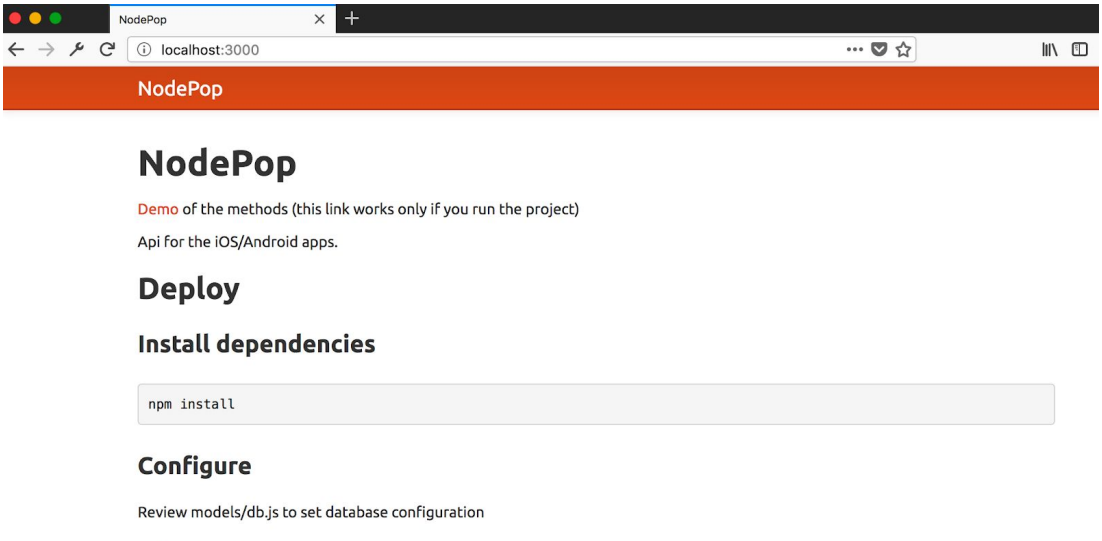
Práctica Backend Avanzado

Bootcamp Web9 2020

La práctica se hará sobre el código que se creó para la práctica de fundamentos de Node.js.

En caso de no disponer de ese código podéis usar dicha práctica resuelta como punto de partida: https://bitbucket.org/agbo/agbo-master-nodejs-practica-web



Índice de retos:

Autenticación

Nuestro API necesita protegerse!

Implementar autenticación JWT al API. (No es necesario implementar autenticación en el website)

Mini-guia:

