Combo - Development #25462

Envoi de notifications web-push aux utilisateurs connectés et abonnés

24 juillet 2018 15:31 - Anonyme

Statut:	Fermé	**Début:**	24 juillet 2018
Priorité:	Normal	**Echéance:**	01 septembre 2018
Assigné à:		**% réalisé:**	0%
Catégorie:		**Temps estimé:**	0:00 heure
Version cible:			
Patch proposed:	Oui	**Planning:**	

Description

Je pense décomposer les patchs ainsi (dans le désordre) :
- les modèles liés à combo.apps.notifications et au compte de l'utilisateur pour stocker l'état des abonnements + l'état des envois
- l'API http pour gérer l'abonnement et le désabonnement d'un utilisateur
- la commande de gestion django pour envoyer toutes les notifications en attente, qui sera vouée à tourner dans un cron.
- les fragments de template django + javascript (ajouts au service-worker.js) pour utiliser toutes ces fonctionnalités

j'ai un doute pour ce qui sera dans publik-base-theme/templates/combo/... et ce qui sera dans le code même de combo. à voir s'il faut un second ticket dans publik-base-theme pour ce qui sera dans l'interface.

Note pour tester les "vraies" notifications sous Chrome-ium (avec images et tout), et pas celles intégrées à Gnome, il faut désactiver chrome://flags/#enable-native-notifications

Demandes liées:

Lié à Publik - Development #25066: PWA : ajouter la possibilité d'envoyer des...	**Fermé**	**05 juillet 2018**

Révisions associées

Révision 88baf45c - 20 décembre 2018 14:14 - Frédéric Péters

pwa: add service worker registration script (#25462)

Révision e039b655 - 20 décembre 2018 14:14 - Frédéric Péters

pwa: add basic support for push notifications (#25462)

Historique

#1 - 24 juillet 2018 15:32 - Anonyme

- Lié à Development #25066: PWA : ajouter la possibilité d'envoyer des notifications web-push aux utilisateurs ajouté

#6 - 24 juillet 2018 15:33 - Frédéric Péters

- l'API http pour gérer les abonnements d'un utilisateur

Tu entends quoi par "les abonnements d'un utilisateur" ?

#7 - 24 juillet 2018 15:34 - Frédéric Péters

- le code python d'envoi des notifications en attente
- la commande de gestion django pour envoyer toutes les notifications en attente, qui sera vouée à tourner dans un cron.

C'est quoi la différenc entre ces deux points ?

#8 - 24 juillet 2018 16:00 - Anonyme

- Description mis à jour

#9 - 25 juillet 2018 14:24 - Anonyme

- Sujet changé de Envoi de notifications web-push aux utliisateurs connectés et abonnés à Envoi de notifications web-push aux utilisateurs connectés

et abonnés

#10 - 25 juillet 2018 18:34 - Anonyme

Je pousse mon travail en cours dans la branche wip/pwapush de combo

#11 - 30 juillet 2018 18:54 - Anonyme

J'ai pu avancer dans combo sur la question des clés et du codage des messages en partance de combo-server
Je dois encore bosser sur le code JS qui réceptionne les messages, et régler côté python la commande de mgmt. Souscription, utilisation de l'API django-push-notification, et nouveaux models sont OK, l'envoie de message de test est OK.

J'ai aussi une branche wip/pwapush pour publik-base-theme

À suivre

#12 - 30 juillet 2018 20:02 - Frédéric Péters

Sans regarder les changements depuis, pour mémoire des discussions du 26/7 :

- avoir une cellule dédiée pour le contrôle de l'usager sur les notifications (dans ce premier temps ce serait seulement le choix de les recevoir en push, ça pourra évoluer pour lister le détail des appareils où il a fait ce choix, pourra contenir d'autres paramétrages pas nécessairement liés concernant les notifications, genre des thématiques, des horaires, etc.)
- ne pas ajouter une nouvelle dépendance (retrying), l'attente induite est incompatible avec un job cron qu'on espère rapide pour pouvoir être appelé fréquemment, s'il y a un soucis à appeler trop fréquemment un webservice pour une notification, il y a possibilité de joindre au modèle d'envoi de celle-ci le nombre d'essais et le timestamp du dernier essai, pour pouvoir sauter une notification qui "ne passerait pas").
- intégrer le code de service worker dans le service worker existant plutôt qu'en créer un second

