Prueba Técnica Node - Backend

Antes de que “ataques” a la prueba técnica que tienes por delante, te damos varios consejos para facilitarte la tarea:

- Lee el apartado Criterios de evaluación con detenimiento antes de pasar a leer el Enunciado del problema a resolver.
- Una vez hayas leído el enunciado del problema, vuelve a releerlo lentamente para captar todos los detalles que te evitarán tener que hacer y rehacer tu código.
- Como pasa siempre, hay diferentes formas de resolver la prueba. Hazlo lo mejor que puedas con los conocimientos que tienes a día de hoy. ☺
- Si hay algo que, por tu nivel técnico, no sabes hacer, intenta conseguir una funcionalidad parecida y si no eres capaz de conseguir algo parecido describe con palabras cómo lo harías.
- Si tienes cualquier duda, principalmente con el servicio WebAPI al que vas a atacar para obtener los datos, mándanos un correo a victormartinez@alub.io empezando el asunto del email con “[DUDA BEJ]”

CRITERIOS DE EVALUACIÓN

Hemos planteado la Prueba como un reto progresivo donde en cada Misión te hemos dejado varios Retos que la hacen “mas interesante”. ¡No es necesario hacerlo todo!

Te recomendamos que resuelvas las Misiones principales, y que dejes para el final los Retos de cada Misión(sobre todo si ves que te dan demasiados dolores de cabeza...Los Retos pueden llegar a ser realmente difíciles).

No es necesario que nos mandes todo resuelto...solo hasta donde tus conocimientos, tus ganas y el límite del tiempo te han dejado llegar. Indícanos en el Read.me qué Retos/Misiones has conseguido acabar. Si algo no te da tiempo, dar una solución aproximativa, imaginativa o descriptiva (texto) es lo que te hará destacar del resto.

Tenemos en cuenta la claridad del código, la estructura y el detalle de las instrucciones para ejecutar el proyecto. En este sentido, **deberás incluir un READ.ME** en la raíz del repositorio que nos compartas donde nos expliques cómo ejecutar el proyecto y cualquier particularidad de setup que sea necesaria (por ejemplo los scripts .sql para generar la base de datos).

Valoraremos cualquier funcionalidad que añadas adicionalmente siempre y cuando estén presentes las que te hemos solicitado, recuerda que **tienes 7 días desde que recibas la Prueba** para hacernos llegar un link a tu repositorio desde donde descargaremos tu solución y la ejecutaremos en un entorno sandboxed. ¡El tiempo pasa volando!

Cuando tengas la solución preparada, mándanos un correo a victormartinez@alub.io empezando el asunto como [SolucionBEJ] e indicándonos una dirección del repositorio con la solución. Recuerda añadir un READ.ME a la raíz del repositorio que como mínimo tiene que tener las instrucciones para ejecutar el proyecto, así como las suposiciones que has hecho, las misiones/retos que has conseguido programar, y las aclaraciones que consideres necesarias (sobre todo si no has podido programar una funcionalidad requerida o de reto).

ENUNCIADO.

PIX-O-MATE LOVING KITTENS es una página web formado por usuarios que aman a sus gatos y utilizan esta web para subir posts de sus mascotas en álbumes.

Debido al nuevo RGPG, Reglamento General de Protección de Gatos, impuesto por la UE, los usuarios están obligados a subir las fotografías de sus gatos encriptadas lo que hace que las fotos parezcan cualquier cosa menos un gato.

Gato sin encriptar

El mismo gato (encriptado)

Por fortuna, nuestra tarea no consiste en encriptar ni desencriptar, la imagen se encripta en la parte del cliente (un SPA en Angular).

PIX-O-MATE LOVING KITTENS está formado por un front en Angular que se comunica contra el Back que vas a crear durante la prueba.

Misión 1. Crear una API de Empresas Gatunas

La red social de Gatos está teniendo bastante éxito, y la dirección quiere ampliar su negocio.

