**URI.JS QUERIES**

[**URI.js - URLs in Javascript**](https://medialize.github.io/URI.js/)

**Manipulación de URLs al estilo "jQuery":**

**Mutar URLs:**

**.username("rodneyrehm"): Añade o cambia el nombre de usuario en la URL.**

**.username(""): Elimina el nombre de usuario de la URL.**

**.directory("bar"): Cambia el directorio de la URL.**

**.suffix("xml"): Cambia la extensión del archivo en la URL.**

**.hash("hackernews"): Añade un fragmento (hash) al final de la URL.**

**.fragment(""): Elimina el fragmento de la URL.**

**.search(""): Elimina la cadena de búsqueda (parámetros) de la URL.**

**.tld("com"): Cambia el dominio de nivel superior (TLD) en la URL.**

**.search({ foo: "bar", hello: ["world", "mars"] }): Añade o modifica parámetros de búsqueda en la URL.**

**Trabajo con cadenas de consulta (query strings):**

**Manipular cadenas de consulta:**

**.addSearch("hello", "mars"): Añade un nuevo valor al parámetro existente en la cadena de búsqueda.**

**.addSearch({ foo: ["bar", "baz"] }): Añade múltiples valores a un nuevo parámetro en la cadena de búsqueda.**

**.removeSearch("hello", "mars"): Elimina un valor específico del parámetro en la cadena de búsqueda.**

**.removeSearch("foo"): Elimina todo el parámetro de la cadena de búsqueda.**

**Trabajar con rutas relativas:**

**Rutas relativas:**

**.relativeTo("/relative/sub/foo/sub/file"): Genera la ruta relativa desde una ruta base especificada.**

**.absoluteTo("/relative/sub/foo/sub/file"): Convierte la ruta a una ruta absoluta en base a una ruta base especificada.**

**Limpieza de URLs:**

**Normalización de cadenas de búsqueda y rutas:**

**.normalizeSearch(): Limpia la cadena de búsqueda, eliminando duplicados y segmentos innecesarios.**

**.normalizePathname(): Limpia la ruta, eliminando segmentos innecesarios como .. y ..**

**Detección de URLs dentro de texto aleatorio:**

**Detectar URLs en texto:**

**URI.withinString(source, function(url) { ... }): Detecta URLs dentro de una cadena de texto y permite manipularlas, por ejemplo, envolviéndolas en etiquetas HTML <a>.**

**Comparación de URLs:**

**Comparar URLs:**

**.equals(b): Compara dos URLs para verificar si son equivalentes, ignorando diferencias menores como mayúsculas, minúsculas o el puerto por defecto.**

**Manipulación de fragmentos (hash):**

**Manipular el fragmento de la URL:**

**.fragment({ foo: "bar", bar: ["hello", "world"] }): Almacena datos estructurados en el fragmento de la URL.**

**.fragment(true): Trata el fragmento como una URL separada y permite manipular partes de esa URL.**

**Parsing de URNs:**

**Análisis de URNs:**

**.protocol(): Devuelve el protocolo de la URL (por ejemplo, "urn").**

**.path(): Devuelve la ruta de la URL (en este caso, la parte de la URN).**

**.query(): Devuelve la cadena de consulta de la URL.**

**Plantillas de URI:**

**Uso de plantillas:**

**URI.expand("/foo/{dir}/{file}", { dir: "bar", file: "world.html" }): Rellena una plantilla de URI con los valores proporcionados, generando una URL completa.**

**2. Code Example:**

**1. Importación de Módulos:**

**javascript**

**var express = require('express');**

**var app = express();**

**var fs = require("fs");**

**var bodyParser = require('body-parser');**

* **express: Importa el marco Express.**
* **app: Crea una instancia de una aplicación Express.**
* **fs: Importa el módulo de sistema de archivos para leer y escribir archivos.**
* **bodyParser: Importa el middleware para analizar el cuerpo de las solicitudes HTTP.**

