Nanterre

Initialisation

- Définir le fichier setting

```
export ZOO_SETTINGS_FILE=`pwd`/local_settings.py
```

- Créer les tables

```
./manage.py migrate
```

- Charger les schémas

```
./manage.py loaddata rsu.json
```

- Créer une base temporaire contenant les dumps[1] du RSUv1 et de SWARM (IdP OAuth2 du RSUv1):

```
createdb rsuv1
psql rsuv1 <nrsu.sql
psql rsuv1 <swarm_nanterre.sql
```

- Charger les dumps dans zoo et produire une fixture pour créer les utilisateurs dans authentic

```
./manage.py rsu-load-dump "dbname=nanterre_rsu" authentic_users.json
```

- Charger les utilisateurs dans authentic

```
authentic2-multitenant-manage tenant_command loaddata -d connexion-moncompte.nanterre.fr authe
ntic_users.json
```

Attention à ce que le fichier authentic_users.json soit dans un répertoire accessible à l'utilisateur authentic-multitenant

1 https://files.entrouvert.org/Clients%20et%20contacts/3%20-%20Clients%20en%20cours/Nanterre/inputs/dump%20du%2020170210/ dernier dump de la preprod (je crois), faut certainement redemander un dump récent

Dédoublonnage

Mise en place

- Définir un seuil de dédoublonnage, disons 80%
- Initialiser la base de dédoublonnage

```
./manage.py rsu-duplicates find --limit 0.8
```

- Mettre en place une mise à jour journalière à partir des derniers individus créés les deux derniers jours:

```
./manage.py rsu-duplicates find --limit 0.8 --days 2
```

À lancer dans un cron régulier

Gestion

- Vérifier les doublons

```
./manage.py rsu-duplicates list
```

- Voir les faux-positifs

```
./manage.py rsu-duplicates list --false-positive
```

- Voir les doublons dédupliqués

```
./manage.py rsu-duplicates list --dedup
```

- Supprimer les doublons en dessous d'un seuil (parce qu'on est descendu trop bas en testant par exemple)

```
./manage.py rsu-duplicates delete --limit 0.8
```

Zoo - stockage de documents et de relations

- Modèle de donnée
- API

Historique

Spécifications (obsolètes)

API

Créer une entité

Paramètres d'URL

Nom	Indication
unflatten	Reconstruire des structures ("xx__yy__zz": 1 donnera {'xx': {'yy': {'zz': 1}}})

```
POST /api/entity/<slug>/?unflatten=1 HTTP/1.1
Content-Type: application/json

{
  "champ1": <valeur1>,
  "champ2": <valeur2>,
  "journal__backoffice_url": "<journa_valeur2>,
}
```

Cas passant

L'entité est créée, un identifiant lui est attribué et est renvoyé dans le champ id, les attributs journal__ sont stockées comme une entrée de journal associée à l'entité enlevant le préfixe journal__.

```
201 Created
Content-Type: application/json

{
   "err": 0,
   "id": 1234,
   "data": {
       "champ1": <valeur1>,
   }
}
```

Cas non-passant

Slug d'entité inexistant

```
404 Not found
Content-Type: application/json

{"err": 1}
```

Schéma invalide

```
404 Not found
Content-Type: application/json

{"err": 1, "errors": ["clé x manquant", "clé y invalide"]}
```

Modifier une entité

On peut soit écraser le contenu actuel avec PUT, soit mettre à jour l'existant avec PATCH. Comme pour la création les clés avec le préfixe journal__ servent à alimenter l'historique et le paramètre unflatten indique de renormaliser les données.

