* **What is the advantage (if any) for a REST-based API of using JWT’s compared to session Cookies**

JSON Web Tokens are stateless unlike cookies (meaning they don’t create or use a session), meaning they’re more lightweight and therefore friendly for many different types of devices (computers, phones etc.). They can also last as long as you want them to by specifying expire time information and used with multiple backends.

They also contain all relevant authentication information (and any requested information when sent as a response) which is encoded, and will always be verified on the server when sent along with a request, meaning you will never be able to access or retrieve anything from the server if your token is invalid (that is, it doesn’t match exactly with what the server has stored).

On top of this, it can be used on the client-side to restrict access to certain areas of an application (it should be stored on the client’s device upon first logging in).

* **What is the disadvantage (if any) with the implemented JWT-solution**

One disadvantage is that since the token can last for a very long time (depending on expire time), if a user changes their authentication information, like password or email, after already logging in, the token generated from the previous authentication will still be valid until expiry.

There are (likely) ways around this, but by default it works like that, which could potentially be a security risk.

* **What will a client (Single Page WEB, Mobile App, etc.) have to do in order to use this API**

Every request sent will need the “x-access-token” HTTP header on it (and the actual JWT token itself as the value), containing authentication information along with any other data the request needs to send to the server. If not present, you won’t be able to access any of the endpoints that require you to be logged in.

In short: Provide the JWT generated upon authentication when logging in every time a request is sent via an “x-access-token” header.