Acteurs :

* ***Resource owner*** (l’utilisateur de l’application Datavatar)
* ***Client*** (server API REST Datavatar)
* ***API Nokia***

1. Le ***resource owner*** demande au ***client***d’initier une autorisation d’accès à son compte Nokia. Pour ce faire en utilisant le service REST du ***client***

@GET

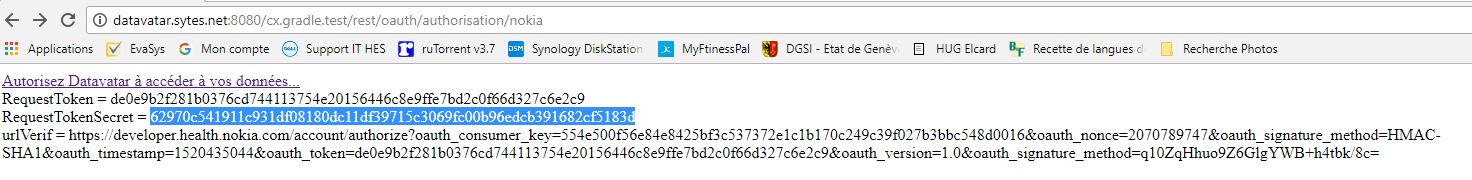
@PATH *URL\_BASE*/nokia/autorisation *(à modifier dans l’API)*

@PRODUCES (MediaType.APPLICATION\_JSON)

public **Oauth1Authorisation** **authorisation**(){

}

1. Le ***client*** demande à l’***API Nokia***un ***RequestTokenKey*** et un ***RequestTokenSecret***, en lui donnant son ***ConsumerKey*** (attribué au client) et en signant l’URL avec son ***ConsumerKey*** (connu uniquement par le *client*).
2. L’*API Nokia* retourne les ***RequestTokenKey*** et un ***RequestTokenSecret*** au client
3. Le ***client*** retourne au ***resource owner*** (en réponse au *GET* initié en *1.*) l’URL de vérification générée/signée, le ***RequestTokenKey*** ainsi que le ***RequestTokenSecret*** (en version cryptée, le ***client*** étant le seul pouvant les décrypter par la suite, mais ne conserve aucune information du ***resource owner***, ce dernier devra donc conserver ses informations et les retourner quand cela sera nécessaire). Le tout dans un *JSON*. *(À modifier dans le retour et le content)*



*On voit après l’appel du resource owner, l’URL de vérification ainsi que les données du request token (ici non cryptées).*

1. Le ***resource owner*** sauvegarde le ***RequestTokenKey*** et le ***RequestTokenSecret*** cryptés dans sa BDD. Il ouvre l’URL de vérification reçu dans un browser (attention on ne peut pas l’ouvrir dans n’importe quel vue, exemple pour *Android* -> *CustomsTabs*)

<https://developer.health.nokia.com/account/authorize?oauth_consumer_key=554e500f56e84e8425bf3c537372e1c1b170c249c39f027b3bbc548d0016&oauth_nonce=2070789747&oauth_signature_method=HMAC-SHA1&oauth_timestamp=1520435044&oauth_token=de0e9b2f281b0376cd744113754e20156446c8e9ffe7bd2c0f66d327c6e2c9&oauth_version=1.0&oauth_signature_method=q10ZqHhuo9Z6GlgYWB+h4tbk/8c>=

|  |  |
| --- | --- |
|  | Le ***resource owner*** autorise le ***client*** à accéder à son compte (une fenêtre d’authentification de l’utilisateur est fait au préalable) |

1. L’***Api Nokia*** redirige le ***resource owner*** sur l’adresse de callback du ***client*** avec les paramètres retournés.

[/nokia/oauth1a/verifier?userid=15492984&oauth\_token=de0e9b2f281b0376cd744113754e20156446c8e9ffe7bd2c0f66d327c6e2c9&oauth\_verifier=MAR18FIc5N6PxR4prPfK](http://datavatar.sytes.net:8080/cx.gradle.test/rest/nokia/oauth1a/verifier?userid=15492984&oauth_token=de0e9b2f281b0376cd744113754e20156446c8e9ffe7bd2c0f66d327c6e2c9&oauth_verifier=MAR18FIc5N6PxR4prPfK)

@GET

@PATH *URL\_BASE*/nokia/verifier

@CONSUMES ("application/x-www-form-urlencoded

@PRODUCES (MediaType.APPLICATION\_JSON)

Public String verifier (@QueryParam("oauth\_verifier") String verifier){

}

1. Afin de pouvoir demander à l’***API Nokia*** l’*AccessToken* et l’*AccessTokenSecret,* le ***client*** a besoin en plus du ***Verifier*** du ***RequestTokenKey*** et du ***RequestTokenSecret*** qu’il avait reçu en 2, cryptés et envoyer au ***resource owner***. Il va donc envoyer le ***Verifier*** ***(éventuellement crypté ???)*** au ***resource owner*** qui lui retournera avec le ***RequestTokenKey*** et le ***RequestTokenSecret.***
2. Le ***resource owner*** renvoi au ***client*** les données nécessaires (***RequestTokenKey***, ***RequestTokenSecret*** et le ***Verifier***).

@GET

@PATH *URL\_BASE*/nokia/oauth1a/tokenaccess

@PRODUCES (MediaType.APPLICATION\_JSON)

Public String tokenAccess (@QueryParam("oauth\_verifier") String verifier){

}

1. blabla