Requête

```
PUT /api/entity/1234/?unflatten=1 HTTP/1.1
Content-Type: application/json

{
   "champ1__0__champ2": 1,
```

```
   "champ1__0__champ3": 2,
   "journal__backofice_url": "https://..."
}
```

Cas passant

```
200 Ok
Content-Type: application/json

{
   "err": 0,
   "data": {
      "champ1": [
         {
            "champ2": 1,
            "champ3": 2
         }
      ]
   }
}
```

Cas non-passant

- slug d'entité inexistant
- schéma invalide (voir plus haut)

Modèle de donnée

Méta-données

EntitySchema

Ce modèle stocke le schéma JSON, le nom et le slug pour un type de donnée

RelationSchema

Ce modèle stocke le schéma JSON, le nom, le slug, les types de données reliées pour une relation, ainsi qu'un booléen indiquant si la relation est symétrique ou pas (l'ordre des données reliées n'a donc pas d'importance)

Données

Entity

Ce modèle stocke les données sous forme de documents JSON, éventuellement associés à une transaction de création, de modification ou de suppression ainsi qu'un document JSON de métadonnées.

Relation

Ce modèle indique la relation entre deux entités.

Log

Ce modèle stocke un document JSON associé à un horodatage et une entité, servant d'historique des actions sur cette entité.

Transaction

Ce modèle stocke les données d'un appel de web-service, requêtes et réponses.

Zoo doit permettre de stocker toute sorte de données non structurées comme des personnes, des adresses, des entreprise ou des associations et les liens qui les unissent. Il doit permettre de maintenir une certaine intégrité relationnelle, par exemple s'il est interdit d'avoir une adresse sans qu'aucune personne n'y habite on s'assurera qu'après chaque modification la contrainte est maintenue.

Schéma SQL pour le stockage de Zoo

Avant toute modification aux données on fera un LOCK zoo_data_transaction pour sérialiser strictement les transactions.

Schéma des métadonnées:

```
--
-- Create model EntitySchema
--
CREATE TABLE "zoo_meta_entityschema" (
    "id" serial NOT NULL PRIMARY KEY,
    "name" varchar(64) NOT NULL UNIQUE,
    "slug" varchar(64) NOT NULL UNIQUE,
    "schema" jsonb NOT NULL,
    "caption_template" text NOT NULL
);
--
-- Create model RelationSchema
--
CREATE TABLE "zoo_meta_relationschema" (
    "id" serial NOT NULL PRIMARY KEY,
    "name" varchar(64) NOT NULL UNIQUE,
    "slug" varchar(64) NOT NULL UNIQUE,
    "schema" jsonb NOT NULL,
    "caption_template" text NOT NULL,
    "is_symmetric" boolean NOT NULL,
    "left_id" integer NOT NULL,
    "right_id" integer NOT NULL
);
CREATE INDEX "zoo_meta_entityschema_name_522bdb58_like" ON "zoo_meta_entityschema" ("name"
 varchar_pattern_ops);
CREATE INDEX "zoo_meta_entityschema_slug_56477a00_like" ON "zoo_meta_entityschema" ("slug"
 varchar_pattern_ops);
ALTER TABLE "zoo_meta_relationschema"
  ADD CONSTRAINT "zoo_meta_relations_left_id_d3422f90_fk_zoo_meta_entityschema_id"
  FOREIGN KEY ("left_id")
  REFERENCES "zoo_meta_entityschema" ("id")
  DEFERRABLE INITIALLY DEFERRED;
ALTER TABLE "zoo_meta_relationschema"
  ADD CONSTRAINT "zoo_meta_relation_right_id_6d6e7138_fk_zoo_meta_entityschema_id"
  FOREIGN KEY ("right_id")
  REFERENCES "zoo_meta_entityschema" ("id")
  DEFERRABLE INITIALLY DEFERRED;
CREATE INDEX "zoo_meta_relationschema_42bb7fcb" ON "zoo_meta_relationschema" ("left_id");
CREATE INDEX "zoo_meta_relationschema_4b6b95be" ON "zoo_meta_relationschema" ("right_id");
CREATE INDEX "zoo_meta_relationschema_name_73c7f5d0_like" ON "zoo_meta_relationschema" ("name"
 varchar_pattern_ops);
CREATE INDEX "zoo_meta_relationschema_slug_42bf92c1_like" ON "zoo_meta_relationschema" ("slug"
 varchar_pattern_ops);
```

