Desarrollo Backend con NodeJS

API's, HTTP Requests y rutas

course[It]

Frameworks

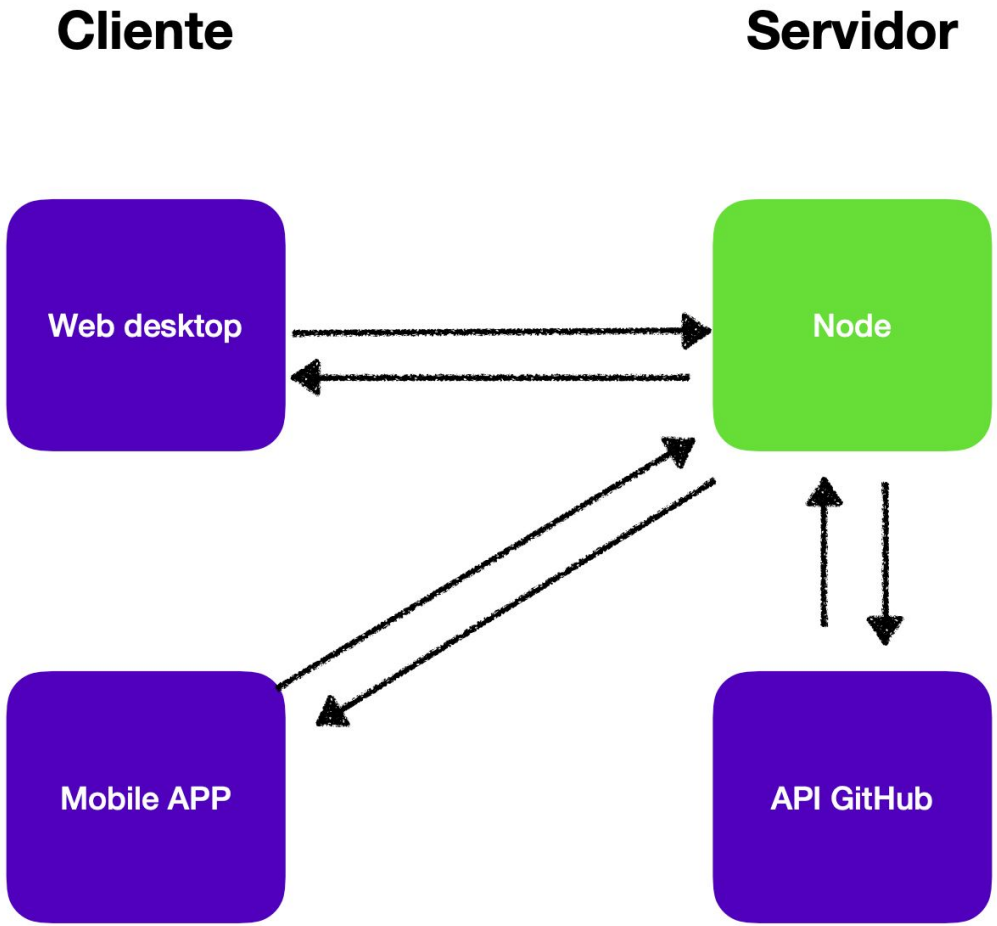
Los frameworks y librerías son porciones de código abierto que van a hacer nuestra vida más fácil como desarrolladores

Express

Express es un framework de NodeJS creado en 2010. Express sirve para crear tanto aplicaciones web como APIs

APIS

El término API viene de las palabras “Application programming interface” y nos va a servir para que nuestra información pueda **ser consumida** de forma genérica por **otros servidores o clientes**. Usualmente se expone en un formato que se llama JSON



HTTP

HTTP es un protocolo que sirve para establecer una comunicación entre dos partes

Métodos HTTP

- GET
- POST
- PUT
- DELETE

GET

Sirve para pedir **información**, se pueden enviar parámetros que viajarán expuestos en la url por eso nunca se deben utilizar para **comunicar información sensible**

POST

Sirve para enviar **información**, se pueden enviar parámetros que viajarán ocultos en el **body** de la request, por eso se lo considera la forma segura de enviar información sensible

PUT

Sirve para modificar **información**, al igual que post se pueden enviar parámetros que viajarán ocultos en el **body** de la request, por eso se lo considera la forma segura de modificar información sensible

DELETE

Sirve para borrar **información**, los parámetros al igual que en el get se envían expuestos en la url

Códigos de estado

O status codes, son una convención que se utiliza para notificar sobre el estado de la comunicación en las requests HTTP

- 10x - informativos
- 20x - éxito
- 30x - redirecciones
- 40x - errores en el cliente
- 50x - errores en el servidor

Importante

Es importante destacar que todo esto es una convención, y por lo tal requiere ser implementada por los desarrolladores de forma correcta

TL;DR: ¿Puedo mandar un mensaje de éxito acompañado de un estado 400? SI. ¿Debería? **CLARO QUE NO**

¿Dónde aplicamos todo este protocolo?

Rutas

También los van a escuchar mencionados como **endpoints** por su nombre en inglés.

Son el **punto de entrada** de nuestra aplicación. Esto quiere decir que va a ser el lugar por donde vamos a recibir las llamadas HTTP de quien nos consuma

Pueden ser estáticas o dinámicas

Express middleware

A nivel general se considera middleware a cualquier capa que exista para facilitar la comunicación entre un sistema y la aplicación que lo usa.

En Express el middleware **nos simplifica** la forma en la **que le pedimos** a node que realice requests HTTP

GET request sin Express

```
const https = require('https')
const options = {
  hostname: 'whatever.com',
  port: 443,
  path: '/todos',
  method: 'GET'
}

const req = https.request(options, res => {
  console.log(`statusCode: ${res.statusCode}`)

  res.on('data', d => {
    process.stdout.write(d)
  })
})

req.on('error', error => {
  console.error(error)
})

req.end()
```

GET request sin Express

```
const express = require("express");
const router = express.Router();
const app = express();

router.get('/handle',(req, res) => {
    // código a ejecutar
});

// add router in the Express app.
app.use("/", router);
```

Los objetos req y res

- **req** - representa la request realizada a ese endpoint, tiene métodos para manejar la información que recibimos cuando la ruta es consumida
  - req.query
  - req.params

- **res** - representa la respuesta que realiza nuestro endpoint al ser consumido, tiene métodos para manejar la información que enviamos
  - res.send
  - res.json
  - res.status

Ahora sí, finalmente a hacer lo que vinimos a hacer

- `npx express-generator nombreDelDirectorioDeseado --no-view`
  - http://expressjs.com/en/starter/generator.html#express-application-generator