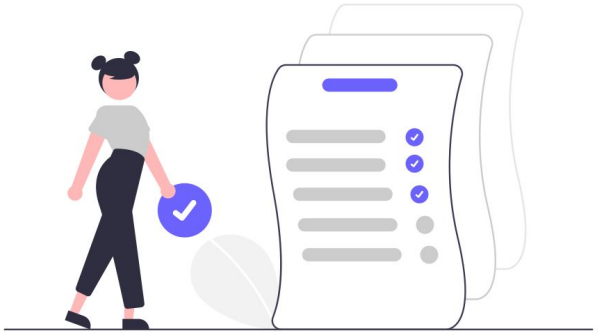
APIs

DEV.F::
DESARROLLAMOS(PERSONAS);

¿Qué aprenderemos?



- ¿Dónde se usan las API?
- ¿Qué es un API, API Rest y API Restful?
- Conceptos de APIs
  - REST Principles.
  - Url y status.
  - Request y response.
- Frameworks de desarrollo back end.
- ExpressJS y documentación.
- Creación de API Rest.

DEV.F:

DEV.F

¿Dónde se usan las API Rest?

Comunicación entre distintos back end

DEV.F

DEV.F::

API

¿Qué es un API?

Application Programming Interface, son **mecanismos** que **permiten** a **dos componentes de software comunicarse entre sí mediante un conjunto de definiciones y protocolos**.

- Normalmente un API conecta un front end y un back end pero también es usual que conecte dos back end.

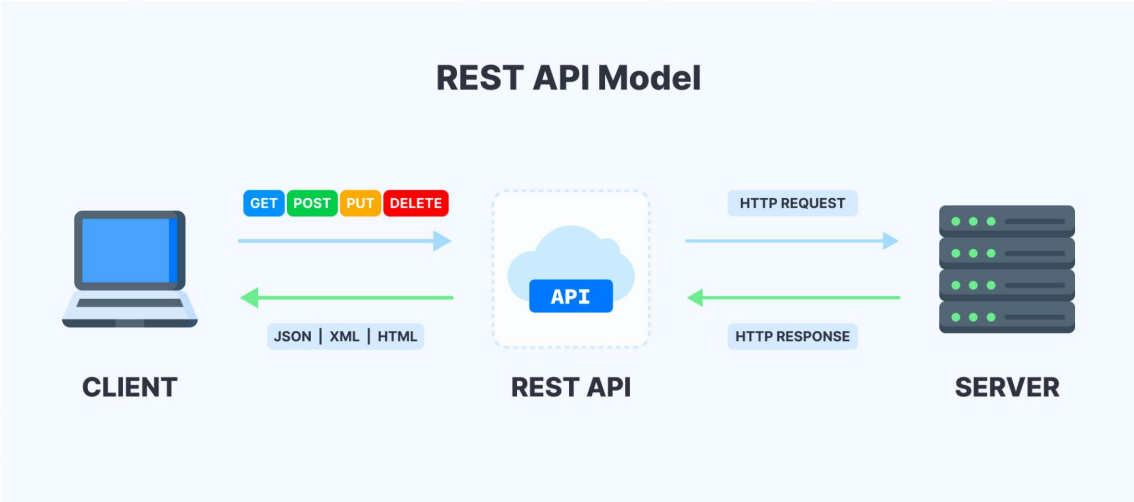
Ejemplo: El sistema de software del instituto de meteorología contiene datos meteorológicos diarios. La aplicación meteorológica de tu teléfono “habla” con este sistema a través de las API y le muestra las actualizaciones meteorológicas diarias en pantalla.

Conceptos de un API

- **Client:** Es quien realiza peticiones de información.
- **Request:** Es la solicitud o petición que realiza el cliente y se conforma de un contrato.
- **Contrato:** Son las condiciones establecidas para poder consumir cada endpoint de un API. Se conforma de:
  - **Url:** Dirección de internet.
  - **Payload:** Información que va en el body de la request.
  - **Método o verbo:** GET, POST, PUT, PATCH y DELETE.
- **Recurso:** Es el endpoint expuesto por el API y consumido por el servidor.
- **Response:** Respuesta de la API al consumir el recurso.
- **Server:** Es donde se aloja el API y es quien responde las peticiones del cliente.

API, API Rest y API Restful

Un **API es un interfaz de comunicación** de aplicación, **REST son los principios que idealmente debe seguir un API** y **ful** hace referencia a un API que está netamente construida bajo los principios REST.



Estructura de una url

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.

Status Codes

NOTA: Consultar estatus recomendados a aprenderse aquí.

DEV.F::

REST

¿Qué es REST?

- Es una **serie de principios** (buenas prácticas) que las **API** siguen para “volverse” **=> API REST (API Restful)**.
- Es una “evolución” de SOAP (servicios web).
- REST es una interfaz para conectar varios sistemas. Se basa en el protocolo HTTP y nos sirve para obtener y generar datos, dichos datos son transmitidos en distintos formatos como como XML y JSON.

Principios de REST

- **Client - server:** El cliente y el servidor deben **ser completamente independientes** entre sí. El cliente solo debe conocer el **contrato** del recurso solicitado.
- **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.
- **Cacheable**: Tanto en el servidor como en el cliente. 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

- **Uniform Interface**: Todas las solicitudes de API para el mismo recurso deben ser iguales. Mismas entradas, mismas salidas.
- **Layered system**: En las API REST, las llamadas y respuestas pasan por diferentes capas. Como regla general, no debe suponerse que las aplicaciones de cliente y de servidor se conectan directamente entre sí.
- **Code on demand (optional):** Las API REST envían recursos estáticos, pero en algunos casos, las respuestas también pueden contener un código ejecutable (como applets de Java). En estos casos, el código solo debería ejecutarse bajo demanda.

