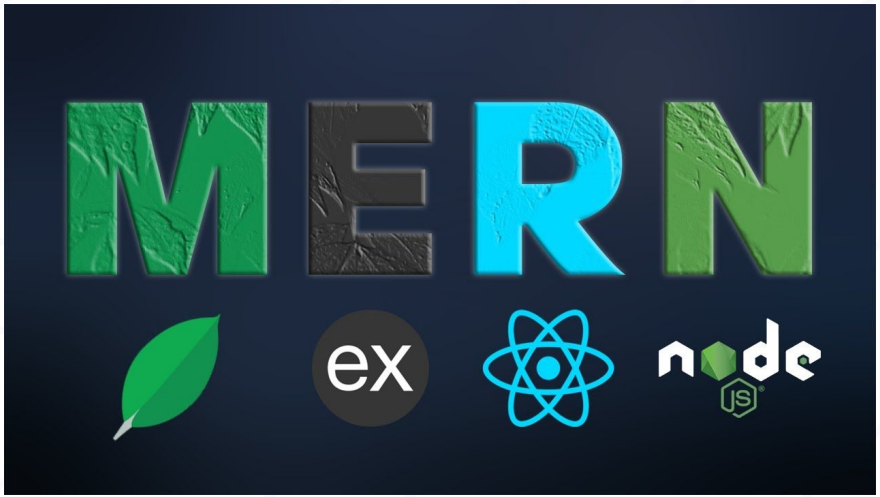
¿Qué veremos en está kata?

- Node.
- Npm y paquetes.
- Node para front y back end.
- Arquitectura de software.
- Stacks de desarrollo web.
- Comparativa entre Node vs JavaScript.
- Event loop.
- Asincronía.
- API Rest.
- ExpressJS (CRUD).
- Deploys.



Semana 1

- Node.
- Npm y paquetes.
- Node para front y back end.
- Arquitectura de software.
- Introducción a los stacks de desarrollo web.

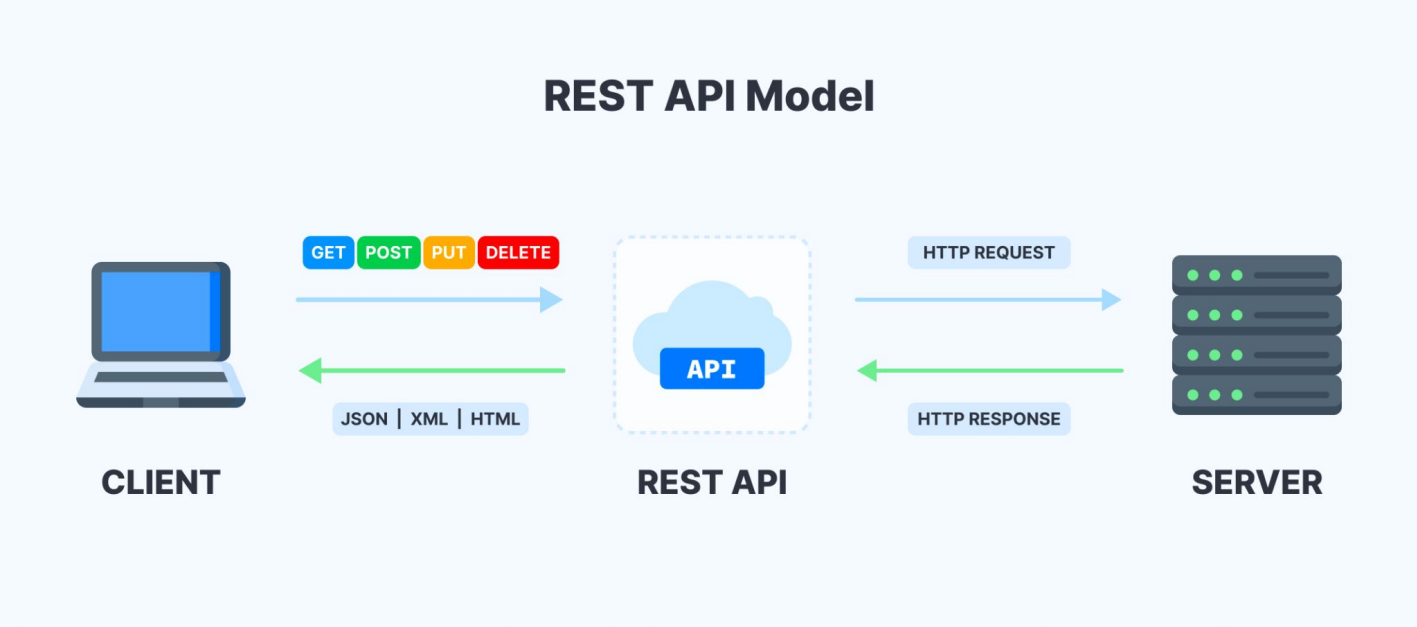


Semana 2

- Comparativa entre Node y JavaScript.
- Event loop.
- Asincronía.

