Contribuer à Publik

Pour démarrer

Publik est composé de différents modules, par ordre alphabétique :

- Authentic, application de gestion d'identité
- BiJoe, bibliothèque de simplification de requêtes et création de tableaux de bord BI
- Chrono, module de gestion de rendez-vous et évènements
- Combo, application de gestion de contenu
- Fargo, module de porte-documents
- Hobo, module de déploiement et d'orchestration des différentes applications, fournit la gestion du multitenants (via la bibliothèque django-tenant-schemas), assure le partage du paramétrage, etc.
- Passerelle, module d'accès aux multiples sources de données et services
- w.c.s., last but not least, application de conception et de paramétrage de formulaires en ligne.

À côté de ces modules applicatifs, d'autres modules participent à l'ensemble :

- django-mellon amène les fonctionnalités de SSO compatible SAMLv2 aux différentes applications;
- eopayment implémente une couche d'abstraction vers différents systèmes bancaires;
- gadjo fournit le style des interfaces de backoffice;
- publik-base-theme reprend le code, partagé et spécifique, des intégrations graphiques.

Tout cela peut fonctionner en s'appuyant sur ces projets majeurs que sont Python et Django, et bien sûr différents autres modules, ponctuellement utiles, sur des besoins particuliers.

Ce survol rapide des briques étant réalisé, avant de contribuer à l'un ou l'autre module, il est important de noter qu'une des forces de Publik est dans sa capacité d'interactions avec des applications tierces; ainsi, le développement de celles-ci, pour leur ajouter la prise en charge du protocole OpenID Connect pour le SSO, ou leur ajouter des API, contribue également de manière importante à Publik.

Pour explorer plus avant cet aspect:

- la documentation du protocole OIDC : https://openid.net/connect/
- API REST dans les grandes lignes : https://restfulapi.net/

Cela posé, ce document se concentre sur la contribution aux modules de Publik.

Pour faciliter le développement local de ces modules, le projet "publik-devinst" existe.

Périmètre de contribution

La contribution ne s'arrête pas au code de Publik et des modules satellites. Vous pouvez également contribuer au développement de connecteurs pour les logiciels métier ou à la rédaction de la documentation. La relecture de correctifs existants est également une contribution très utile.

Obtenir de l'aide

Le développement et le support autour de Publik est centralisé via l'outil redmine, https://dev.entrouvert.org, l'inscription est libre, la création d'un compte se fait à l'adresse https://dev.entrouvert.org/account/register et bien sûr jamais l'adresse électronique mentionnée ne sera utilisée à d'autres fins.

Code de conduite

Dans tous les échanges, une bienséance élémentaire est attendue, soyez poli·e, n'insultez personne.

Où démarrer

Publik en local

Le meilleur endroit pour développer Publik est son propre ordinateur ; un projet et une documentation dédiée à l'installation d'un

environnement de développpement local existe.

Découvrir Publik

Équipé d'une installation locale et du guide de l'administrateur fonctionnel, découvrir ce que Publik peut déjà faire, c'est important parce que beaucoup de choses existent déjà et que les connaitre permet d'ajouter une idée au meilleur endroit, et que cet ajout cadre au mieux avec son environnement.

Une première contribution

Important, scratch your own itch. (même si la réalité des contextes sera souvent différente, c'est important à rappeler).

S'il s'agit d'une nouvelle fonctionnalité, il est utile de commencer par ouvrir un ticket, pour qu'elle puisse être validée, guidée, etc.

Ensuite,

Modifier le code

- ici les considérations de style et de maintenabilité * la présence de tests (et le fait qu'on utilise py.test), etc.

Produire un patch

- produire un patch ** le relire soi-même

La première ligne du message d'un commit doit être de la forme : **petite description du patch (#XXXXX)** où XXXX est le numéro de ticket où le développement est suivi (on notera aussi l'absence d'une majuscule sur la première lettre).

Des indications supplémentaires sont possibles dans le message de commit sans formalisme particulier, laisser cependant une ligne vide entre la première ligne formelle et la suite du message.

Attacher un patch à un ticket

(et avant ça trouver/créer le ticket ?)

- relire une nouvelle fois le patch
- exécuter une nouvelle fois les tests
- attacher le patch / proposer une branche

Attendre

On essaie de regarder rapidement mais pour autant on peut être occupés sur d'autres projets

Échanger

Une fois une contribution proposée vient le temps de la relecture.

Il n'y a pas de règles définissant précisément comment les relectures de code doivent se faire, la seule règle est qu'elles doivent se faire : une modification doit être validée par au moins une personne avant de pouvoir être intégrée.

Les relectures ont deux rôles importants :

- le partage de connaissances : le relecteur peut partager les siennes, fournir davantage de contexte sur le module ou le changement attendu, et assurer qu'il n'y ait pas d'effets de bord inconnus de l'auteur. Par ailleurs, les relectures encouragent tout le monde à lire et comprendre le code, pour éviter les parts de code connues ou comprises par une seule personne.

- l'amélioration du code : l'objectif évident de la relecture est d'éviter les bugs mais elles peuvent aussi être un espace de discussion sur les décisions d'architecture; en effet du code qui fonctionne n'est pas nécessairement du code qui pourra évoluer ou être facile à maintenir.

Ressources / pour aller plus loin

- Domaines particuliers,
  - Développement d'un connecteur
  - Développement d'une cellule JSON
- Python

- Django