REQUERIMIENTO

La empresa Naiofy está finalizando el desarrollo de un sistema para la venta de álbumes online. Dentro de las funcionalidades que posee el sistema desarrollado podemos encontrar los siguientes endpoints:

- Registro
- Login.
- Listado de usuarios.
- Listado de álbumes.
- Listado de fotos de un álbum.
- Listado de álbumes comprados.
- Comprar álbum.
- Invalidar sesiones

El sistema posee 2 tipos de usuarios: *regulares* y *administradores*. La diferencia entre estos usuarios radica en que los administradores podrán obtener, del endpoint de listado, todos los usuarios de la aplicación independiente de su rol, mientras que los regulares sólo podrán ver usuarios de su mismo rol. Asimismo, los usuarios regulares no podrán invalidar las sesiones del resto de los usuarios.

Se desea automatizar, para verificar su constante y correcto funcionamiento, todos los posibles y relevantes flujos y escenarios del sistema.

Reporte

- Se requiere realizar un reporte del resultado de la corrida de las pruebas automatizadas, contemplando casos de éxitos o de error.

SOLUCIÓN

Alojar la solución en un repositorio público de github, utilizando la tecnología que considere y realizar una lista de mejoras/sugerencias a informar a los desarrolladores de Naiofy para una mejor implementación de la automatización.

La solución deberá estar lista para el día **23/04/2021**.

ANEXO

Datos API

La API del sistema se encuentra deployada y lista para iniciar las pruebas.

Ambiente

URL: https://nodejs-qa-training.herokuapp.com/

Credenciales

Admin

- User: *admin@wolox.com.ar*
- Pass: *candidatoWolox2020*

Regular

- User: *regular@wolox.com.ar*
- Pass: *candidatoWolox2020*

Endpoints

Registro

Endpoint para la creación de un usuario regular dentro de la aplicación.

POST {{URL}}/users

Request body

```
{
    "email": <newUserMail>,
    "password": <newUserPassword>,
    "firstName": <newUserFirstName>,
    "lastName": <newUserLastName>
}
```

Criterios de aceptación

- El status code esperado al registrar un usuario es: **201**.
- Los mails admitidos para el registro serán solo aquellos con el dominio de wolox
    - @wolox.com.ar
- No se admitirán números en el *first_name* ni en el *last_name*

- Al introducir varios errores, en la respuesta deberán informarse todos juntos y no de a uno.

Login

Endpoint para el login dentro de la aplicación

POST {{URL}}/users/sessions

Request body

```
{
    "email": <newUserMail>,
    "password": <newUserPassword>
}
```

Criterios de aceptación

- El status code retornado será **200**
- En el header retornado se deberá contar con el atributo *Authorization* el cual no podrá ser nulo ni vacío.
- El cuerpo de la respuesta deberá ser acorde al formato especificado.
- El endpoint devolverá el id del usuario logueado en el atributo *user_id*.

Listado de usuarios

Endpoint para obtener un listado de los usuarios registrados en la aplicación

GET {{URL}}/users

Request headers

Authorization: <tokenObtained> ***necessary***

Criterios de aceptación

- Sólo admitirá el token obtenido en el login.
- El status code retornado será **200**
- Cada uno de los usuarios retornados deberán tener el formato especificado.
- Se devolverán usuarios de ambos tipos: admin y/o regulares (según el usuario que ejecute la prueba).

Listado de álbumes

Endpoint para obtener un listado de los álbumes creados.

GET {{URL}}/albums

Request headers

Authorization: <tokenObtained> ***necessary***

Criterios de aceptación

- Sólo admitirá el token obtenido en el login.
- El status code retornado será **200**
- Cada uno de los usuarios retornados deberán tener el formato especificado.
- Se devolverán todos los álbumes disponibles sin paginar.

Listado de fotos de un álbum

Endpoint para obtener las fotos de un álbum.

GET {{URL}}/albums/{{idAlbum}}/photos

Request headers

Authorization: <tokenObtained> ***necessary***

Request body

{}

Criterios de aceptación

- Sólo admitirá el token obtenido en el login.
- El status code retornado será **200**
- Cada una de las fotos obtenidas deberá tener el formato especificado.