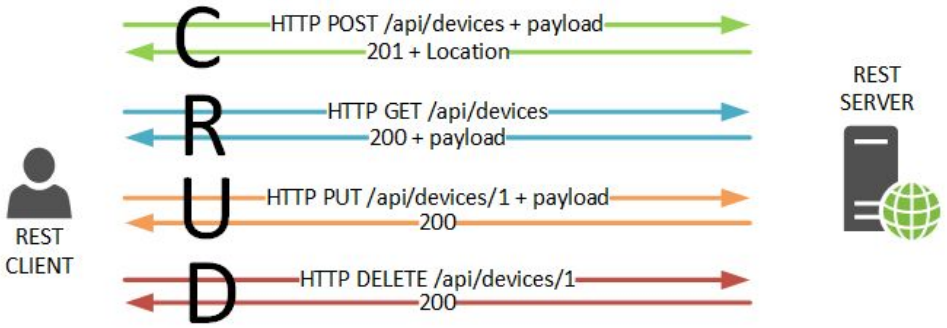
Semana 3

- API Rest.
- ExpressJS (CRUD).
- Deploys.



Semana 4

- Construir una API Rest que se conecte con un fake back end, y realice las 4 operaciones básicas de un CRUD.



Node y NPM

DEV.F::
DESARROLLAMOS(PERSONAS);

Node

- Mayo 2009 (Ryan Lenhiart).
- Entorno en tiempo de ejecución **multiplataforma** para la **capa del servidor (no se limita a ello).**
- Basado en el motor V8 de google.
- Escrito en C++.
- Basado en módulos.
- Es asíncrono y trabaja con base en un bucle de eventos.

¿Qué puedo hacer con Node?

- Desarrollar back end.
- Generar páginas dinámicas en un servidor web.
- Acceder a bases de datos relacionales y no relacionales.
- Crear, leer y escribir archivos.
- Recuperar datos de formularios HTML.
- Procesar y almacenar archivos enviados desde una página web.
- Generar y enviar a aplicaciones cliente (navegador web, aplicación móvil etc.) archivos JSON (API).
- Interactuar con sitios de una sola página (SPA) creados en Angular, React, Vue etc. => **server side render**

Práctica 1

- Instalación de herramientas y validación en la CLI.
- Uso de la documentación oficial.

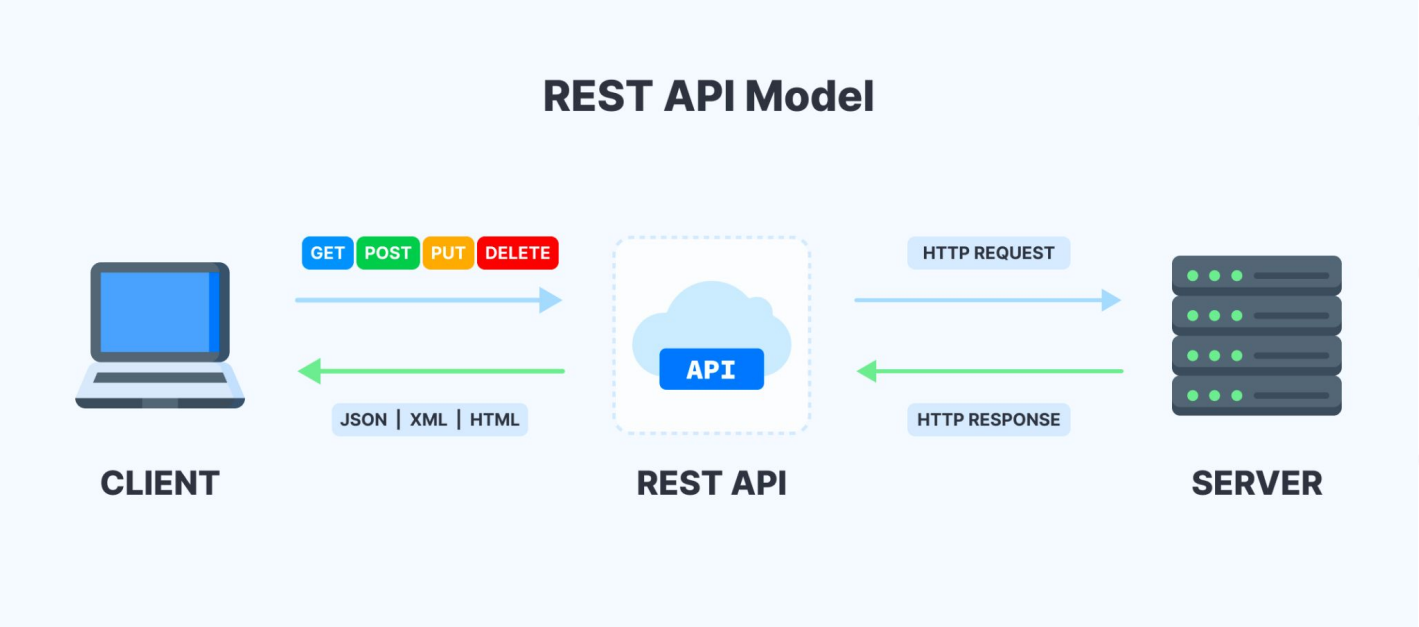
Práctica 2

- Ejecución de código JS en Node.

Práctica 3

- Ejercicio de módulos.
- Objeto Global (this).
- __dirname, __filename, process, setTimeout, setInterval, console.
- Módulos (os, fs, http, url).

Intro a modelo cliente servidor



Concepto de URI

Uniform resource identifier, sirve para acceder a un recurso físico o abstracto por Internet.

protocolo://dominio/path?queryParams#fragment

Concepto de Url

Uniform Resource Locator **es una dirección que es dada a un recurso único en la Web**. En teoria, cada URL valida apunta a un único recurso. Dichos recursos pueden ser páginas HTML, documentos CSS, imagenes, etc.