DEV.F:

Jerga de APIs

Jerga de APIs

- Servicio ≈ endpoint ≈ recurso ≈ servicio web.
- API ≈ API REST ≈ API Restful.
- Request = Petición = Solicitud.
- Response = Respuesta.
- Path ≈ Dirección ≈ Url ≈ URI.
- File = Fichero = Archivo.
- Terminal = Línea de comandos = CLI = Bash.
- Script ≈ Programa ≈ Conjunto de líneas de código o instrucciones.
- Entidad (BD) ≈ Clase (Programación) ≈ Recurso (API).

DEV.F::

Frameworks de desarrollo back end

Frameworks de desarrollo back end

Listado de frameworks back end.

DEV.F::

Express JS

ExpressJS

- Mayo 2010 (TJ Holowaychuk).
- Es un framework de node para la construcción de aplicaciones web (API Rest y Server Side Render).

Express JS

Es el framework backend más popular para Node.js, y es una parte extensa del ecosistema JavaScript.

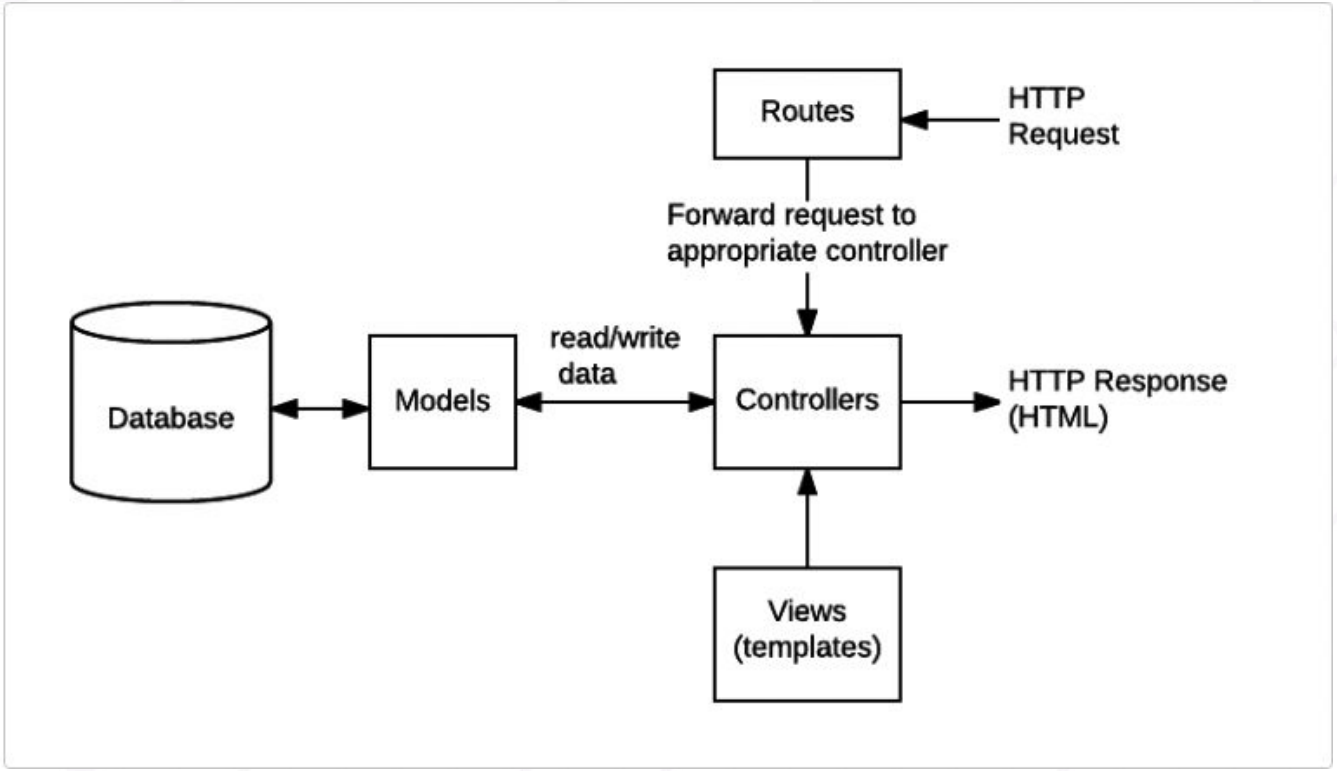
Está diseñado para construir APIs y aplicaciones web de una sola página (SSR), multipágina e híbridas.

- Es minimalista y rápido.
- Proporciona características y herramientas robustas para desarrollar aplicaciones de backend escalables.
- Potente sistema de enrutamiento.
- Usa middlewares.
- Convivencia con todos los framework front end y ssr populares.
- Adaptable arquitecturas de microservicios.

Conceptos en expressJS

- Servidor.
- Request y response.
- Status.
- Query y request params.
- Body.
- Router.
- Controller.
- Service.
- Model.
- Cors y env.

Arquitectura de proyectos node



Vamos al código !