QRCode signés

Aussi :

- le pad du projet publik https://pad.libre-entreprise.org/p/publik-billetique
- le tracim https://publik.tracim.fr/ui/workspaces/1/contents/html-document/1980

Le principe

Le connecteur aura pour objectif de produire des QRcode contenant une donnée signée électroniquement (un certificat électronique), lié ou non via un identifiant unique à un enregistrement en base pour permettre de valider l'origine de la donnée et d'éventuellement l'augmenter. Cette partie sera appelé le backend. Le connecteur fournira aussi une application Web style one-page application permettant la lecture, la validation, le pointage et l'augmentation des donnés d'un QRcode. Cette partie aura pour nom le fronted et visera une utilisation sur tablette et smartphone, i.e. la compatibilité minimale visée est Chrome Android et Safari iOS.

Fonctionnalités du backend

- initialiser automatiquement les données cryptographiques pour la signature (création des clés), assurer leur sécurité (rotation)
- fournir une API pour créer/lire/mettre à jour les enregistrements des QRcodes
  - à minimal le QRcode contiendra un uuid pointant vers l'enregistrement en base
  - une partie sera stockée en base mais pas mise dans le QRcode, métadonnées/données personelles
- fournir une API pour ajouter un évènement à l'historique d'un enregistrement (pointage), par défaut l'évènement sera "vu" mais on pourrait imaginer entrée/sortie, etc..
- contrôler les autorisations pour l'API de pointage et de lecture des métadonnées, tracer ces accès (savoir qui, en tout cas quel jeton d'accès) et quand a pointé tel qr-code
- fournir une API de création de jeton d'accès (pour le frontend) permettant de limiter dans le temps leur usage ou leur accès aux métadonnée (tel pointeur verra le nom mais pas l'adresse, etc...) et de configurer la présentation du lecteur

Fonctionnalités du front

- lire et décoder les QRcodes à l'aide de la caméra d'un smartphone
- afficher et présenter les données des QRcode (sous-ensemble et ordre des champs à afficher)
- valider la date de validité du QRcode
- récupérer et afficher (si autorisé par la clé) les donnés supplémentaire disponibles dans le backend
- filtrer automatiquement en fonction du paramétrage du jeton (ignorer les qrcode zfe quand on scan des billets théâtre et inversement)
- permettre le pointage (à la volée dès le scan ou après une action de l'utilisateur)
- rapporter un pointage existant (déjà passé, billet invalide)
- fonctionner hors-ligne, i.e. stocker les pointages dans localstorage et les pousser en tâche de fond idem pour la récupération des pointages existants

Cas d'usage

ZFE

ZFE = Zone à faible émission, les collectivités émettent des autorisations particulières par exemple pour des artisans ayant de vieux véhicule polluant. On créera un certificat avec la plaque d'immatriculation et le date limite de validité, on conservera en base nom/prénom/téléphone/adresse de la personne.

Un jeton unique de lecture sera transmis à la police municipale, le lien vers le frontend sera transforme en QRcode URL et transmis aux agents (par mail, courrier, whatever) ils n'auront qu'à le lire avec leur téléphone/tablette.

La police municipale sera à même de valider le certificat et si en ligne de lire les informations supplémentaires qui ne seront pas publiquement dans les QRcode.

Contrôle d'accès à un évènement

Pour chaque billet vendu on créera un certificat avec le numéro de la place, zone d'accès, etc.. Des jetons seront produits pour chaque agent de sécurité et un lien vers le frontend contenant ce jeton fournit sous forme de QRcode à lire par le smartphone de l'agent.

Chaque smartphone à la validation d'un QRcode remontera l'évènement de pointage au backend, évènement qui redescendra dans les autres smartphones pout empêcher le double pointage.

Choix techniques

- encodage des jetons via CBOR (c'est compact) et utilisation du mode binaire des qrcode (encodage b45)
- signatures via NaCl (libsodium), signatures courtes, clés faciles à gérer et générer
- utilisation d'une lib javascript de lecture des qrcode par caméra, htmlt5-qrcode (et tweetnacl, cbor et base45-js)
- localstorage pour stocker les informations de pointage
- le backend est un connecteur passerelle
- on parle d'une intégration dans une cellule combo mais je ne sais pas si c'est vraiment utile, un écran de tablette ou de téléphone est étroi y intégrer des bouts du portail agent me semble inutile pour le cas d'usage, je n'en ferai pas une cible pour la v1
- l'application du lecture est servie par le connecteur, autorisation via apikey

Modèle de donnée du connecteur

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")

Prototype

Backend en ligne de commande limité à produire les QRcodes signés, frontend qui ne fait que lire.

https://git.entrouvert.org/entrouvert/misc-bdauvergne/src/branch/main/qrcode-certificate/v1

- backend: dans signed_qrcode, en python utilise les libs cbor2, nacl et qrcode
- frontend: dans qrcode_reader, en JS/NodeJS utilise les libs tweetnacl, cbor, base45-js, html5-qrcode, compilé avec browserify puis raccourci avec terser, ça donne un simple index.min.js à charger

Plan développement :

Pour le 17 Novembre en recette, implémenter le cas d'usage "ZFE". i.e création d'un connecteur "qrcode" avec les endpoints suivants :

- create-certificate : Permet de créer un certificat avec une date de début de validité, fin de validité, données & métadonnées associées
- get-qrcode : permet de récupérer le qrcode associé à un certificat
- create-qr-code-reader : permet de créer un lecteur de qr code avec la liste des métadonnées accessible, date de début de validité, date de fin de validité.
- qr-code-reader : donné un uuid renvoyé par le endpoint précédent, affiche une page web avec un lecteur de QRCode qui permet de scanner les code de ce connecteur, et d'afficher les données associées et les metadonnées associées qui ont été autorisées

Le calendrier & les spécifications précises pour le pointage viendront après.