Diferencia entre URI y URL

Quizás un poco lioso, pero para aclararlo y por decirlo de otra manera:

- Un URL es un URI que identifica un recurso y también proporciona los medios para localizar el recurso mediante la descripción de la forma de acceder a él.
- Un URL es un URI.
- Un URI no es necesariamente una URL.
- Un URN sólo define un nombre, no proporciona detalles acerca de cómo obtener el recurso en una red.
- Puedes obtener una «cosa» a través de una URL, usted no puede conseguir cualquier cosa con un URN
- En general, si el URL describe tanto la ubicación y el nombre de un recurso, el término a utilizar es URI.
- Básicamente, un URI es igual a la suma del URL y del URN.

Práctica 4

- Calculadora Aritmética.
- Exploración del funcionamiento de los módulos.
- Exports e imports.

Práctica 5

- API Rest de páginas web con node.

Práctica 6

- Servidor de archivos multimedia.

Npm

Node Package Manager o manejador de paquetes de node, es la herramienta más popular de JavaScript para compartir e instalar paquetes. Se compone de 2 partes:

- **Un repositorio online para publicar paquetes** de software libre para ser utilizados en proyectos Node.js
- **Una herramienta para la terminal (CLI)** para interactuar con dicho repositorio que ayuda a la instalación de utilidades, manejo de dependencias y la publicación de paquetes.

Comandos de Npm

Inicialización y arranque

- npm init
- npm start

Comandos de Npm

Instalar/desinstalar dependencias

- npm install <package-name>
- npm install -g <pakage-name>
- npm install –save <pakage-name>
- npm install -D <package-name>
- npm uninstall <package-name>
- npm uninstall -g <package-name>

Comandos de Npm

Gestión de dependencias

- npm search <package-name>
- npm ls
- npm update -save
- npm list
- npm list -g –depth 0
- npm outdated

Paquetes

Son módulos distribuidos en forma de librerías que resuelven alguna necesidad de desarrollo. A continuación se listan los más populares al 2022:

- npm.
- create-react-app.
- vue-cli.
- grunt-cli.
- mocha.
- react-native-cli.
- gatsby-cli.
- forever.

Scripts

Son comandos propios que se pueden agregar al package.json para poderlos ejecutar con **npm <my-comand>.**

Estructura de proyecto npm

- node_modules
- package.json
- package-lock.json

Detalle del package.json

- name.
- version.
- description.
- license.
- scripts.
- devDependencies.
- dependencies.

Detalle del package-lock.json

- Este archivo tiene las versiones exactas de las dependencias utilizadas por un proyecto npm.
- No está pensado para ser leído línea por línea por los desarrolladores.
- Es usualmente generado por el comando **`npm install.`**

Práctica 7

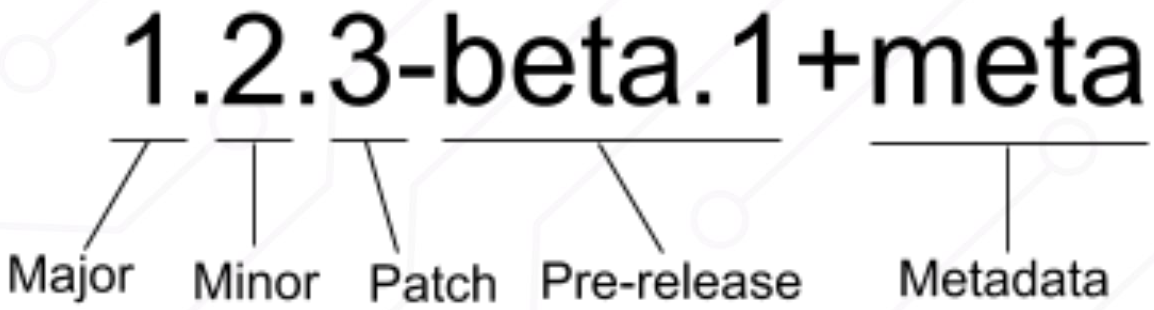
- Agregando npm y nodemon al servidor de archivos.

Semantic Versioning

DEV.F::
DESARROLLAMOS(PERSONAS);

Semantic Versión

Es un conjunto simple de reglas y requerimientos que dictan cómo asignar e incrementar los números de la versión de un software. Evitan la pérdida de versiones y mejoran la gestión de dependencias.



Funcionamiento de semantic versión

Dado un número de versión **MAYOR.MENOR.PARCHE**, se incrementa:

- la versión **MAYOR** cuando realizas un cambio incompatible en el API,
- la versión **MENOR** cuando añades funcionalidad que compatible con versiones anteriores, y
- la versión **PARCHE** cuando reparas errores compatibles con versiones anteriores.

MAYOR.MENOR.PARCHE = MAJOR.MINOR.FIX.

Ejemplo: 1.2.1

Práctica 8

- Definir los siguientes conceptos:

Entorno de ejecución, manejador de paquetes, CLI, comandos, dependencia y script.

Arquitecturas de software

DEV.F::
DESARROLLAMOS(PERSONAS);

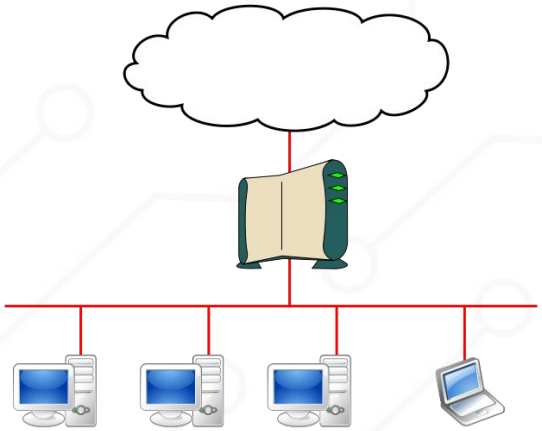
Arquitecturas más comunes

Generalmente, no es necesario inventar una nueva arquitectura de software para cada sistema de información.