**2. Middleware Body-Parser:**

**javascript**

**app.use(bodyParser.json());**

**app.use(bodyParser.urlencoded({ extended: true }));**

* **bodyParser.json(): Analiza las solicitudes con cuerpos en formato JSON.**
* **bodyParser.urlencoded({ extended: true }): Analiza las solicitudes con datos codificados en la URL.**

**3. Obtener Todos los Usuarios (GET /):**

**javascript**

**app.get('/', function (req, res) {**

**fs.readFile(\_\_dirname + "/" + "users.json", 'utf8', function (err, data) {**

**res.end(data);**

**});**

**});**

* **app.get('/'): Define una ruta GET para la URL raíz /.**
* **fs.readFile(): Lee el archivo users.json y devuelve su contenido como respuesta.**

**4. Obtener un Usuario por ID (GET /:id):**

**javascript**

**app.get('/:id', function (req, res) {**

**fs.readFile(\_\_dirname + "/" + "users.json", 'utf8', function (err, data) {**

**var users = JSON.parse(data);**

**var user = users["user" + req.params.id];**

**res.end(JSON.stringify(user));**

**});**

**});**

* **app.get('/:id'): Define una ruta GET para obtener un usuario por ID.**
* [**req.params.id**](https://req.params.id/?form=MG0AV3)**: Obtiene el ID del usuario desde la URL.**
* **JSON.parse(data): Convierte el contenido del archivo users.json en un objeto JavaScript.**
* **res.end(JSON.stringify(user)): Devuelve los datos del usuario como respuesta JSON.**

**5. Añadir un Nuevo Usuario (POST /):**

**javascript**

**app.post('/', function (req, res) {**

**fs.readFile(\_\_dirname + "/" + "users.json", 'utf8', function (err, data) {**

**var users = JSON.parse(data);**

**var user = req.body.user4;**

**users["user" + user.id] = user;**

**res.end(JSON.stringify(users));**

**});**

**});**

* **app.post('/'): Define una ruta POST para añadir un nuevo usuario.**
* **req.body.user4: Obtiene los datos del nuevo usuario desde el cuerpo de la solicitud.**
* **users["user" + user.id] = user: Añade el nuevo usuario al objeto users.**

**6. Eliminar un Usuario por ID (DELETE /:id):**

**javascript**

**app.delete('/:id', function (req, res) {**

**fs.readFile(\_\_dirname + "/" + "users.json", 'utf8', function (err, data) {**

**data = JSON.parse(data);**

**var id = "user" + req.params.id;**

**delete data[id];**

**res.end(JSON.stringify(data));**

**});**

**});**

* **app.delete('/:id'): Define una ruta DELETE para eliminar un usuario por ID.**
* **delete data["user" + req.params.id]: Elimina el usuario del objeto users.**

**7. Actualizar un Usuario por ID (PUT /:id):**

**javascript**

**app.put("/:id", function(req, res) {**

**fs.readFile(\_\_dirname + "/" + "users.json", 'utf8', function (err, data) {**

**var users = JSON.parse(data);**

**var id = "user" + req.params.id;**

**users[id] = req.body;**

**res.end(JSON.stringify(users));**

**});**

**});**

* **app.put("/:id"): Define una ruta PUT para actualizar un usuario por ID.**
* **users[id] =** [**req.body**](https://req.body/?form=MG0AV3)**: Actualiza los datos del usuario con los nuevos datos del cuerpo de la solicitud.**

**8. Iniciar el Servidor:**

**javascript**

**var server = app.listen(5000, function () {**

**console.log("Express App running at http://127.0.0.1:5000/");**

**});**

* **app.listen(5000): Inicia el servidor en el puerto 5000 y muestra un mensaje en la consola indicando que el servidor está corriendo.**

**Este código configura una aplicación** [**Express.js**](https://express.js/?form=MG0AV3) **que permite listar, obtener, añadir, eliminar y actualizar usuarios almacenados en un archivo JSON. Puedes usar un cliente REST como Postman o Insomnia para enviar solicitudes a estos endpoints y ver cómo funcionan.**