Schéma des données:

```
--
-- Create model Entity
--
CREATE TABLE "zoo_data_entity" (
    "id" serial NOT NULL PRIMARY KEY,
    "meta" jsonb NULL,
    "content" jsonb NOT NULL
);
```

```
--
-- Create model Relation
--
CREATE TABLE "zoo_data_relation" (
    "id" serial NOT NULL PRIMARY KEY,
    "meta" jsonb NULL,
    "content" jsonb NOT NULL
);
--
-- Create model Transaction
--
CREATE TABLE "zoo_data_transaction" (
    "id" serial NOT NULL PRIMARY KEY,
    "created" timestamp with time zone NOT NULL,
    "meta" jsonb NULL, "content" jsonb NULL,
    "failed" boolean NOT NULL,
    "result" jsonb NULL
);
--
-- Add field created to relation
--
ALTER TABLE "zoo_data_relation" ADD COLUMN "created_id" integer NULL;
ALTER TABLE "zoo_data_relation" ALTER COLUMN "created_id" DROP DEFAULT;
--
-- Add field deleted to relation
--
ALTER TABLE "zoo_data_relation" ADD COLUMN "deleted_id" integer NULL;
ALTER TABLE "zoo_data_relation" ALTER COLUMN "deleted_id" DROP DEFAULT;
--
-- Add field left to relation
--
ALTER TABLE "zoo_data_relation" ADD COLUMN "left_id" integer NOT NULL;
ALTER TABLE "zoo_data_relation" ALTER COLUMN "left_id" DROP DEFAULT;
--
-- Add field modified to relation
--
ALTER TABLE "zoo_data_relation" ADD COLUMN "modified_id" integer NULL;
ALTER TABLE "zoo_data_relation" ALTER COLUMN "modified_id" DROP DEFAULT;
--
-- Add field right to relation
--
ALTER TABLE "zoo_data_relation" ADD COLUMN "right_id" integer NOT NULL;
ALTER TABLE "zoo_data_relation" ALTER COLUMN "right_id" DROP DEFAULT;
--
-- Add field schema to relation
--
ALTER TABLE "zoo_data_relation" ADD COLUMN "schema_id" integer NOT NULL;
ALTER TABLE "zoo_data_relation" ALTER COLUMN "schema_id" DROP DEFAULT;
--
-- Add field created to entity
--
ALTER TABLE "zoo_data_entity" ADD COLUMN "created_id" integer NULL;
ALTER TABLE "zoo_data_entity" ALTER COLUMN "created_id" DROP DEFAULT;
--
-- Add field deleted to entity
--
ALTER TABLE "zoo_data_entity" ADD COLUMN "deleted_id" integer NULL;
ALTER TABLE "zoo_data_entity" ALTER COLUMN "deleted_id" DROP DEFAULT;
--
-- Add field modified to entity
--
ALTER TABLE "zoo_data_entity" ADD COLUMN "modified_id" integer NULL;
ALTER TABLE "zoo_data_entity" ALTER COLUMN "modified_id" DROP DEFAULT;
--
-- Add field schema to entity
--
ALTER TABLE "zoo_data_entity" ADD COLUMN "schema_id" integer NOT NULL;
```

```
ALTER TABLE "zoo_data_entity" ALTER COLUMN "schema_id" DROP DEFAULT;
CREATE INDEX "zoo_data_relation_46209121" ON "zoo_data_relation" ("created_id");
ALTER TABLE "zoo_data_relation" ADD CONSTRAINT "
zoo_data_relatio_created_id_4ad84537_fk_zoo_data_transaction_id" FOREIGN KEY ("created_id")
REFERENCES "zoo_data_transaction" ("id") DEFERRABLE INITIALLY DEFERRED;
CREATE INDEX "zoo_data_relation_9225a33e" ON "zoo_data_relation" ("deleted_id");
ALTER TABLE "zoo_data_relation" ADD CONSTRAINT "
zoo_data_relatio_deleted_id_910b58aa_fk_zoo_data_transaction_id" FOREIGN KEY ("deleted_id")
REFERENCES "zoo_data_transaction" ("id") DEFERRABLE INITIALLY DEFERRED;
CREATE INDEX "zoo_data_relation_42bb7fcb" ON "zoo_data_relation" ("left_id");