Lo habitual es adoptar una arquitectura conocida en función de sus ventajas e inconvenientes para cada caso en concreto.

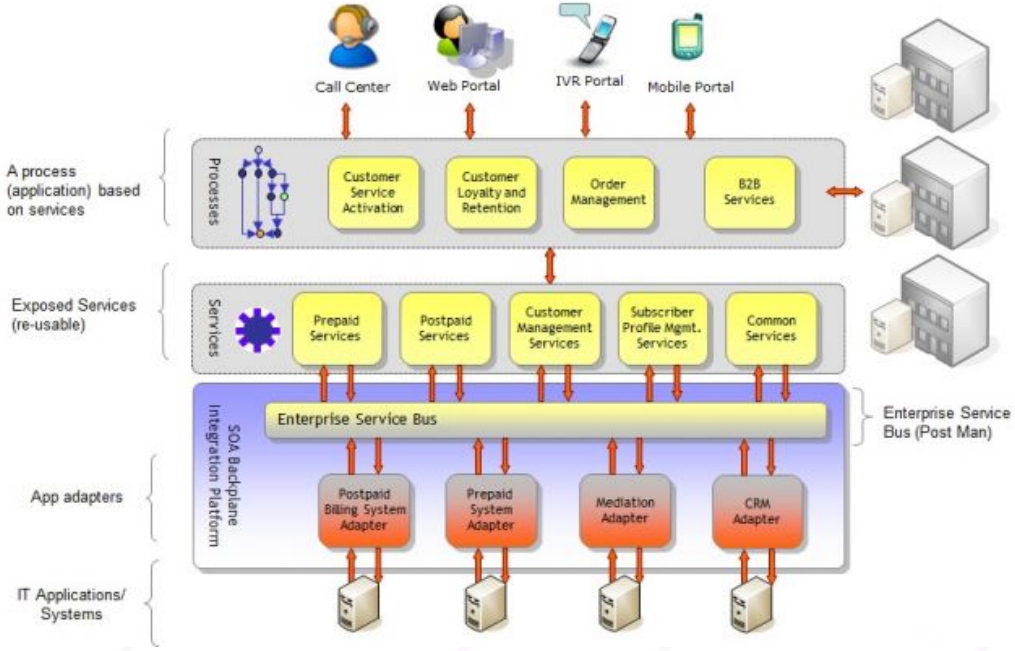
¿Qué es una arquitectura en software?

- Es la estructura y forma en que los componentes de software o hardware se distribuyen y relacionan en el stack.
- La arquitectura de software es el diseño de más alto nivel de la estructura de un sistema.
- Definir las herramientas, patrones y lineamientos con los que una aplicación va a trabajar.



¿Por qué hablar de arquitectura?

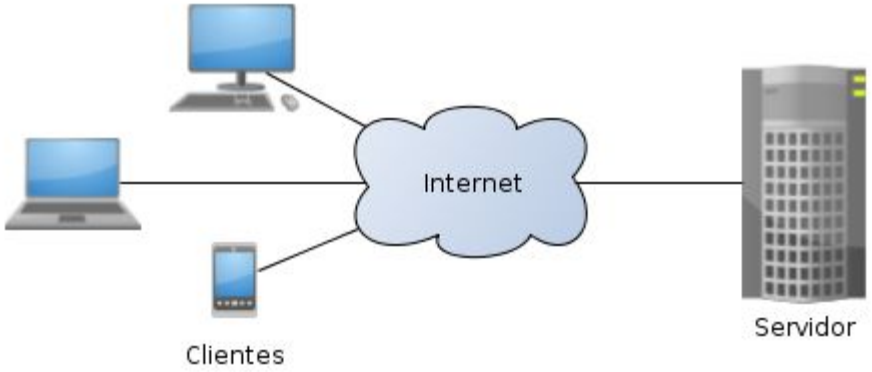
A semejanza de los planos de un edificio o construcción, estas indican la estructura, funcionamiento e interacción entre las partes del software.



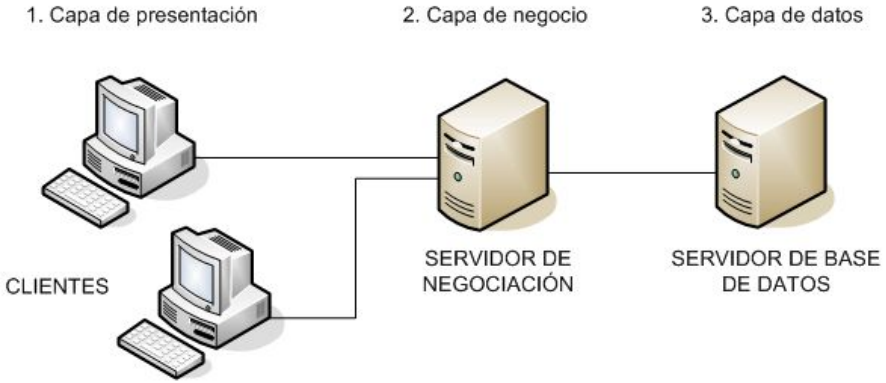
Stacks de desarrollo web

DEV.F:
DESARROLLAMOS(PERSONAS);