Listado de álbumes comprados

Endpoint para obtener los álbumes comprados por un usuario

GET {{URL}}/users/{{userIdLogged}}/albums

Request headers

Authorization: <tokenObtained> ***necessary***

Request body

{}

Criterios de aceptación

- Sólo admitirá el token obtenido en el login.
- El status code retornado será **200**
- Cada uno de los álbumes retornados deberán tener el formato especificado.
- El endpoint deberá retornar todos los álbumes comprados por el usuario especificado.

Comprar álbum

Endpoint para comprar un álbum.

POST {{URL}}/albums/{{idAlbum}}

Request headers

Authorization: <tokenObtained> ***necessary***

Request body

{}

Criterios de aceptación

- Sólo admitirá el token obtenido en el login.
- El status code retornado será **201**
- No permitirá la compra de un álbum más de una vez por usuario.

Invalidar sesiones

Endpoint para comprar un álbum.

POST {{URL}}/users/sessions/invalidate_all

Request headers

Authorization: <tokenObtained1> ***necessary***

Request body

{}

Criterios de aceptación

- Sólo admitirá el token obtenido en el login.
- El status code retornado será **200**

Formatos

El formato de los JSON que deberán retornar cada uno de los endpoints se encuentran especificados con la referencia al tipo de dato que deberían retornar e indicando si es obligatoria su presencia o no.

Referencias (ejemplos):

- *#number*, indica que se espera un valor numérico obligatoriamente
- *#string*, indica que se espera un valor string obligatoriamente.
- *#notnull*, indica que se espera un valor distinto a null obligatoriamente.
- *##string*, indica que se espera un valor string opcionalmente. (aplicable a cualquier otro tipo de dato).
- *#boolean*, indica que se espera un valor booleano obligatoriamente.
- *#? _ == "admin" || _ == "regular"*, indica que se espera uno de estos valores: “admin” o “regular”.
- *regular*, indica que se espera el valor “regular”.

Login

```
{
 "user_id": "#number"
}
```

Pagination

```
{
 "page": “[] schema”,
 "count": "#number",
 "limit": "#number",
 "offset": "#number",
 "total_pages": "#number",
 "total_count": "#number",
 "previous_page": "#number",
 "current_page": "#number",
 "next_page": "#number",
 "previous_page_link": "##string",
 "current_page_link": "##string",
 "next_page_link": "##string",
 "first_page_link": "##string",
 "last_page_link": "##string",
 "n_page_link": "##string"
}
```

IMPORTANTE: el atributo page deberá ser un array de un objeto en particular especificado por cada una de las entidades que se están consultando. Es decir que si se consultan usuarios, dentro de page, vendrá un array de objetos con el formato de usuarios.

Users

```
{
 "page": [
   {
    "first_name": "#string",
    "last_name": "#string",
    "email": "#string",
    "role": “#string”, // Valores posibles: "admin" || "regular"’
   }
 ],
 "count": "#number",
 "limit": "#number",
 "offset": "#number",
 "total_pages": "#number",
 "total_count": "#number",
 "previous_page": "#number",
 "current_page": "#number",
 "next_page": "#number",
 "previous_page_link": "##string",
 "current_page_link": "##string",
 "next_page_link": "##string",
 "first_page_link": "##string",
 "last_page_link": "##string",
 "n_page_link": "##string"
}
```

Albums

```
{
 "user_id": "#number",
 "id": "#number",
 "title": "#string"
}
```

Photos

```
{
 "album_id": "number",
 "id": "#number",
 "title": "#string",
 "url": "#string",
 "thumbnail_url": "#string"
}
```

WOLOX

Purchased Album


```
{
 "user_id": "#number",
 "album": {
   "user_id": "#number",
   "id": "#number",
   "title": "#number"
 },
 "created_at": "#string"
}
```


Errors

Formato que se espera recibir siempre que haya un error en alguna solicitud.


```
{
 "errors": [
   { "code": "##string", "message": "#notnull" }
 ],
 "origin": "##string",
 "stack_trace": "##string",
 "timestamp": "#string"
}
```