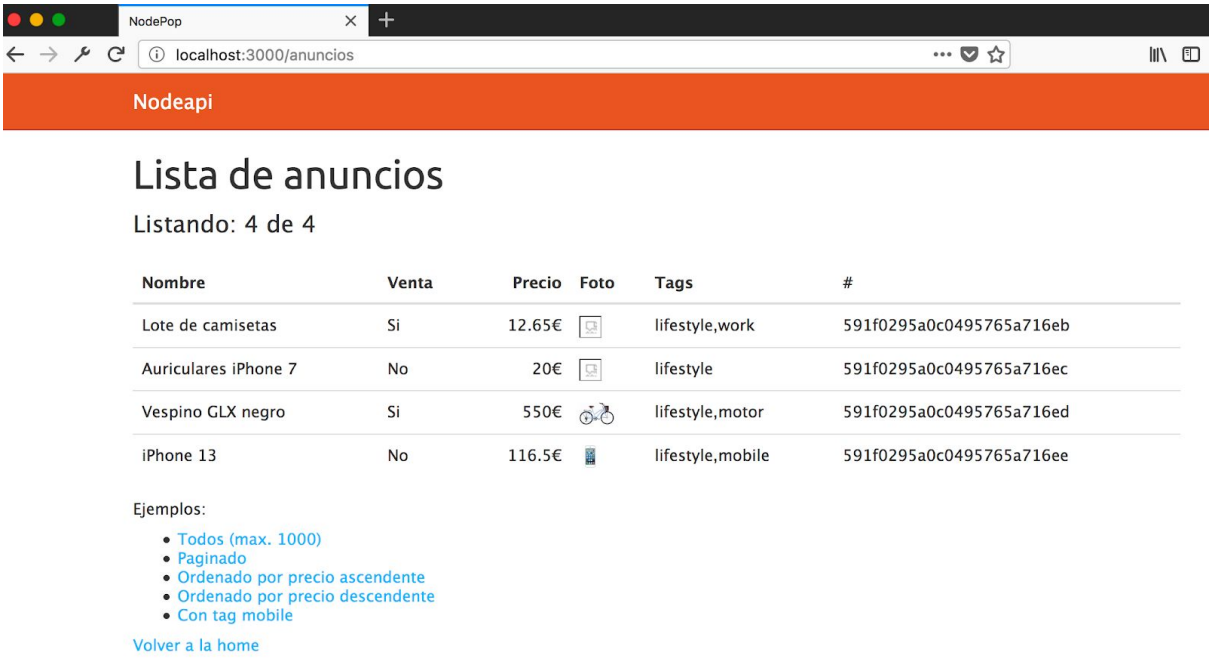
1. POST /api/authenticate para hacer login y devolver un token JWT
2. GET /api/anuncios incluyendo el JWT en una cabecera o query-string hará la petición correcta (200 OK)
3. GET /api/anuncios sin token responderá con un código de status HTTP 401 y un json con info del error
4. GET /api/anuncios con un token caducado responderá con un código de status HTTP 401 y un json con info del error

El API tendrá al menos un usuario con email user@example.com y clave 1234

Os recomiendo hacer este punto el primero, ya que así haréis los siguientes teniendo en cuenta la autenticación.

Internacionalización

Este reto consiste en convertir el frontend de anuncios de la aplicación Nodepop en multi-idioma.



Idiomas disponibles:

- Español
- Inglés

No será necesario internacionalizar el API.

Deberá disponer de un selector de idioma donde el usuario pueda cambiar de inglés a español o viceversa.

Subida de imagen con tarea en background

El API necesita un end-point para crear anuncios. A por ello

Podría ser tal que POST /api/anuncios y debería permitir que el cliente del API suba una imagen y esta sea guardada en el servidor, de tal forma que cuando hagamos las peticiones GET /api/anuncios nos sean devueltas las rutas a éstas imágenes y dichas rutas funcionen.

Cada imagen que se suba debe tener un thumbnail!

Podemos hacer un microservicio que reciba trabajos (con cote.js o de una cola de RabbitMQ), creando dichos thumbnails.