Cliente Servidor



Arquitectura de Tres Niveles



Arquitectura MVC

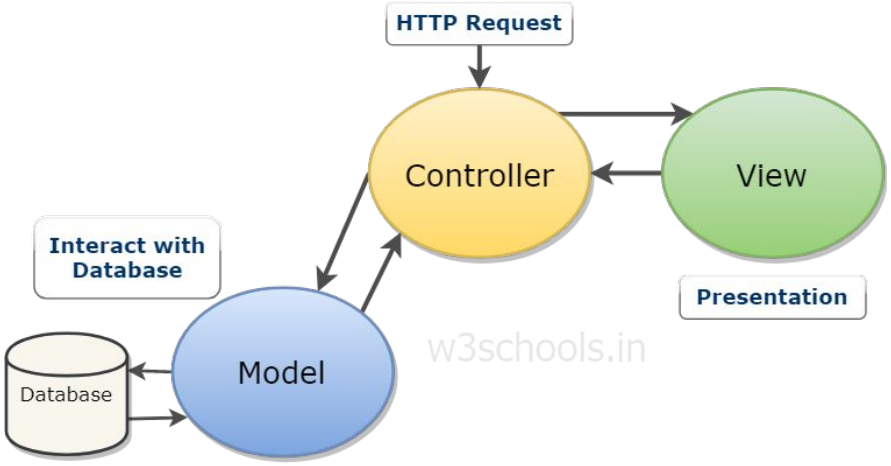


Fig: MVC Architecture

Arquitectura dirigida por eventos

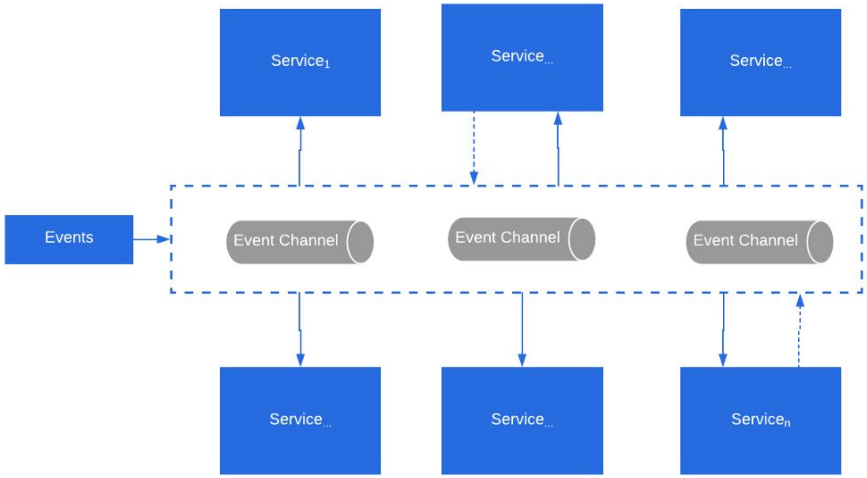
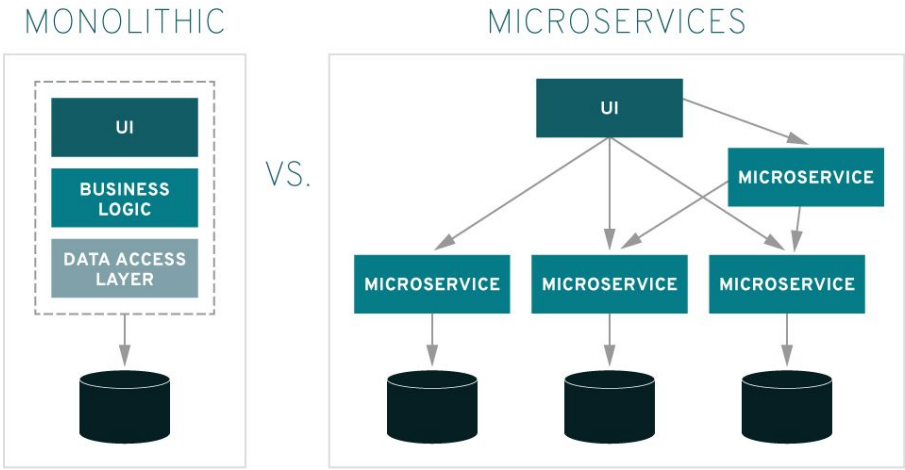