ALTER TABLE "zoo_data_relation" ADD CONSTRAINT "
zoo_data_relation_left_id_148c2f0a_fk_zoo_data_entity_id" FOREIGN KEY ("left_id") REFERENCES "
zoo_data_entity" ("id") DEFERRABLE INITIALLY DEFERRED;
CREATE INDEX "zoo_data_relation_13d4ad79" ON "zoo_data_relation" ("modified_id");
ALTER TABLE "zoo_data_relation" ADD CONSTRAINT "
zoo_data_relati_modified_id_156ac57d_fk_zoo_data_transaction_id" FOREIGN KEY ("modified_id")
REFERENCES "zoo_data_transaction" ("id") DEFERRABLE INITIALLY DEFERRED;
CREATE INDEX "zoo_data_relation_4b6b95be" ON "zoo_data_relation" ("right_id");
ALTER TABLE "zoo_data_relation" ADD CONSTRAINT "
zoo_data_relation_right_id_a5cc1faf_fk_zoo_data_entity_id" FOREIGN KEY ("right_id") REFERENCES "
zoo_data_entity" ("id") DEFERRABLE INITIALLY DEFERRED;
CREATE INDEX "zoo_data_relation_80cf15b5" ON "zoo_data_relation" ("schema_id");
ALTER TABLE "zoo_data_relation" ADD CONSTRAINT "
zoo_data_relat_schema_id_aee8488b_fk_zoo_meta_relationschema_id" FOREIGN KEY ("schema_id")
REFERENCES "zoo_meta_relationschema" ("id") DEFERRABLE INITIALLY DEFERRED;
CREATE INDEX "zoo_data_entity_46209121" ON "zoo_data_entity" ("created_id");
ALTER TABLE "zoo_data_entity" ADD CONSTRAINT "
zoo_data_entity_created_id_e3928a2a_fk_zoo_data_transaction_id" FOREIGN KEY ("created_id")
REFERENCES "zoo_data_transaction" ("id") DEFERRABLE INITIALLY DEFERRED;
CREATE INDEX "zoo_data_entity_9225a33e" ON "zoo_data_entity" ("deleted_id");
ALTER TABLE "zoo_data_entity" ADD CONSTRAINT "
zoo_data_entity_deleted_id_d18c1b20_fk_zoo_data_transaction_id" FOREIGN KEY ("deleted_id")
REFERENCES "zoo_data_transaction" ("id") DEFERRABLE INITIALLY DEFERRED;
CREATE INDEX "zoo_data_entity_13d4ad79" ON "zoo_data_entity" ("modified_id");
ALTER TABLE "zoo_data_entity" ADD CONSTRAINT "
zoo_data_entity_modified_id_9fad10fc_fk_zoo_data_transaction_id" FOREIGN KEY ("modified_id")
REFERENCES "zoo_data_transaction" ("id") DEFERRABLE INITIALLY DEFERRED;
CREATE INDEX "zoo_data_entity_80cf15b5" ON "zoo_data_entity" ("schema_id");
ALTER TABLE "zoo_data_entity" ADD CONSTRAINT "
zoo_data_entity_schema_id_7071fe89_fk_zoo_meta_entityschema_id" FOREIGN KEY ("schema_id")
REFERENCES "zoo_meta_entityschema" ("id") DEFERRABLE INITIALLY DEFERRED;

--
-- Create model Log
--
CREATE TABLE "zoo_data_log" ("id" serial NOT NULL PRIMARY KEY, "timestamp" timestamp with time zon
e NOT NULL, "content" jsonb NULL, "url" varchar(200) NULL, "entity_id" integer NOT NULL, "transact
ion_id" integer NOT NULL);
ALTER TABLE "zoo_data_log" ADD CONSTRAINT "zoo_data_log_entity_id_a7395203_fk_zoo_data_entity_id"
FOREIGN KEY ("entity_id") REFERENCES "zoo_data_entity" ("id") DEFERRABLE INITIALLY DEFERRED;
ALTER TABLE "zoo_data_log" ADD CONSTRAINT "
zoo_data_log_transaction_id_0ba87eff_fk_zoo_data_transaction_id" FOREIGN KEY ("transaction_id")
REFERENCES "zoo_data_transaction" ("id") DEFERRABLE INITIALLY DEFERRED;
CREATE INDEX "zoo_data_log_d7e6d55b" ON "zoo_data_log" ("timestamp");
CREATE INDEX "zoo_data_log_dffc4713" ON "zoo_data_log" ("entity_id");
CREATE INDEX "zoo_data_log_f847de52" ON "zoo_data_log" ("transaction_id");
```

