COVAR DARWIN

Bruno POMMEREL / Cédric DIRAND

11 Avril 2019

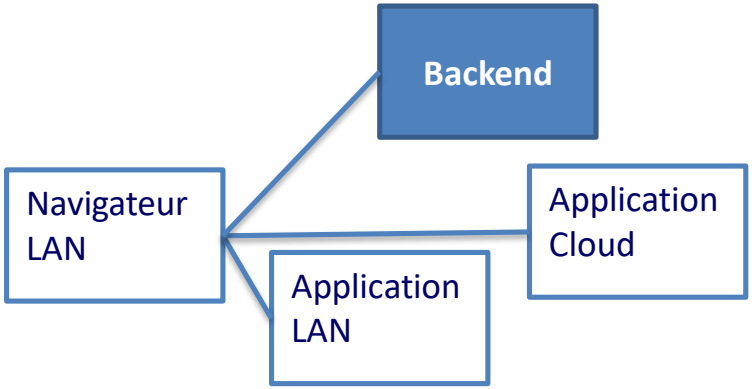
Aspects d'architecture décrits dans ce COVAR

1. Hébergement Cloud ou LAN du backend (**décision attendue**)
2. Authentification multiple PingFederate
3. Alimentation du catalogue applicatif Swap
4. Alimentation des droits utilisateurs (**décision attendue**)
5. Mise en cache de plusieurs informations pour améliorer la robustesse et la disponibilité de la solution (**décision attendue**)

Non présenté dans ce COVAR

- Gérer l'affichage des fenêtres des applications en fonction de situations de travail (en cours de travail UX)

Le Backend Swap est écrit en Java et utilise le framework Springboot 2 (stack Fun). Il gère les historiques sociétaires/événements, le catalogue applicatif et les droits 0 des utilisateurs dans une base de données.



Scénario 1 : Hébergement dans le Cloud (PFS)

Base: MongoDB
Ingénierie PFS : Github, Travis
POC Réalisé et fonctionnel.

+ Dans le sens de la démarche Cloud First
+ Accessible depuis d'éventuels partenaires
- Vigilance supplémentaire à la sécurisation standard
- Nécessite de dupliquer / déporter / exposer des référentiels des droits présents dans le LDAP (et des personnes dans le cas où tout n'est pas sous Ping Federate).

Scénario 2 : hébergement sur le LAN

Hébergement du backend / bdd sur le LAN Maif

Base: PostgreSQL
Ingénierie standard : Gitlab, Jenkins

+ Accès aux référentiels contenus dans LDAP
+ Hébergement sur le même socle que les applications
- Backend non accessible de l'extérieur nativement, au même titre que les applications métiers

Validation Attendue

Préconisation projet : hébergement dans le Cloud.

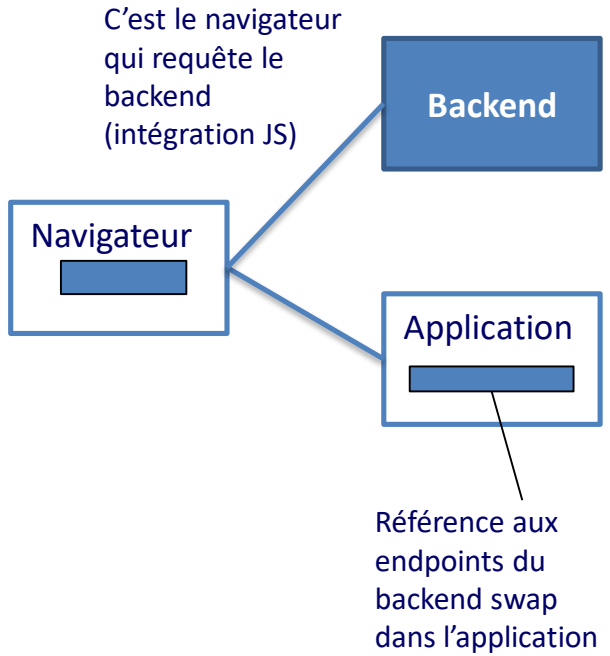
Le backend expose des endpoints restituant :

- des données des sociétaires (numéro, nom, prénom), d'événements (numéro)
- des données sur les applications (nom, url) pour les débranchements

Le backend est accédé par le browser via les applications qu'il affiche (Nora, Madere, …) sans redemander une authentification tout en authentifiant l'utilisateur.