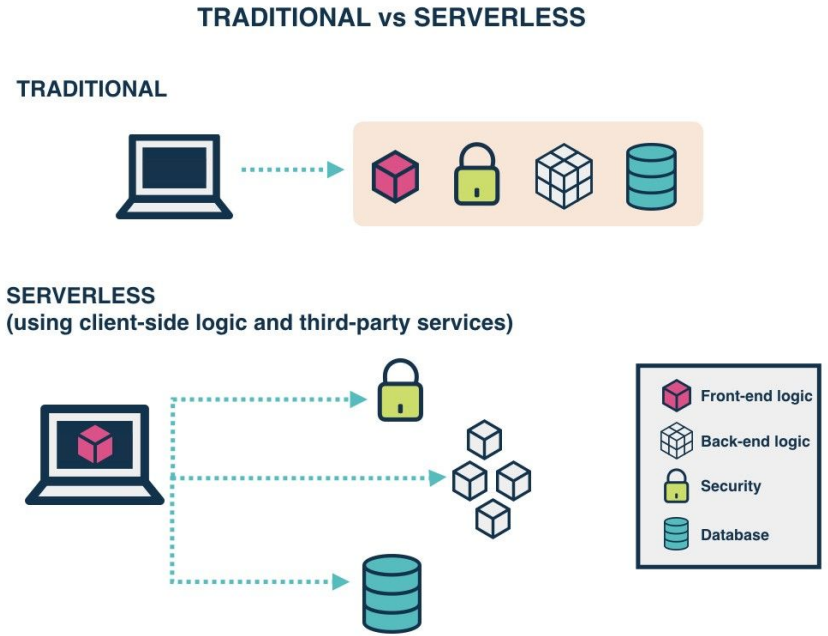
Figure 2 - Broker Topology

Arquitectura de Micro Servicios



DEV.F::

Serverless


TRADITIONAL vs SERVERLESS
TRADITIONAL
SERVERLESS
(using client-side logic and third-party services)
Front-end logic
Back-end logic
Security
Database

Stacks de Desarrollo web

DEV.F:: DESARROLLAMOS(PERSONAS);

¿Qué es un stack tecnológico?

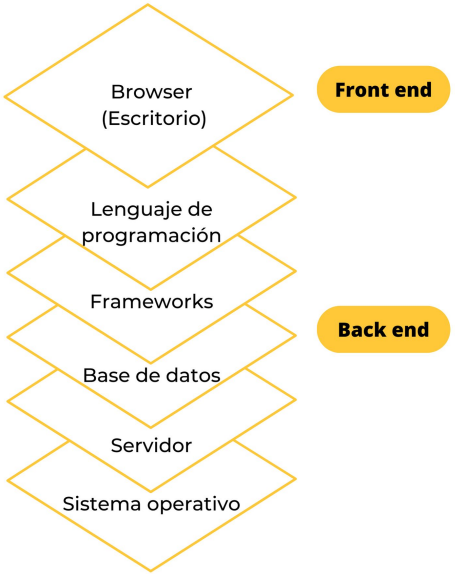
Un stack tecnológico es un **conjunto de servicios de software que se utilizan para el desarrollo de aplicaciones**. Normalmente, un Stack está formada por lenguajes de programación, Frameworks, bibliotecas, herramientas de desarrollo y enfoques de programación.

Hace referencia al método de apilamiento de los componentes de este conjunto de herramientas, uno encima del otro.

¿Qué es un stack tecnológico?

Un stack de **desarrollo** web se compone de:

- Un sistema operativo.
- Un servidor web.
- Una base de datos.
- Un intérprete de lenguaje de programación.

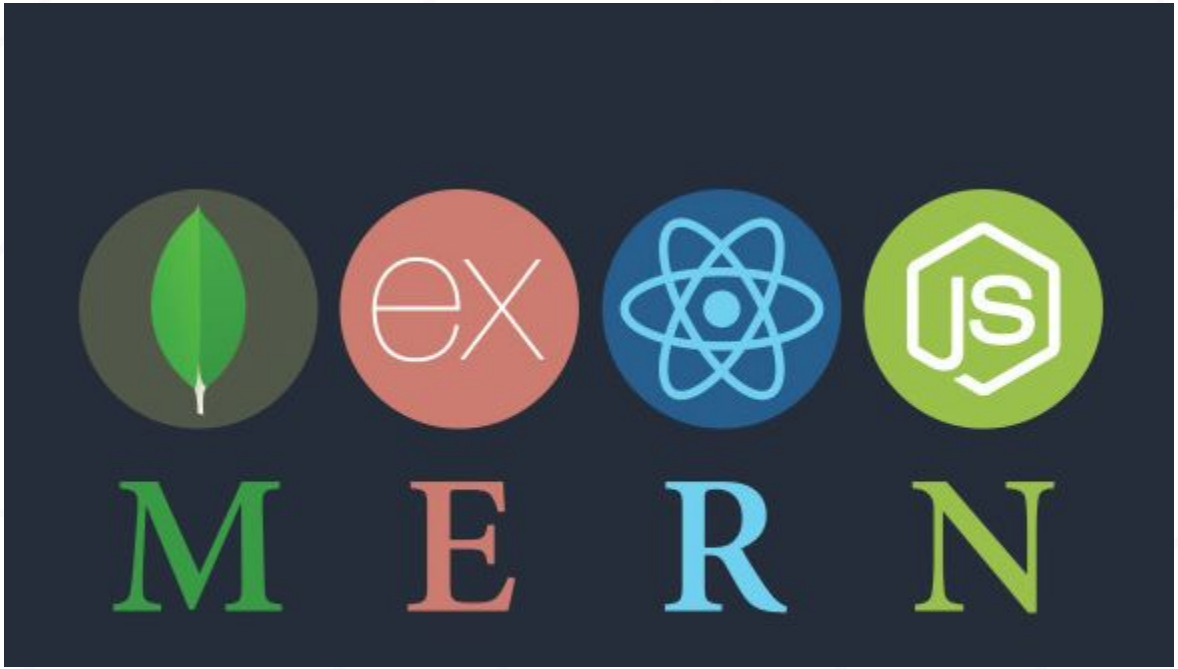


LAMP

ASP.NET

MEAN

MERN



Comparativa entre Node vs JavaScript

DEV.F:
DESARROLLAMOS(PERSONAS);

Notas interesantes

El motor v8 tiene las siguientes características:

- 0 retraso.
- No hay temporizador (setTimeout y setInterval).
- No hay manejo de eventos (eventListener).
- No hay solicitudes Ajax (consumo de API’s).
- Navegadores usan Libevent.
- Node usa Libuv.