La empresa **ya tiene creadas** APIs para acceder/editar/listar a sus Usuarios/Dueños (users), a sus Posts (posts), y a sus Comentarios (comments).

Pero ahora quiere crear un nuevo endpoint para gestionar Empresas que se quieran anunciar dentro de la plataforma para dar servicio a los Dueños de los gatos.

Tu primera Misión es crear las APIs necesarias en un endpoint "/companies" que permitan **crear** una nueva empresa, **editar** una empresa ya existente, **listar** todas las empresas y hacer **búsquedas**.

A. Creación de Empresas

A la hora de crear una nueva empresa tienes que tener en cuenta las siguientes particularidades.

Necesitarás crear una Tabla "companies" en una Base de Datos Relacional MySQL llamada "pixomatic*XYZ*", siendo *X* la primera letra de tu nombre, *Y* la primera letra de tu primer apellido y *Z* la primera letra de tu segundo apellido. Así si te llamas "Víctor Martínez Calvo", tu base de datos debería llamarse "pixomaticvmc".

En la tabla "Companies" deben aparecer las siguientes columnas: "id" (una columna autoincremental), "name" (nombre de la empresa), "CIF" (número de registro de la empresa, **siempre** comienza con una letra y le siguen 8 números y es único para cada empresa), "shortdesc" (descripción corta de la empresa, máximo 100 carácteres), "description" (descripción larga con sus servicios), "email" (el email del admin de la Empresa), "CCC" (el número de cuenta corriente de la Empresa), "date" (fecha de fundación de la empresa), "status" (si está abierta o cerrada por vacaciones), y "logo" (una url a una imagen almacenada externamente).

Añade cualquier otra columna que creas que pueda ser interesante.

Crea un endpoint que al recibir “name”, “description”, “email”, “CIF”, “logo” cree una nueva Empresa en esta tabla, en caso de no existir. Los campos “shortdesc”, “CCC”, “date” y “status” son opcionales.

NOTA.

> Tú decides, tanto para las columnas obligatorias como para las que hayas añadido, el tipo de datos (SMALLINT, INT, TEXT, ...). Deberías justificar la elección de ese tipo de datos, así como cualquier Restricción/CONSTRAINT que creas necesaria añadir a la tabla.

B. Editar Empresa

Tenemos que permitir que la empresa pueda modificar sus datos. Por lo tanto, debería existir un endpoint al que atacar para poder realizar estas modificaciones.

La empresa puede editar, a través del endpoint, todos los campos salvo el “CIF”, el “date”, su “email” y su “id”. En caso de que la petición intente modificar alguno de estos datos, deberás devolver un error e ignorar totalmente la petición sin realizar ningún cambio en la base de datos.

Te dejamos que elijas el código y mensaje de Error que creas mas conveniente.

NOTA.

> Por simplificar, ya nos encargaremos más adelante, la API es abierta. Es decir, cualquiera podría modificar los datos de cualquier empresa a su antojo. Esto supone un gran riesgo de seguridad, y ,como ya sabes, nunca debería hacerse en producción.

C. Listar Empresas

La aplicación Angular necesita mostrar a los Dueños de los gatos el listado de Empresas disponibles para que puedan elegir a quién contratar un determinado servicio, por lo que se hace necesario facilitar un endpoint que devuelva todas las empresas de la Base de Datos.

NOTA.

> En tus manos dejamos que decidas (y justifiques) qué datos deberían mostrarse, y cuáles no, de todos los guardados.

D. Buscar Empresas.

Los Dueños de los gatos están abrumados por la cantidad de Empresas registradas y necesitan poder tener un buscador básico.

Se ha implementado un buscador básico en Angular donde los Dueños pueden introducir una cadena de texto, y tu misión es crear un endpoint que devuelva aquellas empresas en las que dicha cadena de texto aparezca contenida en su columna “description”.

NOTA.

> El frontal obliga a que el usuario introduzca una cadena de texto de mínimo 3 letras para permitir que el Dueño haga una Búsqueda. Si el usuario introduce una cadena de texto con menos de 3 letras el frontal no activa el botón Buscar.