Web services proposés

Route	Résultat
GET /zoo/relations/	liste des des définitions des relations
GET /zoo/entities/	liste des définitions des entités
GET /zoo/entities/search/?query	recherche des entités
POST /zoo/transaction/	génère une transaction de modification du zoo

Action de recherche d'une entité

Paramètre	Valeur	Effet
order	expression JSONPATH[1]	Trie le résultat en fonction de l'expression
relation__<name>	'*'	Renvoie tous les objets ayant au moins une relation de ce type
relation__<name>	1234	Renvoie tous les objets ayant au moins une relation de ce type avec l'entité 1234
field__<name>__	'*'	Renvoie tous les objets ayant au moins un champ <name>
field__<name>__similarity	<value>	Renvoie tous les objets ayant au moins un champ <name> contenant <value> de manière approchée
relation_follow__<name>	x=1 / 2 / 3 / ...	Renvoie aussi tous les objets en suivant la relation <name> ou d'autres jusqu'à la profondeur x

Création d'une transaction

Exemple: création de deux personnes et d'un nouveau lien de mariage entre eux

Contenu du POST :

```
{
    "meta": { # métadonnées concernant la procédure, qui, pourquoi, comment, etc..
       "by": "Jean darmette",
       "procedure": "nouvelle-fiche-usager",
       "synchro": true / false / ["technocarte"],
    }
    "actions": [
        {
           "@id": 1,
           "type": "create-entity",
           "entity_def": "personne",
           "content": {
                "nom": "Dax",
                "prenom": "Micheline",
           },
        },
        {
           "@id": 2,
           "type": "create-entity",
           "entity_def": "personne",
           "content": {
                "nom": "Dax",
                "prenom": "Jean",
           },
        },
        {
           "@id": 3,
           "type": "create-relation",
           "relation_def": "marriage",
           "left": "@1",
           "right": "@2",

        },
        {
           "@id": 4,
           "type": "add-note",
```

```
            "entity": "@1",
            "content": {
               "html": "Déclaration de mariage avec Jean Dax"
            }
         }
    ]
}
```

Retour JSON:

```
{
   "err": 0,
   "transaction": 1234,
   "synchro_results": [
         {
            "type": "technocarte",
            etc...
         }
   ]
}
```

Schémas pour les contenus JSON

Les schémas utilisés sont au format JSONSCHEMA[2]. Ils sont stockés dans les définitions des objets.

