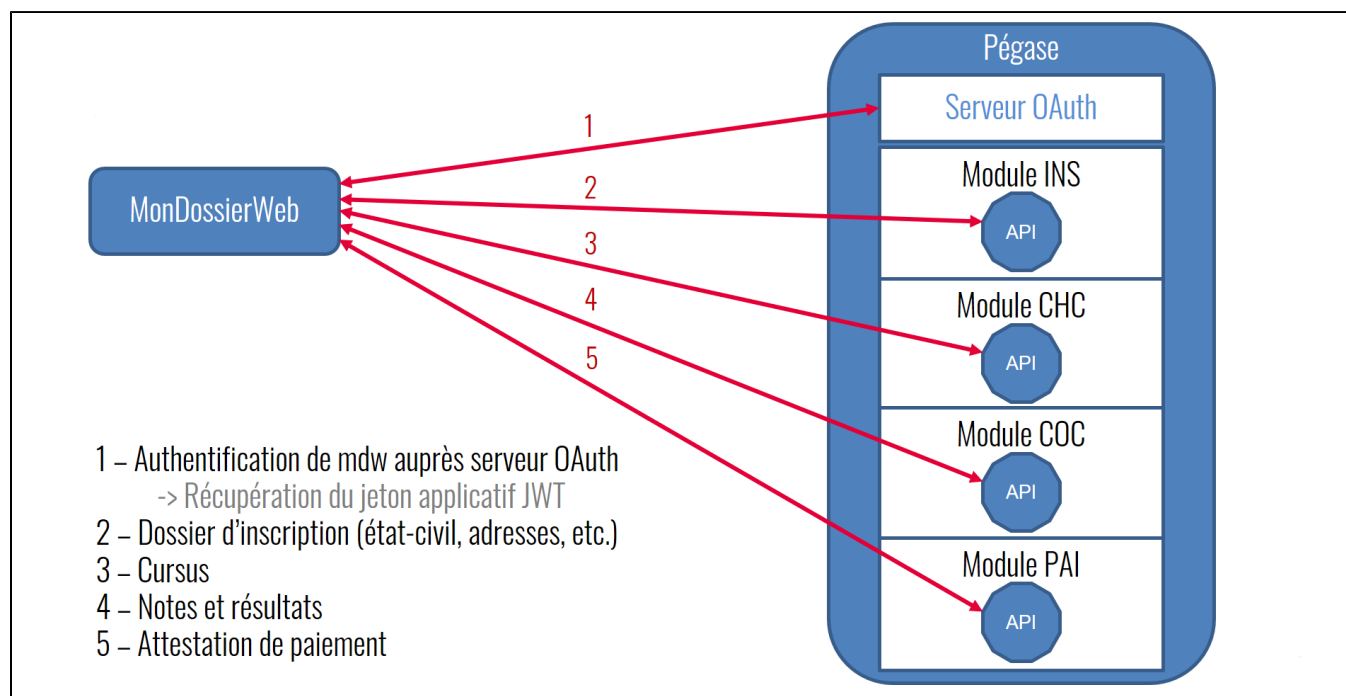


MDWP -2- APIs Pégase

- Architecture
- APIs utilisées
 - Exemple de configuration
 - Méthodes utilisées
- Tester une API
 - Récupération du token
 - Appel de l'API
- Génération des classes clientes java
 - Installation d'OpenAPI Generator
 - Génération du client java
 - Récupérer le code généré dans le projet
 - Modification des classes générées
 - Implémentation

Architecture

MDW est autorisé à interroger les APIs Pégase en s'identifiant à l'aide d'un token (JWT). Ce dernier est obtenu en s'authentifiant (username + password) auprès du serveur OAuth dédié.



Ce token a une durée de vie limitée. MDW garde donc ce token en mémoire et vérifie régulièrement sa validité afin de le renouveler si nécessaire.



Le paramètre `pegase.accesstoken.duration` (application.properties) permet d'indiquer la durée (en heure) de conservation du token.