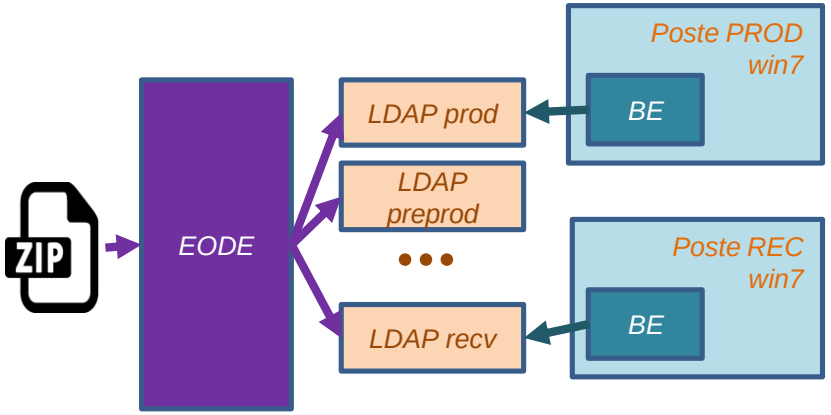
L'authentification Swap est portée par une authentification préalable de l'application par PingFederate : Swap utilise le cookie PingFederate pour générer son propre token de manière silencieuse via un script JavaScript.

➔ Toutes applications qui intègrent une fonction du backend doit être sous PingFederate.
➔ Ceci garantit le fonctionnement sur l'ensemble des postes (MAIF ou non) et sur l'ensemble des formes d'authentification (double facteur, kerberos, login/pass).

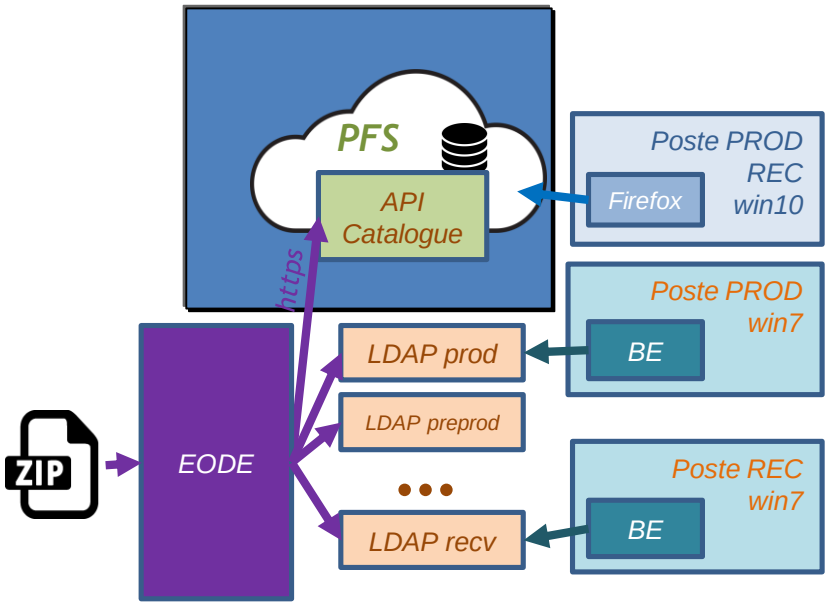


La liste des applications accessibles par le BE se trouve dans un annuaire LDAP. Les informations sur les applications sont nécessaires dans le débranchement pour connaître l'URL et les paramètres, et dans le moteur de recherche d'application.

La mise à jour de ce référentiel ne contenant pas de données personnelles est réalisé par EODE



Alimentation en place pour le BE



Alimentation en Y pour Swap

Pour Swap :

Mise en place d'une alimentation en Y.

Le catalogue applicatif Swap est conçu multitenant pour héberger dans une seule instance tous les environnements projet.

A terme, EODE n'alimente plus le LDAP.

4/ Les droits utilisateurs pour affiner l'affichage

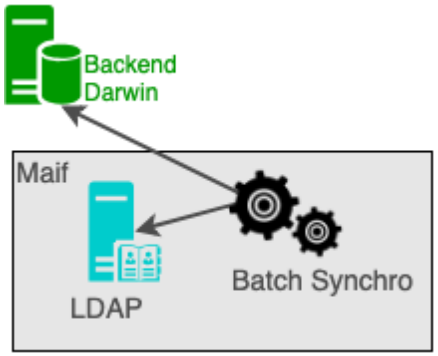
Les droits sont nécessaires pour restituer uniquement les applications accessibles par l'utilisateur. Ces habilitations correspondent au droit 0, géré dans l'OU Droits de l'annuaire LDAP alimenté par IDM.

Chaque application garde la responsabilité de la gestion des ces accès.

Actuellement, le BE lit les droits 0 dans le LDAP au travers des fonctions de recherche. En déplaçant le catalogue applicatif, l'interrogation de ces droits devient un sujet à instruire.