#13 - 24 août 2018 16:03 - Anonyme

- Description mis à jour

#14 - 24 août 2018 17:31 - Anonyme

- Fichier Capture d'écran de 2018-08-24 16-44-31.png ajouté

- Fichier 0001-pwa-add-django-push-notification-and-compatibility-t.patch ajouté

- Fichier 0002-pwa-create-WebPushRecord-and-WebPushCell-25462.patch ajouté

- Fichier 0003-pwa-add-url-views-and-javascript-for-webpush-25462.patch ajouté

- Fichier 0003-pwa-add-views-and-statics-for-webpush-25462.patch ajouté

- Fichier 0004-pwa-create-a-management-command-send_webpush-25462.patch ajouté

- Fichier 0005-pwa-add-test_webpush-25462.patch ajouté

- Fichier 0006-pwa-activate-django-admin-model-UI-25462.patch ajouté

- Statut changé de En cours à Solution proposée

- Patch proposed changé de Non à Oui

Voilà ma proposition décomposée en 6 patchs.

- 1er commit "paramétrage" : configurez PUSH_NOTIFICATIONS_SETTINGS avec les instructions dans settings.py pour générer ses propres clés VAPID. À noter que la dépendance vers django-push-notification ne peut pas être mise dans setup.py ou le requirements.txt car elle est incompatible avec Django<1.11.
- 2e commit "modèles" : les modèles dans combo.apps.pwa.models, dont une cellule Combo bien à part pour l'interface d'autorisation du Push, et la migration qu'il faut
- 3e commit "vues" : les urls pour s'abonner, et celle pour "ack" ou "forget" la notification qui présente des actions. Le service-worker unique, et le webpush.js qui est le controlleur injecté par la cellule combo WebPushCell.
- 4e "send_webpush" : la commande de management, sa fonction utilitaire, et la modif du cron.hourly
  - Pour tester, créez une notification dans l'admin (cf. ci-dessous), puis insérez la cellule Combo WebPush dans une page et abonnez vous, la commande suivante va envoyer toutes les notifications pas encore envoyées, et qui ont moins de 24h depuis leur création.

```
$ combo-manage tenant_command send_webpush --all-tenants
```

- 5e "test": un test de la commande send_webpush
- 6e commit "admin": pour le debug et le test, c'est pratique d'avoir les modèles dans l'admin django : crez

La capture montre une notification reçue avec chromium (en bas à droite). L'action OK va "ack sur le modèle Notification associé, et tout clic va ouvrir ou "focus" sur l'URL de Notification.url

La seule impasse faite par rapport est la fin du second point de Fred dans son commentaire 12 : celle de l'enregistrement du nombre d'essais et timestamp d'envoie des requêtes en erreur au service de push (google, ff, etc) car je préfère passer cette option pour le moment au profit d'un log du statut des envois dans le modèle WebPushRecord. Cela alourdirait pas mal le code, **or on ne sait pas pour le moment** si c'est nécessaire dans le

cadre de l'utilisation du push avec les donneées de combo.apps.notifications.

#15 - 27 août 2018 09:41 - Anonyme

- Fichier 0001-add-pwa-webpush-cell-style-25462.patch ajouté

Elias Showk a écrit :

> Voilà ma proposition décomposée en 6 patchs.

- et un 7e patch pour publik-base-theme pour simplement gérer le style de la cellule combo "webpush"

#16 - 27 août 2018 10:34 - Anonyme

- Fichier 0005-pwa-add-test_webpush-25462.patch ajouté

Elias Showk a écrit :

> Elias Showk a écrit :
>
> > Voilà ma proposition décomposée en 6 patchs.

- Voilà une correction du patch 0005 pour les tests (suivre le résultat ici https://jenkins2.entrouvert.org/job/combo-wip/)

#17 - 15 novembre 2018 09:10 - Frédéric Péters

- Statut changé de Solution proposée à En cours

- Priorité changé de Haut à Normal

- Patch proposed changé de Oui à Non

#18 - 27 novembre 2018 17:08 - Frédéric Péters

- Statut changé de En cours à Solution proposée

- Patch proposed changé de Non à Oui

Branche wip/25462-pwa-push avec le minimum nécessaire.