Retos de la Misión 1

Se han quedado muchos detalles en el tintero, asi que vamos a hacer estas APIs más interesantes. Estos Retos van creciendo en complejidad y puede ser que no seas capaz de hacerlos todos, como comentábamos en el apartado de CRITERIOS DE EVALUACIÓN. Los retos son independientes, por lo que si no sabes atacar uno, ataca al siguiente. Si ves que se te complican demasiado pasa a la Misión 2, que es más sencilla.

RETO A. API paginadas

Hasta ahora Listar y Buscar devolvían todos los resultados y esto puede ser inasumible a nivel de base de datos o a nivel de frontal. Ya se han registrado cerca de 100.000 Empresas y devolver una lista con esta cantidad de empresas, o una búsqueda con 1.000 entradas no es lo óptimo.

Por ello se quieren mejorar estas APIs para que sean paginadas. De esta forma la petición tiene un parámetro adicional “page” que indica el número de página.

Por ejemplo:

Antes:
/companies -> Devolvería todas las empresas de la base de datos.

Ahora:
/companies -> Devuelve las 20 primeras empresas de la base de datos

/companies?page=1 -> Devolvería, asimismo, las 20 primeras empresas.
/companies?page=2 -> Devolvería las empresas 21 a la 40.

En general (siendo N un número):
/companies?page=N -> Devolvería las empresas (N-1)*20+1 a N*2.

PISTAS.

Para implementar esta paginación solo tienes que trabajar con dos conceptos en las consultas MySQL: LIMIT y OFFSET. Ajustando bien el LIMIT y el OFFSET y usando el valor de “page” es suficiente para conseguir la paginación.

NOTA.

Deberías devolver un error en caso de que el frontal te solicite una página que no exista.
Por ejemplo, si recibes /companies?page=10000000000 probablemente no existan tantas Empresas registradas en la base de datos.

RETO B. Securizando la API Editar (un poco, no mucho...)

Como se comentó anteriormente la API Editar no está securizada. Es decir, cualquiera podría desde Postman editar cualquier Empresa, lo que implica un grave fallo de seguridad.

Para mitigarlo te proponemos lo siguiente:

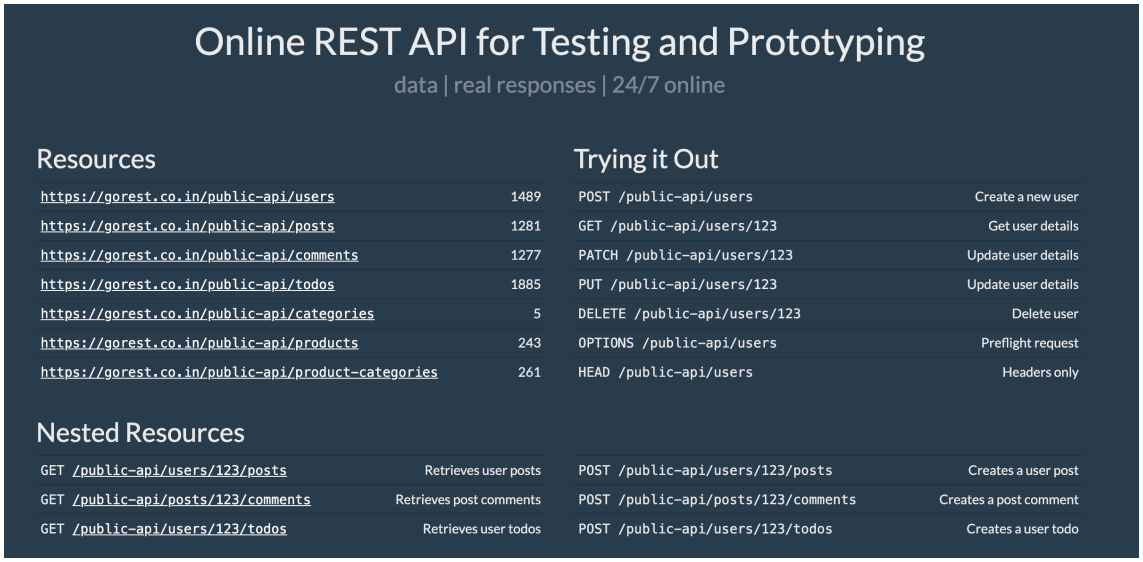
- Añade una nueva columna “token” a la tabla “companies”.
- Al Crear una Empresa genérale un token único.
- El frontal, ahora, te mandará en la petición Editar un Token en la cabecera Authorization.
- Permite la edición de la Empresa si el Token recibido coincide con el Token de la empresa a Editar.