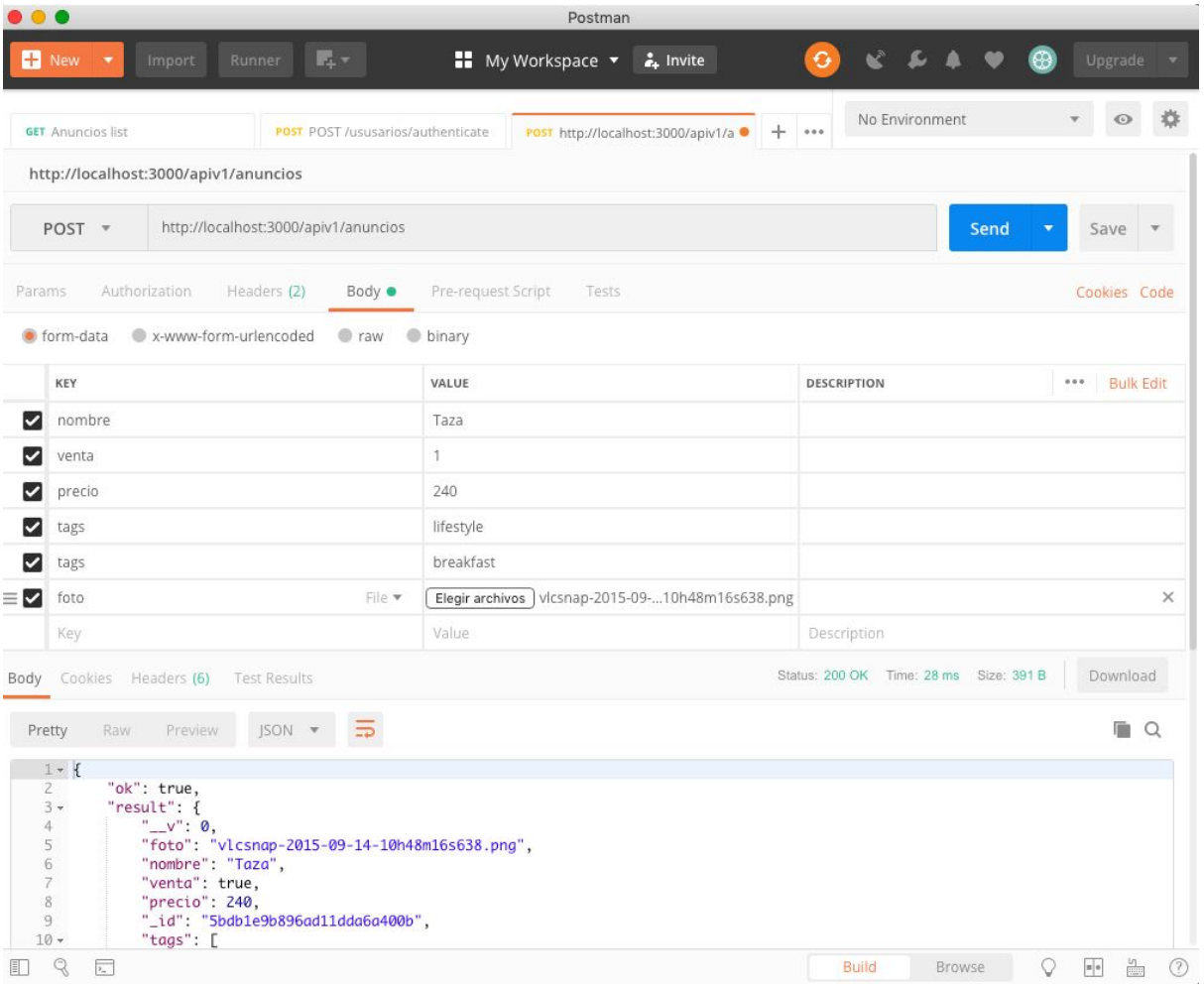
Para crear un thumbnail podemos:

- buscar node.js image resize en google y ver cual probáis, por ejemplo jimp
- instalarla y probarla

- hacer que el API mande un mensaje (con cote.js o con una cola de RabbitMQ) con la ruta del filesystem a la imágen.

- crear un worker que esté suscrito al mensaje/cola y vaya haciendo thumbnails de cada imágen a un tamaño de 100x100 píxeles.

Ejemplo de la llamada de prueba en Postman (fijaros que el tipo del campo 'foto' en la petición es 'File'):



Testing (Opcional)

Hagamos que la calidad sea una característica de nuestro software.

Este reto consiste en incluir tests del API de Anuncios con Supertest
https://github.com/visionmedia/supertest

Los tests se deberán poder ejecutar con `npm test`.

BONUS TRACK! - Crear un módulo público

Si hemos llegado hasta aquí con ganas... publiquemos un trabajo!

Se propone hacer alguna utilidad de nuestra invención que pueda resultar útil para nosotros mismos o para otros.

Puede ser cualquier tipo de utilidad, a continuación unas sugerencias para evocar vuestra creatividad:

- Dhrase, Frase del día
- RabinstonWabbit, transport de Winston a RabbitMQ
- Imandom, que devuelve una URL a una imágen aleatoria online
- Frine, Lector de la primera línea de un fichero de texto
- HipoTalculator, calculador de hipoteca

Colocar la URL de npm.js del módulo que hayáis publicado al inicio del README.md del repo que entreguéis.