Hub Rdv ANTS

Avoir un service pour servir à l'ANTS les RdV pour les demandes de CNI et passeports disponibles de nos clients. Il doit aussi permettre de lister les RdV pris pour des identifiants de pré-demande ANTS. Il sera alimenté par une application "optionnelle" (activé par la configuration d'une apikey) dans chrono.

Le besoin

API ANTS: https://rendez-vous-api.france-identite.fr/docs

Tous les appels sont authentifiés via un entête HTTP "X-Hub-Rdv-Auth-Token".

Endpoints à exposer :

- /api/getManagedMeetingPoints :
  - Description: retourne une liste géolocalisée de lieux pour les RdV, ainsi qu'une URL pour chacun d'eux.
  - Paramètres en entrée: aucun
  - Schéma de la réponse : (JSON) une liste de dico avec les attributs,
    - id : un identifiant unique pour notre service du lieu de RdV
    - name/longitude/latitude/public_entry_address(numéro + rue)/zipcode/city_name : description géolocalisée du lieu
    - decoded_city_name: le nom avec des accents ???, il n'y a pas de description ni d'exemple pour ce champ, optionnel
    - website: url pour la prise de RdV sur ce lieu (site des démarches), optionnel
    - city_logo: url du logo de la collectivité / administration, optionnel
- /api/getManagedMeetingPoints
  - Description: retourne une liste de RdV disponibles pour un ensemble de lieux de RdV et un type de démarche
  - Paramètres en entrée (query-string)
    - meeting_point_ids: une liste d'identifiants de lieux de RdV (attribut id cité plus haut) (répétition du paramètre dans la querystring)
    - start/end_date: période, date iso8601
    - reason: le type de démarche: CNI, PASSPORT, CNI-PASSPORT, ~~RETRIEVAL (retrait du document)~~ (ce type est supprimé)
    - documents_number: le nombre de documents ou de personnes pour le RdV (je suppose pour des RdV multiples), à voir comment on interprète ça
  - Schéma de la réponse: (JSON) un dico indexé par les identifiants (sous forme de chaîne cette fois) des lieux de RdV, contenant pour chacun une liste de dico avec les attributs,
    - datetime: l'heure du rdv au format ISO8601
    - callback_url: URL pour obtenir un RDV à cette date, URL directement vers une demande en posant la bonne date si possible, ça suppose que ces demandes ait un champ calendrier en première page
- /api/searchApplicationIds
  - Description: retourne une liste de RdV existants étant donnée une liste d'identifiants de pré-demande ANTS
  - Paramètres en entrée (query-string)
    - applications_ids : une liste d'identifiants de pré-demandes ANTS (répétition du paramètre dans la query-string)
  - Schéma de la réponse: (JSON) un dico indexé par les identifiants de pré-demande ANTS avec pour chacun une liste de dico avec les attributs (mais on a pas de schéma JSON donc on ne sait pas ce qui est requis ou pas :/):
    - meeting_point: le nom du lieu de rdv (et pas l'id)
    - datetime: l'heure du rdv
    - management_url: l'URL de gestion du RdV, à voir ce qu'on pourra bien servir ici qui soit secure, on ne peut pas se permettre de servir des URLs avec les codes de suivi,
    - cancel_url: l'URL pour annuler le RdV, idem

Le hub

Le but du hub est d'aggréger les données des clients pour :
1. éviter que le hub de l'ANTS ne vienne charger l'infra Publik à moissonner tous nos clients avec beaucoup de requêtes
2. mutualiser les réponses entre clients
3. avoir du code pas tellement pertinent dans Publik
4. avoir un modèle économique pour la fourniture du service (abonnement)
5. permettre au client à nous de ne pas avoir à gérer le raccordement pour chacun d'entre eux (un seul token pour Entr'ouvert)

Modèle de donnée

Error executing the **plantuml** macro (Missing partial wiki_external_filter/_macro_image with {:locale=>[:fr, :en], :formats=>[:pdf], :variants=>[], :handlers=>[:raw, :erb, :html, :builder, :ruby, :rsb]}. Searched in: * "/usr/share/redmine/plugins/wiki_external_filter/app/views" * "/usr/share/redmine/plugins/wiki_external_filter/app/views" * "/usr/share/redmine/plugins/redmine_tags/app/views" * "/usr/share/redmine/plugins/redmine_entrouvert/app/views" * "/usr/share/redmine/plugins/plantuml/app/views" * "/usr/share/redmine/plugins/localizable/app/views" * "/usr/share/redmine/app/views")