RETO C. Seguridad.

¿Por qué crees que el método anterior puede no ser del todo seguro?

Misión 2. Integrando con la API de Usuarios.

Como se comentaba antes, la empresa **ya tiene creadas** APIs para acceder/editar/listar a sus Usuarios/Dueños (users), a los Posts (posts) sobre sus gatos, y a sus comentarios (comments).



Por cuestiones de seguridad y arquitectura estas APIs **no se pueden consultar directamente desde el frontal de Angular**. Solo pueden ser consultadas desde el backend Node que estás creando.

NOTA.

Se hará uso del servicio gorest.co.in que genera fake data, lee el **Anexo Técnico al final del documento** que explica en detalle como hacer uso de estas APIs.

A. API Pasarela de Usuarios

El primer objetivo es crear una API Pasarela para **listar los Dueños** registrados en la base de datos. Básicamente consiste en crear una API "/owners" que se puede consultar desde el frontal, y cuyo controlador consulta a su vez a la API de usuarios "/users" ya existente.

En este esquema se explica el flujo de peticiones.

Es necesario, por tanto, crear un endpoint "/owners". El controlador de este endpoint hace la consulta a https://gorest.co.in/public-api/users y devuelve al usuario la información que haya recibido.

Una particularidad es que las peticiones a https://gorest.co.in/public-api/users deben realizarse con un Token porque deben ir autenticadas. Por tanto cuando se hagan peticiones a a https://gorest.co.in/public-api/users deberás añadir una cabecera Authentication con un Token válido

[NOTAS: Para conseguir un Token válido debes ir a la página https://gorest.co.in y crearte un usuario. El Token lo puedes encontrar en https://gorest.co.in/user/settings/api-access.html una vez registrado]

NOTA.

La api de https://gorest.co.in/public-api/users es una API paginada que devuelve 20 usuarios/dueños por página, para ello debes pasarle el parámetro "page=X". Para simplificar, el endpoint "/owners" que debes crear deberá ser también paginado con 20 usuarios/dueños por página.

B. Crear una API de Favoritos.

Una nueva funcionalidad que se desea ofrecer a los Dueños es la posibilidad de guardar en Favoritos aquellas Empresas que les resulten más interesantes. Para ello en el frontal se va a implementar un botón “Añadir Empresa como Favorita” en la landing de cada Empresa.

Para implementar esta funcionalidad deberás crear una nueva tabla en la base de datos denominada “favorites” que permita almacenar las relaciones entre los Dueños y las Empresas marcadas como favoritas. Un Dueño puede marcar tantas Empresas favoritas como desee.

El endpoint “favorites” recibirá el id del Dueño y el id de la Empresa que el Dueño ha marcado como favorita. El controlador que implementes deberá comprobar en la tabla “Companies”, que ya creaste en la Misión 1, que el id de la Empresa existe, así mismo deberás comprobar que el id del Dueño existe consultando a la API https://gorest.co.in/public-api/users (mira la Documentación en https://gorest.co.in para ver como podrías consultar la existencia de un ID específico), antes de añadir la relación en la tabla “favorites”.

C. Extrayendo la info del Dueño

Ahora que estás creando el endpoint “/owners” la empresa desea implementar “/owners/:id”(p.e, /owners/4) para obtener la info de un Dueño específico.

La info deseada es su info personal, además de los Post que ha creado en la plataforma.

