PROJET BAN
COMPTE-RENDU

Objet de la réunion :
Journée BAN Team à St Mandé
IGN / La Poste / Etalab

Date 20/11/2015

Participants :
ETALAB : Christian, Yohann.
IGN : Elodie, Mélanie, Olivier.
LA POSTE : Isabelle, Christophe, Philippe, Brice.

Ordre du jour :

- Lancement du co-développement des lots API et modélisation
- Prise en main du proto d'API écrit par EtaLab

Principales conclusions :

- L'équipe projet (EtaLab, LP et l'IGN) a pu prendre en main l'API développée par EtaLab. Chaque participant a pu :
  - Ajouter une fonction dans l'API pour importer une partie des données du RAN dans la BAN (voies HEXAVIA du dépt 33).
  - Tester quelques routes de l'API (consultation, écriture, premiers tests sur la sécurité)
- L'équipe projet va continuer à développer/tester cette API. L'objectif est d'aboutir à une version plus avancée de l'API pour mi-janvier, branchée sur une base test de la BAN (par exemple obtenue par montée en base des données ban actuelle). Cette instance serait accessible à quelques utilisateurs en test.
- En parallèle, pour consolider le modèle et l'API, les lots suivants doivent démarrer :

    - Test de branchement du guichet adresse (action IGN)
    - Test de branchement du SI IGN (action IGN)
    - Test de branchement du SI La Poste (action La Poste)
- Le lot Import massif / appariement doit aussi démarrer (action Equipe projet)
- La Poste et l'IGN rédigent un document sur les aspects sécurité dans l'API et le soumettront à leur expert sécurité respectif. Il faudra inclure dans ce document les différents types d'utilisateurs ainsi qu'une proposition pour leur droit d'accès sur les données.
- L'avancement des différents travaux sera présenté à l'OpenLab du 15/12. Une quinzaine d'utilisateurs sont prévus, dont des SDIS. Recherche d'une salle à ETALAB.
- Un point audio est prévu tous les lundis à 10 h (EtaLab, LP et IGN)

Installation :

L'API et son environnement avaient été installés sur les différents PC windows par Brice et Melanie en début de semaine pour gagner du temps (installation de VirtualEnv pour python, PostgreSQL, clone de l'API sur github …)

Quelques éléments sur le modèle de données proposé:
Le modèle contient les tables principales suivantes (en anglais) :

- Municipality
- Street
- Housenumber (l'adresse avec son numéro)
- Locality (0 ou n localisation de l'adresse)

Le MCD a été envoyé par Christophe mardi matin

Métadonnées : chaque objet possède une date de création et de modification ainsi qu'un lien vers l'utilisateur qui l'a créé/modifié (lien vers quelle table ?).

Version : Chaque objet possède un numéro de version (1 pour sa création, 2 pour la version suivante …). Les différentes versions d'un même objet sont stockées dans la table version (une seule table pour toutes les classes d'objets). La table Version contient les colonnes suivantes :

- Identifiant de l'objet
- Classe de l'objet
- Numéro de version

- L'objet en binaire (accessible en python)

Différentiel : une autre table Diff contient toutes les différences entre 2 états successifs d'un objet

Quelques éléments sur l'API :

Voici quelques routes utiles :

Type de route	Route	Résultat
consultation	http://localhost:5959/street/22088	La voie d'identifiant 22088 en json
consultation	http://localhost:5959/municipality/insee:33001	La commune d'insee 33001 en json
consultation	http://localhost:5959/municipality/insee:33001/versions	Toutes les versions de la commune d'insee 33001 en json
consultation	http://localhost:5959/municipality/insee:33001/versions/1	La version 1 de la commune d'insee 33001 en json
Ecriture (étape 1 : demande d'un token)	Ban auth::dummytoken	Renvoi un token pour l'utilisateur (« token » pour l'instant)
Ecriture	http -f POST http://localhost:5959/municipality Authorization:"Bearer token" name=jard insee=85000 siren=10 version=1	Créé la commune de jard Pour l'instant, il faut passer tous les champs
Ecriture	http -f POST http://localhost:5959/municipality/insee:85000 Authorization:"Bearer token" name=jard insee=85000 siren=11 version=2	Modifie le SIREN de la commune de jard

L'API gère les conflits : si on essaye de modifier une commune en mettant un numéro de version déjà stockée et pas le n+1→ l'outil n'effectue pas la mise à jour et indique qu'il y a un conflit.

Aspect sécurité :

Les réflexions commencent sur ce point. Pour l'instant il est proposé que :

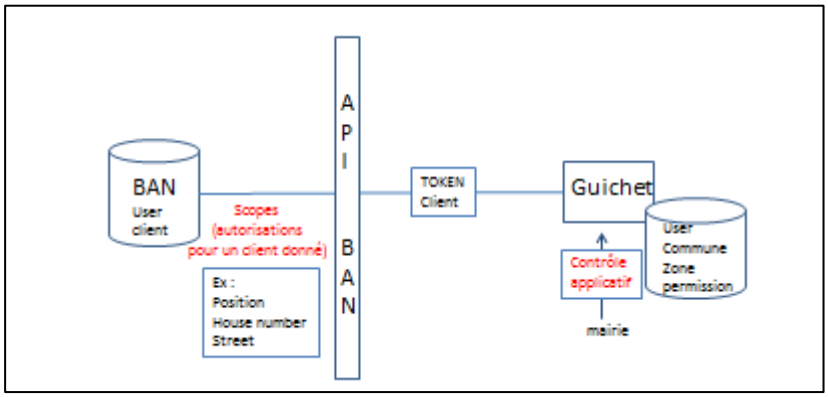
- l'API gère l'authentification et les autorisations par applicatif et pas pour chaque utilisateur de chaque applicatif. L'applicatif commencera par demander un token à l'API. Ce token sera ensuite utilisé par l'applicatif pour communiquer avec l'API.
- Au moment de son appel à l'API, l'applicatif pourra fournir des infos supplémentaires sur l'utilisateur (de l'applicatif) à stocker dans les métadonnées de la BAN
- L'API gère uniquement quelques profils d'autorisations (scopes) qui s'appliqueraient plutôt sur des classes entières et qui ne descendraient pas au niveau attributaire:

- Lecture seule (par défaut pour tout le monde)
- Ecriture sur la classe localisation ???
- Ecriture sur la classe localisation, adresse, voie ???
- Admin

  -> Ecrire un document pour préciser les applicatifs concernés par ces scopes, et les particularités d'accès

- Les accès par zonages géographiques ne seraient pas gérés par l'API mais par l'applicatif appelant l'API (par exemple le guichet adresse)

Schéma général dessiné par Yohan Boniface:



L'ensemble de ces propositions est à valider rapidement pour spécifier l'API sur 3 plans:

- au niveau sécurité pure en la soumettant à des « experts sécurité »
- en la soumettant aux différents applicatifs à brancher sur la BAN (guichet, SI IGN, SI Poste …) et à des cas d'utilisation réelle
- enfin en listant pour chaque type d'utilisateur les droits en lecture/écriture sur les données réellement souhaitées et en regardant les différents problèmes que cela peut entrainer sur la solution proposée : par exemple, un accès en lecture seule sur le champ « complément d'adressage » pourrait ne pas être compatible avec un accès en écriture sur la classe « Localisation » …