Event Loop

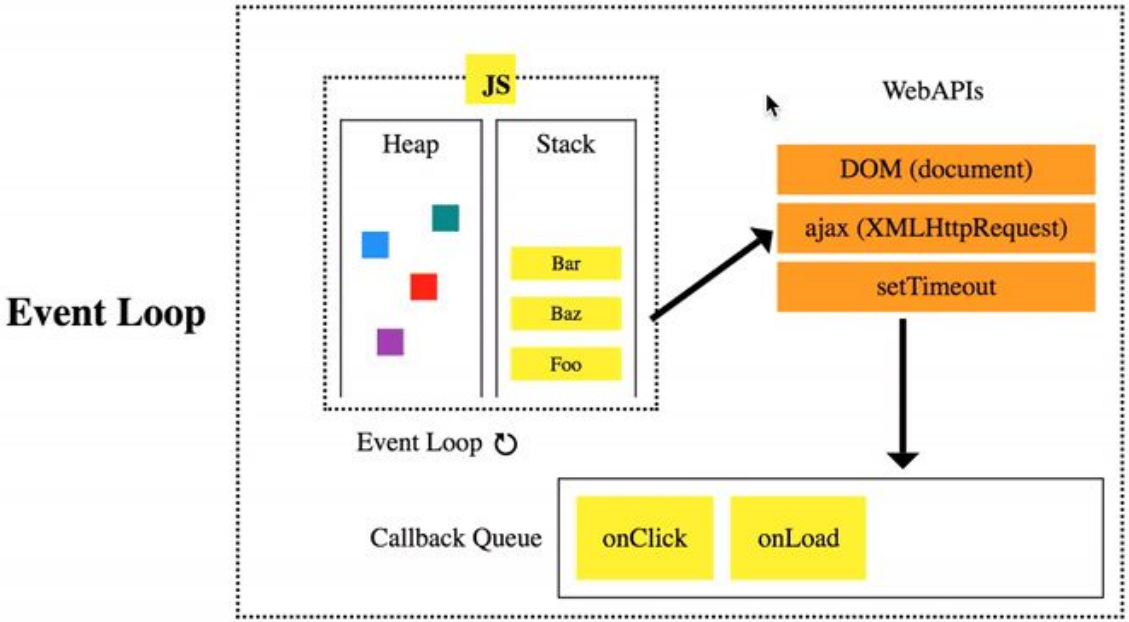
DEV.F:
DESARROLLAMOS(PERSONAS);

Asincronía

DEV.F::
DESARROLLAMOS(PERSONAS);

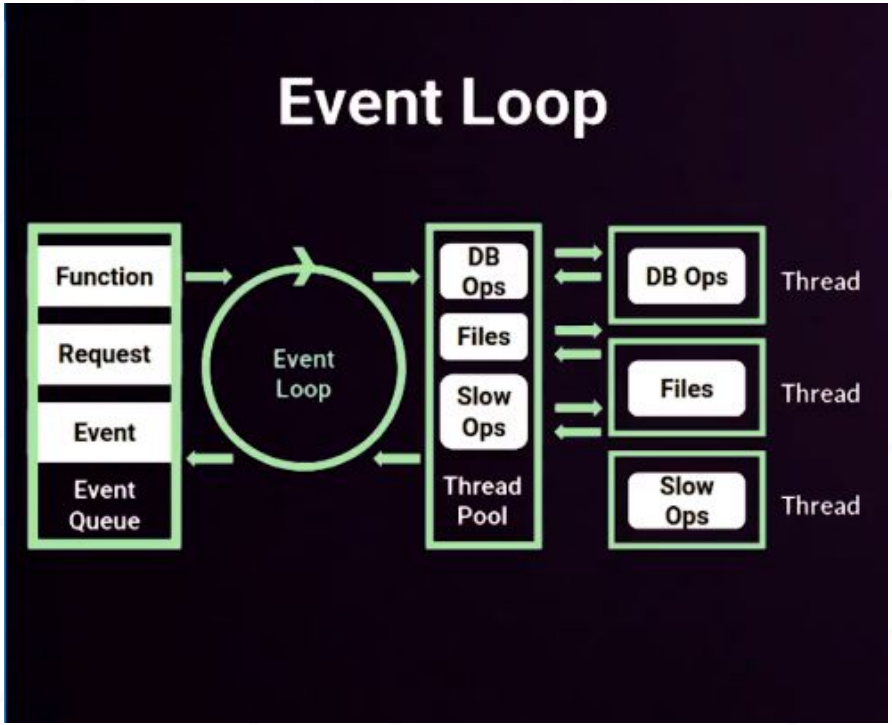
Event Loop JavaScript (libevent)

- Asincronismo y monohilo.



Event Loop Node (libuv)

- Asincronismo y monohilo.