La respuesta a devolver tendría el siguiente formato:


```
{
   "code": 200,
   "data":{
      "id": 4,
      "name":"Anuraag Reddy",
      "email":"reddy_anuraag@olson-goyette.org",
      "gender":"Female",
      "status":"Active",
      "created_at": "2020-09-17T03:50:03.727+05:30",
      "updated_at":"2020-09-17T03:50:03.727+05:30",
      "posts": [
        {
          "id":4,
          "user_id":4,
          "title":"Arbustum rem amplus(...)",
          "body":"Defessus vel aptus. (...)."
        },
        {
          "id":5,
          "user_id":4,
          "title":"Volubilis armarium(...)",
          "body":"Creta colo videlicet(...)"
        }
      ],
   }
}
```

Retos de la Misión 2

Los Retos comienzan a complicarse. Por lo que tendrás que decidir si quieres saltar a la Misión 3 antes de meterte en estos Retos o por el contrario quieres seguir profundizando en la Misión 2.

RETO A. Endpoint “owners” con paginado múltiple

En la Misión 2 tuviste que crear un endpoint “owners” para listar los dueños consultando a la API “/public_api/users” con la característica de que devolvían ambas 20 resultados por página.

Este Reto consiste en añadir un nuevo parámetro “limit” a “owners” para indicar el número de resultados que debe devolver cada página. Para simplificarlo considera que “limit” solo puede ser múltiplo de 20.

Si no se le pasa el parámetro limit, se entenderá que “limit” es igual a 20.

Es decir:
/owners -> Devolvería los 20 primeros dueños
/owners?page=1 -> Devolvería los 20 primeros dueños

/owners?page=1&limit=40 -> Devolvería los 40 primeros dueños
/owners?page=2&limit=40 -> Devolvería los dueños 41 al 80

/owners?page=1&limit=60 -> Devolvería los 60 primeros dueños

RETO B. Posts y Comentarios

El endpoint “/owners/:id” que creaste durante la Misión devuelve los Posts que el Dueño del gato ha escrito en la red social.

Pero cada Post, a su vez, puede tener asociados Comentarios.

Tu Reto aquí es modificar el endpoint “/owners/:id” para devolver una información mas completa añadiendo la información de Comentarios a cada Post

Mision 3. Securidad y rendimiento

A. Logs de Seguridad

El desarrollador del frontal en Angular es un auténtico máquina y es escrupulosamente exquisito. Esto quiere decir que nunca vas a recibir una petición malformada o que no se atenga a las reglas. Por ejemplo, el endpoint de creación de empresas nunca recibirá una petición a la que le falte el “CIF” (parámetro obligatorio) o el endpoint de búsqueda nunca recibirá una petición con un string de búsqueda menor a 3 caracteres, ya que lo controla perfectamente en el frontal.

Por tanto, vamos a suponer que cualquier petición que no cumpla los criterios/requisitos es un potencial ataque de un Hacker externo.

Tienes que registrar en uno (o múltiples ficheros) las peticiones detectadas como sospechosas, almacenando los datos de la petición así como cualquier otro dato que consideres de interés. Considera como petición sospechosa toda aquella que no cumpla los requisitos que se han ido detallando en las Misiones y Retos **anteriores**.

NOTA.

> En el Read.me deberás añadir el listado de peticiones que has decidido identificar como sospechosas.

B. Rendimiento.

La información de los Owners es bastante cara de obtener (en términos de búsqueda y extracción) a nivel de base de datos. Cada vez que alguien quiere ver el perfil de un Owner hay que hacer múltiples consultas a un servicio externo para generar la respuesta. Se ha detectado que los Owners raramente modifican su perfil una vez se han registrado en la plataforma por lo que la respuesta es casi siempre la misma. ¿Se te ocurre alguna manera para mejorar el rendimiento de la plataforma en estas circunstancias?

Retos de la Misión 3

Estos Retos son, probablemente, los más difíciles de todos. Como te dijimos, no te preocupes si no sabes como atacarlos. Los Retos son optativos.