- dépendance directe sur pywebpush, on évite django-push-notifications et la série de dépendances supplémentaires qui viendraient avec, on évite aussi ses modèles pas nécessairement adaptés et plus généralement ça permet de rester à un niveau sans trop d'abstractions compliquées à justifier.

Les notifications push, comment ça marche ?

Alors premier point, on peut oublier le "notification" de la question pour le moment, s'intéresser uniquement à "push".

Il s'agit donc d'un mécanisme permettant à un navigateur^Wsite de recevoir des données sans rien avoir demandé.

Comment ça marche ? D'une manière ou d'une autre le navigateur polle un serveur de notifications, s'il y en a une pour lui, il la prend, "réveille" le site, lui file la donnée.

On a donc trois morceaux : le navigateur, le serveur de notifications et le site django.

Le site django est configuré avec une paire de clés "VAPID"; il y a un python-vapid pour générer ça mais bête et méchant, ça se fait avec openssl.

```
openssl ecparam -name prime256v1 -genkey -noout -out vapid_private.pem
openssl ec -in vapid_private.pem -pubout -out vapid_public.pem
openssl ec -in vapid_private.pem -outform DER|tail -c +8|head -c 32|base64|tr -d '=' |tr '/+' '_-' > private_k
ey.txt
openssl ec -in vapid_private.pem -pubout -outform DER|tail -c 65|base64|tr -d '=' |tr '/+' '_-' > public_key.t
xt
```

(lignes de https://mushfiq.me/2017/09/25/web-push-notification-using-python/, no comment, toujours bien qu'un outil qui ne marche pas)

Clé publique clé privée ça va dans la config django,

```
PWA_VAPID_PUBLIK_KEY = "BFzvUdXB-HLKQV1mYEtmIOaSYkVdit-qLgoWpfmfQjQUlpAfnd37GyRHXRRKJ7iTTFOjXjiChtAGbcF8b7_mXR
Q"
PWA_VAPID_PRIVATE_KEY = "..."
```

Il y a aussi une notion de "claims", va savoir, on en a aussi besoin, hop dans la config :

```
PWA_VAPID_CLAIMS = {'sub': 'mailto:admin@entrouvert.com'}
```

Le site ainsi configuré, direction le navigateur.

L'usager manifeste son envie de recevoir des notifications. Il y a appel à combo_pwa_subscribe_user. (ce patch expose cette fonction, mais rien qui ne l'appelle, c'est présent côté toodego).

Cet appel abonne l'usager :

```
  swRegistration.pushManager.subscribe({
    userVisibleOnly: true,
    applicationServerKey: applicationServerKey
  }).then(...)
```

En résultat on obtient un objet PushSubscription et vite fait on l'envoie côté django pour l'enregistrer :

```
function combo_pwa_update_subscription_on_server(subscription) {
  $.ajax({
       url: '{% url "pwa-subscribe-push" %}',
       data: JSON.stringify(subscription),
```

On a donc côté django une table avec identifiant usager / blob ~opaque d'abonnement.

C'en est fini pour le navigateur, on regarde côté django.

Il y aurait des trucs à faire évoluer côté notification, recevoir des paramètres, gérer des priorités, etc. mais en l'état, au plus simple, signal post_save, et à la création d'une notif, on arrive ainsi dans combo.apps.pwa.signals.

Là, on a l'utilisateur (notification.user), on itère sur tous les abonnements push de cet utilisateur, et pour chacun d'eux, on webpush :

```
    for subscription in PushSubscription.objects.filter(user_id=instance.user_id):
        pywebpush.webpush(
                subscription_info=subscription.subscription_info,
                data=message,
                vapid_private_key=settings.PWA_VAPID_PRIVATE_KEY,
                vapid_claims=settings.PWA_VAPID_CLAIMS
                )
```

Ça chiffre ça signe que sais-je et c'est envoyé au serveur de notifications (dont l'url est dans le subscription_info) (et donc, Google).