Implémentation

- Application django
- Authentification sur notre infra d'authentification (cresson)
- Alimentation via une API django-rest-framework basique, un seul endpoint push, authentification via l'API key document JSON (schéma, ants-hub-push-schema.json)

Fonctionnement

Le hub expose deux blocs d'APIs:

- un bloc vers l'ANTS
- un bloc vers les chronos raccordés

API pour l'ANTS

- sur une requête pour les lieux de RdV:
  - on renvoie la liste des lieux connues tout raccordement confondues identifiés par leur identifiant interne (primary key sql)
- sur une requête les RdV disponible d'un certain type:
  - on recherche les lieux dans le rayon géographique recherché,
  - on récupère les plages du type de rendez-vous demandé dans la plage temporelle recherchée,
  - pour chaque plage on génère les rdv individuels à partir de la durée,
  - on retourne la liste des rendez-vous agrégés par identifiant du lieu
- sur une requête concernant des numéros de pré-demande
  - on recherche les rdv avec les même numéros
  - on les renvoie enrichi du nom du lieu

URL de prise de rendez-vous ou de gestion: le hub fera relai pour ne pas exposer les URLs réelles des raccordements :

```
https://ants.entrouvert.com/rdv/10-ville-de-machin/11-lieux-truc/CNI/20200110T1200/
https://ants.entrouvert.com/gestion/10-ville-de-machin/11-lieux-truc/CNI/20200110T1200/
https://ants.entrouvert.com/annulation/10-ville-de-machin/11-lieux-truc/CNI/20200110T1200/
```

API pour les sites de prise de rendez-vous raccordés

Il y a aura deux endpoints :

- /api/rendez-vous-disponibles/ pour pousser la configuration (collectivités, lieux) et les plages disponibles
- /api/predemande/ pour pousser les rendez-vous pris liés à des numéros de pré-demande

Application chrono

Nouvelle application dans chrono permettant l'alimentation du HuB. Une nouvelle entrée dans le menu hamburger "RdV ANTS" vers /manage/ants/.

Par défaut l'écran n'affiche rien à part un champ demandant l'API key, si le champ est rempli, on teste l'API key sur l'URL /api/push/ du hub, si ça renvoie {"err": 0}, l'apikey est valide, on l'enregistre en base et on affiche l'écran de configuration.

L'écran de configuration propose :

- de configurer une ou des collectivités (adresse via BAN + geoloc)
  - pour chacune des types de RdV (choix fermés) et les URLs,
- des lieux par collectivité,
  - pour chacun des types de RdV (choix fermés) et éventuellement des URLs (sinon URL générique au niveau collectivité, l'un ou l'autre, en l'absence des deux impossible de valider)

Modèle de donnée

Le même que plus haut pour la partie configuration des lieux/collectivités/types de RdV, ça se synchronise avec le hub à chaque modification et régulièrement avec en plus :

- pour chaque lieu, une clé vers un agenda
- un nouveau modèle Dirty(dirty: bool, agenda: fk(agenda), date: date, last_update_time: datetime) pour noter les jours d'un agenda ou un changement dans les plages disponibles a eu lieu

Fonctionnement

- ajout de deux signaux created-meeting / canceled-meeting, appelé pour chacun des évènements ils mettront à jour l'objet dirty pour le jour correspondant en le mettant à dirty=true et en mettant à jour last_update_time.
- les objets dirty des jours passés sont supprimés régulièrement,
- aussi régulièrement que possible on agrège les plages horaires des agendas pour chaque lieu pour les jours marqués dirty et on pousse cela dans le hub, puis on remet les objets dirty à False si last_update_time == à la valeur de lecture, sinon on y touche pas (il sera poussé de nouveau au prochain cycle, on évite ainsi une race condition entre le cron qui pousse et les signaux qui marque les jours "dirty")

Fichiers

ants-hub-push-schema.json	7,42 ko	01 avril 2023	Benjamin Dauvergne