

Agencia de
Aprendizaje
a lo largo
de la vida

Desarrollo Fullstack



Les damos la bienvenida

Vamos a comenzar a grabar la clase

Clase 26

Express JS

- ▶ Express JS
- ▶ Express Generator
- ▶ Servidor Estático con Node

Clase 27

Node JS

- ▶ Request y Response
- ▶ GET
- ▶ Rutas Parte I
- ▶ Path Params
- ▶ Query Params

Clase 28

Node JS

- ▶ Express Router
- ▶ Body Parser
- ▶ Middlewares

NODE JS

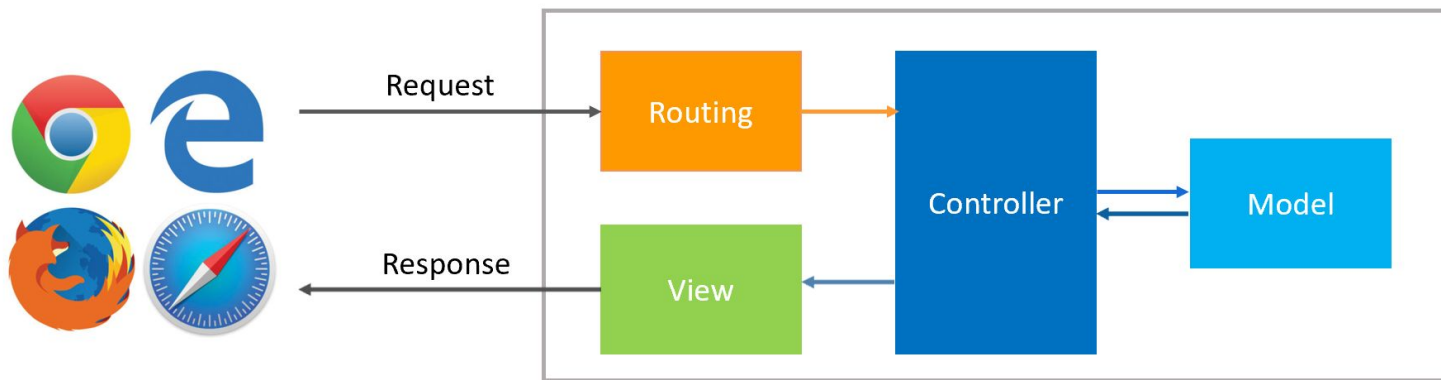
Rutas - Parte I



La clase anterior nos quedamos en...

Rutas

El **cliente** **buscará** acceder al contenido **NO ESTÁTICO** de nuestro **Backend** a través de **peticiones HTTP** a diferentes **rutas** o **endpoints** configurados en nuestra aplicación.



Las rutas son la manera que tiene nuestro servidor de exponer contenido “*no estático*” a través de la web.

Rutas

Recordemos que al usar el **protocolo HTTP**, las peticiones o **requests** se hacen mediante el uso de los HTTP methods.

GET

POST

PATCH

PUT

DELETE

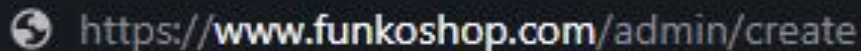
Rutas

Por ende, nuestro servidor no solo escuchará “**paths**” o rutas si no que también tendrá en cuenta el método utilizado en la request.

Esto nos permite usar la misma ruta para diferentes cosas.

Por ejemplo...

No es lo mismo solicitar la ruta **“/admin/create”** mediante **GET** en la **URL**:



Que utilizar un formulario para enviar datos a través de **POST**:

CREAR NUEVO ITEM

Categoría: Licencia:

Nombre del producto: Kakashi Hatake Shippuden Saga

Descripción del producto

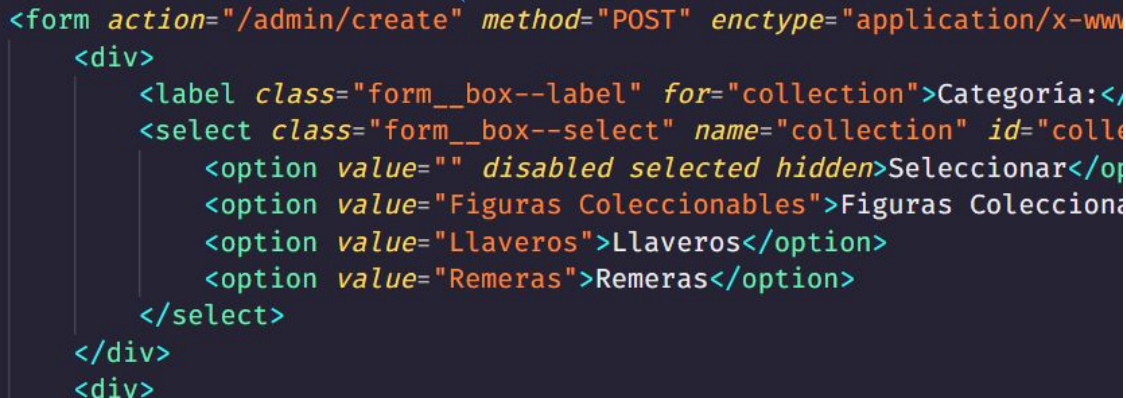
SKU: SSK111AB001 Precio: \$ 0.000,00 Stock: 0

Descuento: 0% Cuotas:

Imágenes: No se ha seleccionado ningún archivo

Agregar Producto


Limpiar



```
<form action="/admin/create" method="POST" enctype="application/x-www
<div>
  <label class="form__box--label" for="collection">Categoría:</
  <select class="form__box--select" name="collection" id="colle
    <option value="" disabled selected hidden>Seleccionar</op
    <option value="Figuras Coleccionables">Figuras Colecciona
    <option value="Llaveros">Llaveros</option>
    <option value="Remeras">Remeras</option>
  </select>
</div>
</div>
```

Rutas

Nuestras rutas serán definidas en Express de la siguiente manera:



El método **get** de **app** escuchará las peticiones a la ruta **/admin/create** a través del método **HTTP GET** y responderá el archivo solicitado.

```
app.get('/admin/create, (req, res) => {  
  res.send(__dirname + './create.html');  
})
```

la variable **app de express puede escuchar a todos los métodos HTTP, entre ellos **GET, POST, PATCH, PUT** y **DELETE**, entre otros.*

__dirname nos permite tomar como referencia **el lugar actual de nuestro archivo** dentro del servidor y **llegar a un recurso** desde esa ruta.

Respondiendo a rutas

Tenemos un archivo `items.json`, el cual **queremos leer** y **devolver** cuando un cliente pida la ruta `"/items"`.

```
app.get("/items", (req, res) => {  
  const getItems = fs.readFileSync(__dirname + '/data/items.json');  
  res.send(JSON.parse(getItems));  
})
```

En este caso si **entramos** a `localhost:3000` seguiremos obteniendo nuestro archivo `index.html` (estático) pero si solicitamos esta ruta nos devolverá un **array con los productos** en formato JSON.

***readFileSync** nos devuelve un string o cadena de texto con la información y nosotros la convertimos a JSON a través del método `JSON.parse()`.

Rutas Parametrizadas

Rutas parametrizadas - params

Ahora supongamos que queremos **traer** solo **un ítem** de la **lista**

¿Cómo podríamos resolverlo?

Las respuestas son las **“*rutas parametrizadas*”**, gracias a ellas **podemos leer una parte de la URL** y utilizarla para devolver una respuesta diferente según el caso.

```
app.get("/items/:id", (req, res) => {  
  const id = req.params.id;  
  // lógica  
})
```

En este caso **:id** será un valor que pasaremos en la URL y será leído al momento de recibir la petición.

A través de **params**, capturamos el valor de la URL.

Rutas parametrizadas - query

Otra manera de pedir datos a través de la URL es mediante los **querys**.

```
http://localhost:3000/items?licence=pokemon
```

```
app.get("/items", (req, res) => {  
  const licence = req.query.licence; //pokemon  
  // lógica que filtra los ítems de la licencia  
  pokémon  
})
```

Usamos **la misma ruta que para todos los items**, solo que si recibimos un **query param**, capturamos el valor de la URL e **incluimos una lógica** para devolver solo los valores coincidentes.

No te olvides de dar el presente

Recordá:

- **Revisar la Cartelera de Novedades.**
- **Hacer tus consultas en el Foro.**

Todo en el Aula Virtual.

Gracias