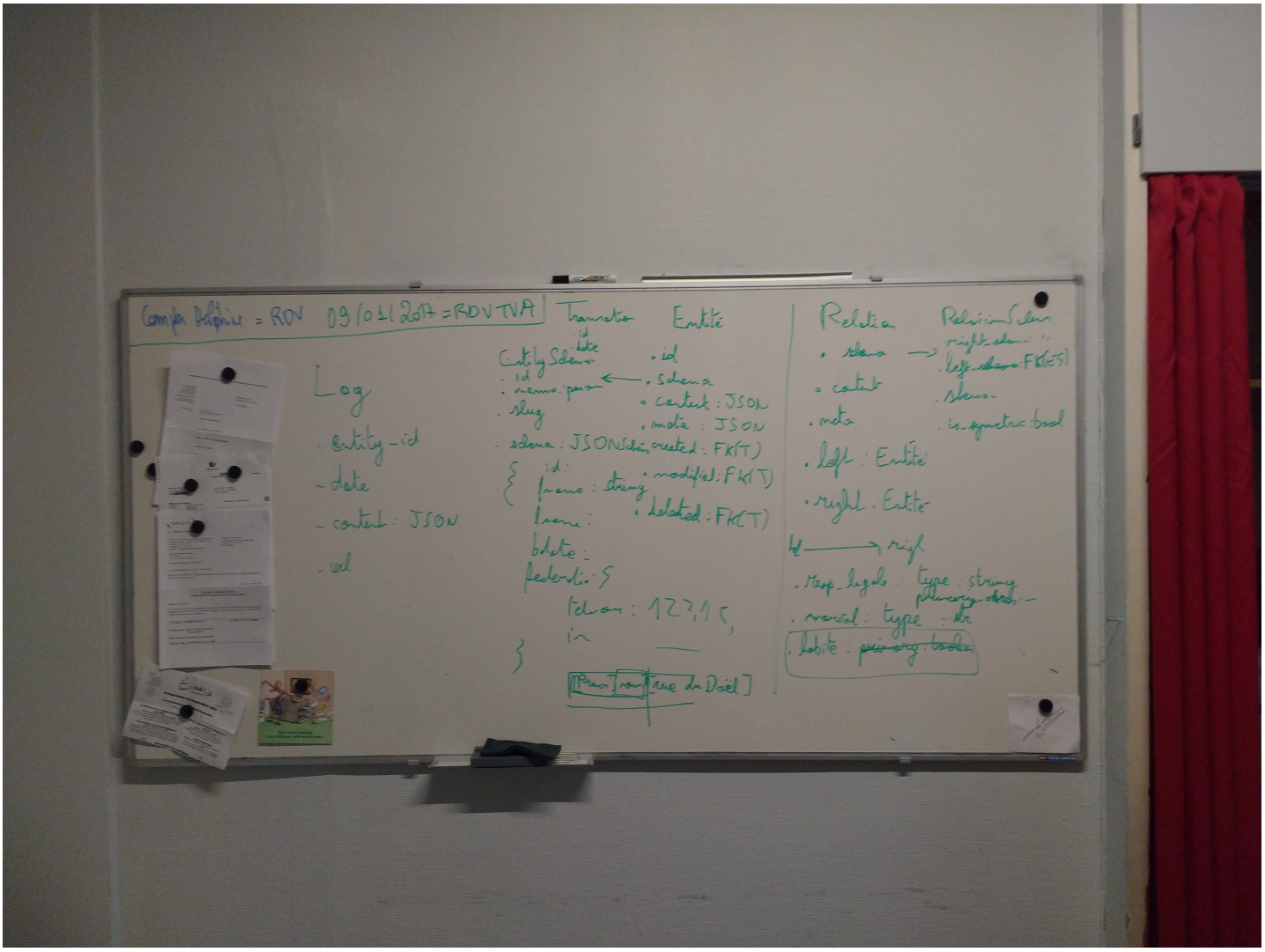
Vues de gestion

À définir.

Contrôle d'accès

- HTTP Basic : accès complet aux web-service si l'utilisateur a un droit en écriture/création sur les objets Entity.

Notes



[1] http://goessner.net/articles/JsonPath/

[2] http://json-schema.org/latest/json-schema-core.html#rfc.section.6

Fichiers

tableau-blanc-du-4-janvier-2017.jpg	3,59 Mo	04 janvier 2017	Thomas Noël