APIs utilisées

Exemple de configuration

Voici un exemple des URLs à configurer dans la vue "Configuration" de l'application

API	variables	URL exemple
OAuth	url_authn_app_tickets	https://authn-app.univ.pc-scol.fr/cas/v1/tickets

INS EXT	url_api_ins_ext	https://ins.univ.pc-scol.fr/api/ins/ext/v1
INS	url_api_ins	https://ins.univ.pc-scol.fr/api/v5/ins
CHC	url_api_chc	https://chc.univ.pc-scol.fr/api/chc/v6
COC	url_api_coc	https://coc.univ.pc-scol.fr/api/coc/publication/v2
PAI	url_api_pai	https://pai.univ.pc-scol.fr/api/v1

Méthodes utilisées

URL (voir variables ci-dessus)	Module /API	Méthode	Utilisation
url_authn_app_tickets	serveur OAuth		Récupération du Jeton JWT pour authentification des API
url_api_ins_ext/gestion/inscription/\${etab}/\${codeApprenant}	INS EXT	lireInscriptions	Récupération du dossier de l'apprenant et de ses inscriptions
url_api_ins/gestion/inscriptions/\${etab}/\${codeApprenant}/\${codeVoeu}/certificat-de-scolarite	INS	imprimerCertificatDeScolarite	Récupération du certificat de scolarité
url_api_ins_ext/gestion/inscriptions/\${etab}/\${codeApprenant}/\${codeVoeu}/\${codePiece}/contenu	INS EXT	contenuPiece	Récupération de la photo de l'étudiant
url_api_chc/cursus-dca?codeApprenant={codeApprenant}	CHC	lireCursusApprenant	Récupération du cursus
url_api_coc/etablissements/\${etab}/periodes/\${codePeriode}/apprenants/\${codeApprenant}/chemins/\${chemin}	COC	listerCursusPubliableApprenant	Récupération des notes et résultats de l'étudiant
url_api_pai/pai/attestation-de-paiement/\${etab}/\${codeApprenant}/\${codePeriode}	PAI	imprimerAttestationDePaiement	Récupération de l'attestation de paiement

Tester une API

Nous allons voir comment tester le bon fonctionnement d'une API en la requêtant "à la main".



Plutôt que d'utiliser la commande "curl" pour exécuter les requêtes http, il est possible d'utiliser un outil comme [Insomnia](#) afin de se faciliter la tâche.

Appeler une API s'effectue via une requête HTTP.

Pour être autorisé à appeler l'API d'un module Pégase, il faut préalablement avoir récupéré un token d'authentification.

Par exemple pour tester la méthode lireInscriptions de l'API INS (qui retourne la quasi totalité du dossier), il faut :

1. Faire un premier appel (POST) au serveur OAuth pour récupérer le token
2. Utiliser ce token pour appeler l'API (GET)

Récupération du token

```
POST https://authn-app.[univ].pc-scol.fr/cas/v1/tickets?username=[username]&password=[password]
```

Le username et le password à passer sont ceux permettant d'identifier MDW auprès du serveur OAuth de Pégase.

Ces informations correspondent aux paramètres suivant dans **application.properties** :

- pegase.accesstoken.url
- pegase.accesstoken.username
- pegase.accesstoken.password

Appel de l'API

```
GET https://ins.[univ].pc-scol.fr/api/v5/ins/gestion/inscription/[codeEtablissement]/[codeApprenant]/
```

Ajouter la chaîne "Bearer " (attention à bien ajouter l'espace) au début du token et le passer dans un header nommé "Authorization".



Via Postman le passage du token se fait facilement depuis l'onglet "Authorization" (Type "OAuth 2.0")

L'url de l'API et le code établissement correspondent aux paramètres suivant dans **application.properties** :

- pegase.api.ins.url
- pegase.etablissement

Génération des classes clientes java

MDW embarque directement dans son code les classes java clientes utilisées pour interroger les APIs. Il n'y a pas de dépendance vers une librairie externe.

Pour générer ces classes java, OpenAPI Generator est utilisé depuis la v1.

Dans un soucis de transparence et afin de permettre de rejouer cette procédure (normalement pas nécessaire), voici, ci-dessous, la démarche utilisée lors du développement.