A. Listado de Empresas limitada.

(Solo si has conseguido resolver el Reto A de la Misión 1). En el Reto A de la Misión 1 se te pedía modificar el Listado de Empresas para que fuese paginada.

Este Listado de Empresas es una API pública, es decir, no es necesario que lleve ningún Token asociado. Se ha decidido, para evitar robo de datos, que solo sean de libre consulta las 10 primeras páginas, pero que el resto de páginas no se muestren salvo que la petición lleve asociada un Token válido (considerando Token válido cualquier Token que aparezca en la tabla “companies”)

De esta forma:

/companies?page=1 (con/sin token) -> Devolvería las 20 primeras empresas.
/companies?page=10 (con/sin token) -> Devolvería las empresas 181 a la 200.

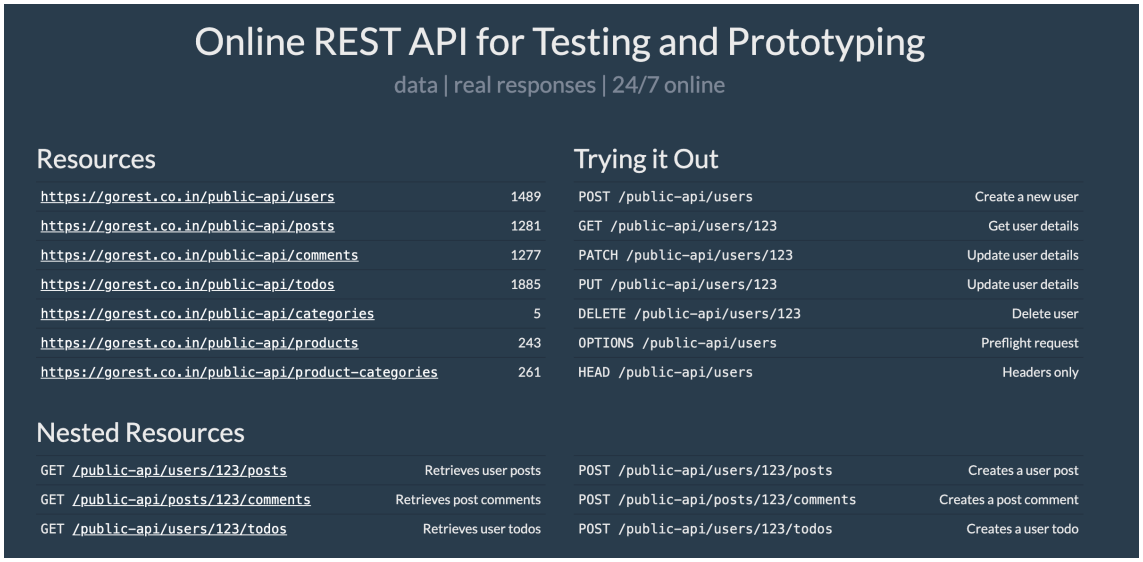
/companies?page=11 (con token válido) -> Devolvería las empresas 201 a la 220.
/companies?page=11 (sin token válido) -> Error. Fallo de Autorización.

B. Detectando y evitando un Hacker.

¿Cómo detectarías un Hacker?, ¿Cómo ignorarías sus peticiones?, ¿Te atreves a implementar un sistema sencillo para demostrar como lo harías?

Anexo Técnico

Para resolver parte de esta Prueba se va a hacer uso de un servicio gratuito llamado GoRest: https://gorest.co.in/



Este servicio está protegido, por lo que necesitas registrarte (gratuito) para conseguir tu token personal (TU-TOKEN) con el que hacer peticiones GET/POST/PATCH.

GoRest sugiere diferentes formas de usar el token, nosotros te recomendamos que a la hora de hacer las peticiones añadas una cabecera “Authorization” con valor “Bearer TU-TOKEN”.

Te recomendamos que leas detenidamente la página principal
https://gorest.co.in/

sobre todo las secciones “Trying it Out” y “Authentication”.