Revient le navigateur qui continuait à poller le serveur de notifications et qui voit un truc pour lui, le prend et le passe au site, qui l'attrape au vol :

```
self.addEventListener('push', event => {
```

Dans event, il y a data, c'est le data=message de l'appel à pywebpush. On parse le json, on en sort les paramètres de la notif et hop, on en demande l'affichage :

```
  self.registration.showNotification(title, options))
```

Et ça s'affiche, tout le monde est content, etc.

Donc, dans la branche, d'abord un commit qui prend le code d'enregistrement du service worker, qui était dans publik-base-theme, et le tape côté combo.

Puis un commit qui fait tout le brol raconté ici.

#19 - 27 novembre 2018 18:07 - Benjamin Dauvergne

Tentative de relecture de http://git.entrouvert.org/combo.git/log/?h=wip/25462-pwa-push :

- ~~pourquoi veut-on que service_worker_registration_js soit un template plutôt qu'un fichier static ? Je n'y vois même pas de {% url ... %} ?~~ je m'auto répond, la raison vient dans le patch suivant
- je rendrai PushSubscription unique par utilisateur, avec un get_or_create() uniquement sur user, en plus je vois que si on a un évènement pushsubscriptionchange (je ne sais pas trop à quoi ça correspond) on va pousser encore une nouvelle souscription pour le même utilisateur sur le même terminal vraisemblablement sans nettoyer les autres
- détail:

  ```
      for subscription in PushSubscription.objects.filter(user_id=instance.user_id)
  ```

  plutôt user=instance.user et peut-être renommer instance en notification ça mange pas de pain
- je découplerai la création des Notification de l'envoi des Push, ça impose un nouveau modèle PushNotification mais:
  - avec le code actuel un plantage dans pywebpush.send() va complètement foirer la création de la Notification
  - un plantage de pywebpush.send() ne sera jamais repris, et la notification n'arrivera jamais à destination (je ne sais pas si pywebpush

remonte des trucs genre, "souscription invalide" qui permettrait de nettoyer les souscriptions devenues mortes, ou en tout cas de différencier les plantages temporaires des plantages définitifs)
    - en vrai on devrait faire ça pour tout nos communications fire&forget (les mails aussi), on le fait pas, soit, mais là c'est nouveau, alors à moins qu'on soit toujours au stade démo pour voir à quoi ça ressemble il faudrait aller jusque là, ça m'irait que ça passe par un truc générique genre un modèle BackgroundJob, mais du spécifique ira très bien
- de ce que je comprends il n'y a que le terminal qui semble savoir où on en est des souscriptions, comment elles sont stockés et la têtes qu'elles ont pour pouvoir comparer à ce qu'on stocke coté serveur, j'aimerai bien ta compréhension du truc (ça reprend des idées dites plus haut sur comment s'assurer de ne pas stocker une série de subcription_info qui ne servent à rien ad vitam)

Et je vais en rester là tout le reste est bien.

Je souligne que ce code a le mérite de nous faire avancer et comprendre comment tout ça fonctionne.

#20 - 27 novembre 2018 18:29 - Frédéric Péters

> je rendrai PushSubscription unique par utilisateur, avec un get_or_create() uniquement sur user, en plus je vois que si on a un évènement pushsubscriptionchange (je ne sais pas trop à quoi ça correspond) on va pousser encore une nouvelle souscription pour le même utilisateur sur le même terminal vraisemblablement sans nettoyer les autres

En fait un utilisateur peut abonner plusieurs navigateurs; c'est un détail que j'avais noté côté GL et pas repris ici : « L'activation des notifications est liée à un appareil/navigateur précis, il est possible de les activer en même temps sur le mobile et sur le desktop. Par contre la désactivation est globale. ».

> je découplerai la création des Notification de l'envoi des Push, ça impose un nouveau modèle PushNotification mais:

Pas noté mais ce serait en perspective. Dans l'immédiat, j'en resterais au fonctionnement simple via signal (mais il faudrait un try/except et logguer l'éventuelle erreur de pywebpush). En fait :

> la notification n'arrivera jamais à destination

il y a plein de circonstances qui feront ça, (il me semble), et les notifs sont doublées par une présence dans le portail, et les trucs importants sont de toute façon également envoyés par email/sms côté workflow.