Installation d'OpenAPI Generator

Télécharger la version zip de node (<https://nodejs.org/en/download/>) et dézipper puis :

```
# Se rendre dans le répertoire d'installation
cd E:/node-v14.16.0-win-x64

# Installer openapi-generator-cli
npm install @openapitools/openapi-generator-cli
npm install @openapitools/openapi-generator-cli -g

# Forcer l'utilisation de la version 5.4.0
# /!\ IMPORTANT /!\ A vérifier avant chaque génération sous peine d'obtenir des imports non valides
openapi-generator-cli version-manager set 5.4.0

# Aide sur la config
npx @openapitools/openapi-generator-cli config-help -g jav
```

Génération du client java

Exemple pour générer les classes de clients java des principales API :

Copier les **YML descripteurs des APIs Pégase** dans **C:\tmp\YML** et lancer depuis le répertoire où se trouve le **YAML** une génération des classes java :

Génération des clients java à partir des yaml

```
# Forcer l'utilisation de la version 5.4.0
# /\ IMPORTANT /\ A vérifier avant chaque génération sous peine d'obtenir des imports non valides
openapi-generator-cli version-manager set 5.4.0

# INS EXT YAML
npx @openapitools/openapi-generator-cli generate -i inscription-ext-api-v1-1.2.0.yaml -g java -o C:/tmp/openapi-generator/ -p apiPackage=fr.univlorraine.pegase.api.insext -p modelPackage=fr.univlorraine.pegase.model.insext -p dateLibrary=java17-localdatetime

# INS GESTION YAML
npx @openapitools/openapi-generator-cli generate -i ins-gestion-api-v5-20.0.0.yaml -g java -o C:/tmp/openapi-generator/ -p apiPackage=fr.univlorraine.pegase.api.insgestion -p modelPackage=fr.univlorraine.pegase.model.insgestion -p dateLibrary=java17-localdatetime

# CHC YAML
npx @openapitools/openapi-generator-cli generate -i chc-application-api-v6-6.2.0.yaml -g java -o C:/tmp/openapi-generator/ -p apiPackage=fr.univlorraine.pegase.api.chc -p modelPackage=fr.univlorraine.pegase.model.chc -p dateLibrary=java17-localdatetime

# COC PUBLICATION YAML
npx @openapitools/openapi-generator-cli generate -i coc-publication-api-v2-2.0.0.yaml -g java -o C:/tmp/openapi-generator/ -p apiPackage=fr.univlorraine.pegase.api.coc -p modelPackage=fr.univlorraine.pegase.model.coc -p dateLibrary=java17-localdatetime

# PAI YAML
npx @openapitools/openapi-generator-cli generate -i pai-api-v1-21.0.0.yaml -g java -o C:/tmp/openapi-generator/ -p apiPackage=fr.univlorraine.pegase.api.pai -p modelPackage=fr.univlorraine.pegase.model.pai -p dateLibrary=java17-localdatetime
```

Récupérer le code généré dans le projet

Récupérer les classes générées dans les répertoires :

- `/src/main/java/fr/univlorraine/pegase/api`
- `/src/main/java/fr/univlorraine/pegase/model`

Modification des classes générées

Il peut être nécessaire d'apporter des modifications au code généré.

Exemple :

- Corriger l'assignation aux énumérations dans les constructeurs des classes qui le nécessite. Par exemple, le code :

```
this.canalCommunication = this.getClass().getSimpleName();
```

devient

```
this.canalCommunication = CanalCommunicationEnum.fromValue(this.getClass().getSimpleName());
```

- Supprimer l'attribut **est1** et la constante **SERIALIZED_NAME_EST1** de la classe **Periode**
- Remplacer dans les classes qui le nécessite, le code :

```
@javax.annotation.Generated
```

par

```
@jakarta.annotation.Generated
```

Implémentation

Pour des exemples d'appels aux APIs effectués grâce aux classes générées ci-dessus, voir le code source du PegaseService sur github : <https://github.com/EsupPortail/esup-mdw-pegase/blob/master/src/main/java/fr/univlorraine/mondossierweb/service/PegaseService.java>