Scénario Copie

Copie des droits 0 dans une base MongoDB sur PFS.



+ backend autonome
- Duplication du référentiel
- Ajout d'une brique batch à la solution

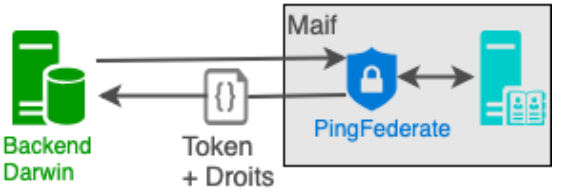
Scénario Extension

Extension de l'annuaire LDAP dans le cloud et accès direct.

+ Synchro référentiel automatique
+ Liaison simplifiée avec Backend Darwin
- Scénario complexe à instruire (sécurité, technique).

Scénario PingFederate

Exposition par PingFederate d'une API de consultation de droits.



+ Un seul référentiel des droits (cible)
- Ping Federate devient de + en + un SPOF

Validation Attendue

Préconisation projet : Scénario Ping Federate (dev spécifique PingFederate à POCer)

Les modules Javascript fournis aux applications se connectent régulièrement aux APIs pour :

- Lister les applications disponibles pour un utilisateur
- Connaître l'url pour débrancher vers une application
- Enregistrer/consulter l'historique des sociétaires, dossiers traités.

Appels systématique

Cache partiel accédé en cas d'indispo du backend

Cache partiel systématique rafraichi toutes les X minutes

Cache complet rafraichi toutes les X minutes

	Pas de cache	Cache backup	Cache par domaine	Cache multi-domaine
Simplicité & maintenance des dev	★★★	★★	★★	★
Performance	★	★	★★★	★★★
Continuité, autonomie		★★	★★	★★★
Synchronisation référentiel / front	★★★	★	★	★

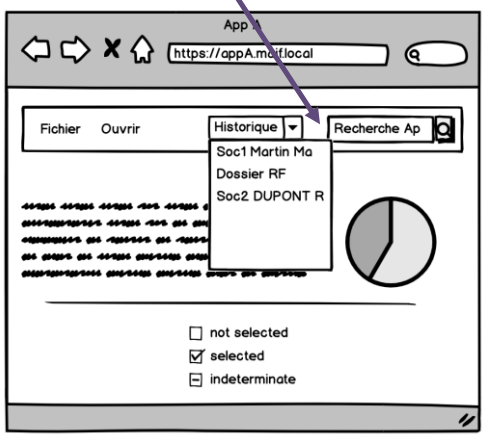
Validation Attendue

Préconisation projet : mise en place du cache si impacts performances sur l'UX

Old slides & annexes

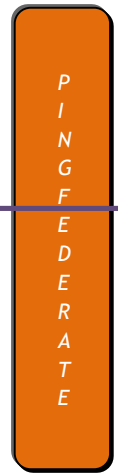
Architecture des livrables DARWIN

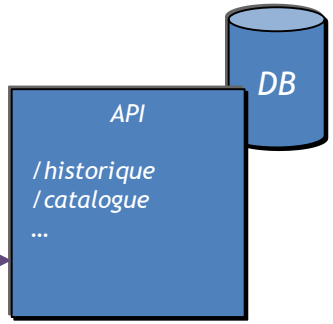
Livraison de composants pour accéder :
- Aux catalogues d'applications
- A l'historique
- Au applications favorites



Librairies JS pour accéder aux API

Authentification aux API via PingFederate à l'identique des backend applicatifs





Des APIs pour gérer :
- L'historique
- Les applications
- Les droits
- Les tenants (prod/preprod/…)

Source App A

```
if (docucument.be) {
  document.be.launch(…)
} else {
  darwin.launch(…)
}
```

- Librairies JS pour remplacer les appels au BE.
- Procédures pour rendre compatible une application dans et hors BE

Scénario d'authentification

1. Accès à App AA qui redirige vers PingFederate car l'utilisateur n'est pas authentifié

2. Le brower envoi une authentification Kerberos à PingFederate ou ce dernier propose une fenêtre login/password

3. PingFederate valide l'autentification et renvoie un code en positionnant également un cookie

4. Le browser envoi le code à App AA qui vérifie auprès de PF que celui-ci est valide.

5. Le front App AA est chargé dans le browser et les librairies Darwin sont exécutés.

6. Le client Javascript Darwin tente de s'authentifier auprès de PF en envoyer le précédent cookie. PF le valide et envoi automatiquement un code correspondant au backend Darwin

7. Le code est renvoyée au backend darwin pour validation (idem 4) et ce dernier renvoie un token. App AA accède maintenant aux 2 backends