> de ce que je comprends il n'y a que le terminal qui semble savoir où on en est des souscriptions, comment elles sont stockés et la têtes qu'elles ont pour pouvoir comparer à ce qu'on stocke coté serveur, j'aimerai bien ta compréhension du truc (ça reprend des idées dites plus haut sur comment s'assurer de ne pas stocker une série de subcription_info qui ne servent à rien ad vitam)

Un truc qui est fait dans django-push-notification c'est tenter de construire un identifiant (qui au final me semble juste être le nom du navigateur) et de l'associer à l'abonnement, l'unicité se fait alors sur (user_id, identifiant_inventé_pour_le_terminal) plutôt que sur (user_id, subscription_info). C'est peut-être nécessaire dans la vraie vie™ mais c'est du bricolage.

Ce qui pourrait être fait aussi c'est limiter un utilisateur à genre 10 abonnements (j'ajouterais alors un creation time).

Pour information, la tête d'un subscription_info :


```
{"keys": {"auth": "F2Ja6r45jWB__2Qgq_2Eiw", "p256dh": "BJy9AR_qE_G94-_2XNFwMn1AhbUa9MFhtg6K46iMNrITXfz0y6nWzAH
wchWI1CIAAY7xwJhZUvCToB1wKMnH9No"}, "endpoint": "https://fcm.googleapis.com/fcm/send/erlgwyxKNwM:APA91bHh7gYBi
IcMYe209QW2Q_3umTiQMpKixzQTlFzRT9kkQxcbc0AnQHVZYoC9HWcfs_RPivcjdQzwO4s8pocifnd-EVqPQZv1cYzmS84dK3iZ3uu8ddvGvnm
m6KhwxfBtqmoACqJB", "expirationTime": null}
```


> (...) au stade démo (...)

mais surtout, ça, oui, on est loin de pouvoir considérer ces notifications comme un moyen de communication réaliste.

#21 - 06 décembre 2018 10:59 - Frédéric Péters

J'ai poussé dans la branche l'ajout d'un timestamp à l'abonnement, pour permettre plus tard du nettoyage, mais ça m'irait d'en rester à ça sur ce ticket, pour ne pas obliger GNM à vivre dans une branche.

#22 - 19 décembre 2018 16:51 - Benjamin Dauvergne

- Statut changé de Solution proposée à Solution validée

Il manque juste le try/except dans le signal autour de pywebpush, sinon c'est ok.

#23 - 20 décembre 2018 14:15 - Frédéric Péters

- Statut changé de Solution validée à Résolu (à déployer)

try/except ajouté, merci.

```
commit e039b655e1aa4c66c43e61b945f3e4e0685ec349
Author: Frédéric Péters <fpeters@entrouvert.com>
Date:   Mon Nov 26 17:14:05 2018 +0100

    pwa: add basic support for push notifications (#25462)

commit 88baf45c715c4c98bffeb3b90c778294c1f465da
Author: Frédéric Péters <fpeters@entrouvert.com>
Date:   Mon Nov 26 15:46:38 2018 +0100

    pwa: add service worker registration script (#25462)
```

#24 - 23 décembre 2018 15:09 - Frédéric Péters

- Statut changé de Résolu (à déployer) à Solution déployée

Fichiers

Capture d'écran de 2018-08-24 16-44-31.png	540 ko	24 août 2018	Anonyme
0001-pwa-add-django-push-notification-and-compatibility-t.patch	4,6 ko	24 août 2018	Anonyme
0002-pwa-create-WebPushRecord-and-WebPushCell-25462.patch	9,25 ko	24 août 2018	Anonyme
0003-pwa-add-url-views-and-javascript-for-webpush-25462.patch	22,3 ko	24 août 2018	Anonyme
0003-pwa-add-views-and-statics-for-webpush-25462.patch	22,3 ko	24 août 2018	Anonyme
0004-pwa-create-a-management-command-send_webpush-25462.patch	3,68 ko	24 août 2018	Anonyme
0005-pwa-add-test_webpush-25462.patch	3,93 ko	24 août 2018	Anonyme
0006-pwa-activate-django-admin-model-UI-25462.patch	1,82 ko	24 août 2018	Anonyme
0001-add-pwa-webpush-cell-style-25462.patch	864 octets	27 août 2018	Anonyme
0005-pwa-add-test_webpush-25462.patch	4,31 ko	27 août 2018	Anonyme