Ejemplo de restaurante para asincronismo

API Rest

DEV.F:
DESARROLLAMOS(PERSONAS);

Comunicación entre distintos back end

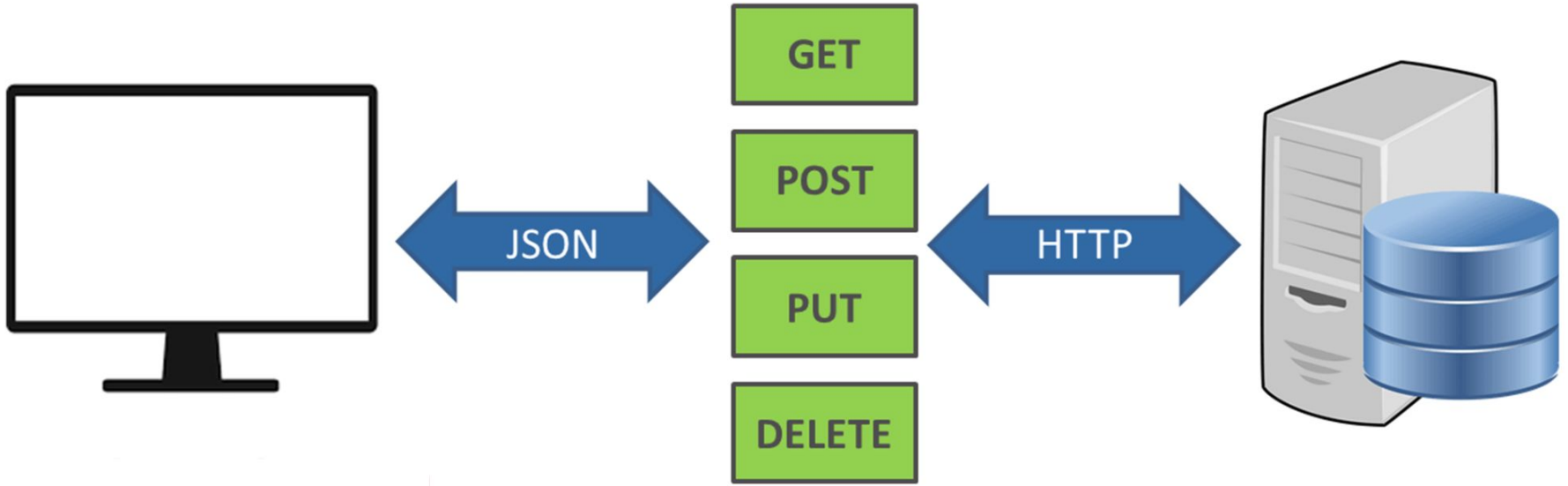
¿Qué es REST?

- REST es una interfaz para conectar varios sistemas basados en el protocolo HTTP) y nos sirve para obtener y generar datos y operaciones (conectar sistemas), devolviendo esos datos en formatos muy específicos, como XML y JSON.
- Es una alternativa a SOAP.
- Soluciona la complejidad de SOAP, haciendo mucho más fácil el desarrollo de una API REST.

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 recurso en el servidor.
- **DELETE**: Verbo que elimina física o lógicamente el recurso en el servidor.

Arquitectura REST



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”

Conceptos

- **TCP/IP** : Transfer control protocol/Internet Protocol.
- **REST**: Representational State Transfer. NO ES UN PROTOCOLO, es una convencion
- **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 hacia el 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.

REST - CRUD

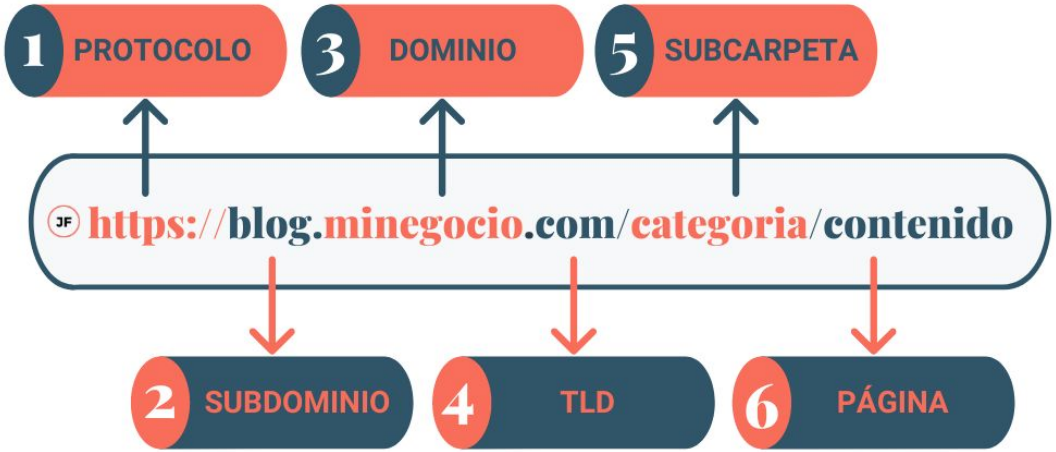
CREATE - READ - UPDATE - DELETE

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

Request y Response

- Un request es una solicitud de una URL o recurso. Los recursos se componente de:
- https => Protocolo
- mi.banco.com => Dominio
- customers/1 => Path
- **Protocolo, host, puerto, path, query strings, fragments y body.**



Terminología.

- **Servicio, endpoint, recurso**: básicamente son lo mismo.

  Una url que tiene una dirección y ciertas directrices de acceso.

- **API:** Es un conjunto de servicios disponibles para su uso. Se debe cumplir un contrato para usarlas.
- **API REST:** Una api que cumple con los principios de RESTful.

ExpressJS (CRUD)

DEV.F:.
DESARROLLAMOS(PERSONAS);

Tema

Algo

Deploys

DEV.F::
DESARROLLAMOS(PERSONAS);

Tema

Algo