Dauphine | PSL
UNIVERSITÉ PARIS

MIDO-SVG

Groupe :
BRULE Jérémy
DOUR Marcellino
GUO Zhenyi
LANGLOIS Camille
PARDINI Raphaël
TAJOURI Sarra

L3 MIAGE – JAVA/UML – Présentation projet

03/07/2020

Enseignant encadrant :
CAILLOUX Olivier

Dauphine | PSL UNIVERSITÉ PARIS

Faisons connaissance …

Bonjour !

La cible : Un enseignant extérieur ou débutant ou encore les étudiants

Le besoin: Représenter graphiquement l'offre de formation MIDO

Le problème: Les représentations graphiques deviennent rapidement obsolètes

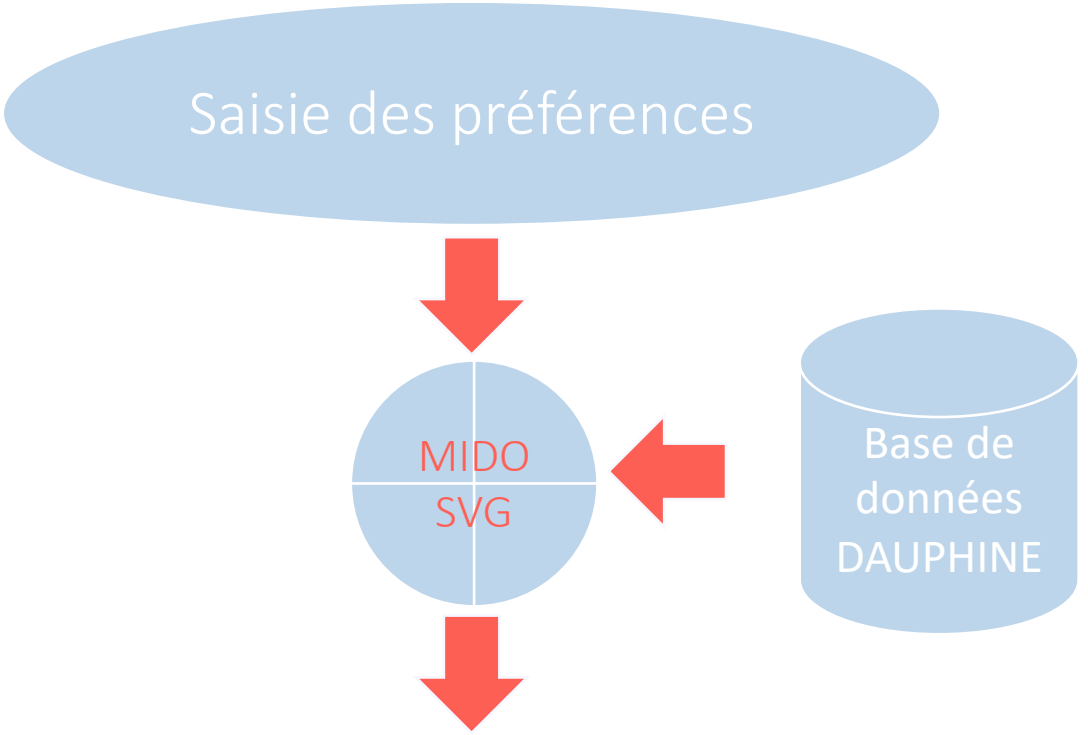
La solution: L'application MIDO-SVG propose de générer automatiquement des images graphiques représentant les formations de l'université et les différents liens entre elles

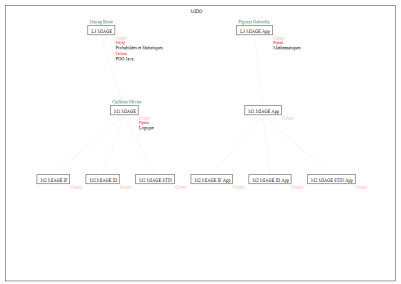
Concrètement, comment ça marche ?

1- Saisie des préférences de l'utilisateur:
- Le format
- Les éléments à afficher

2-Récupération de données de la base de données de Dauphine

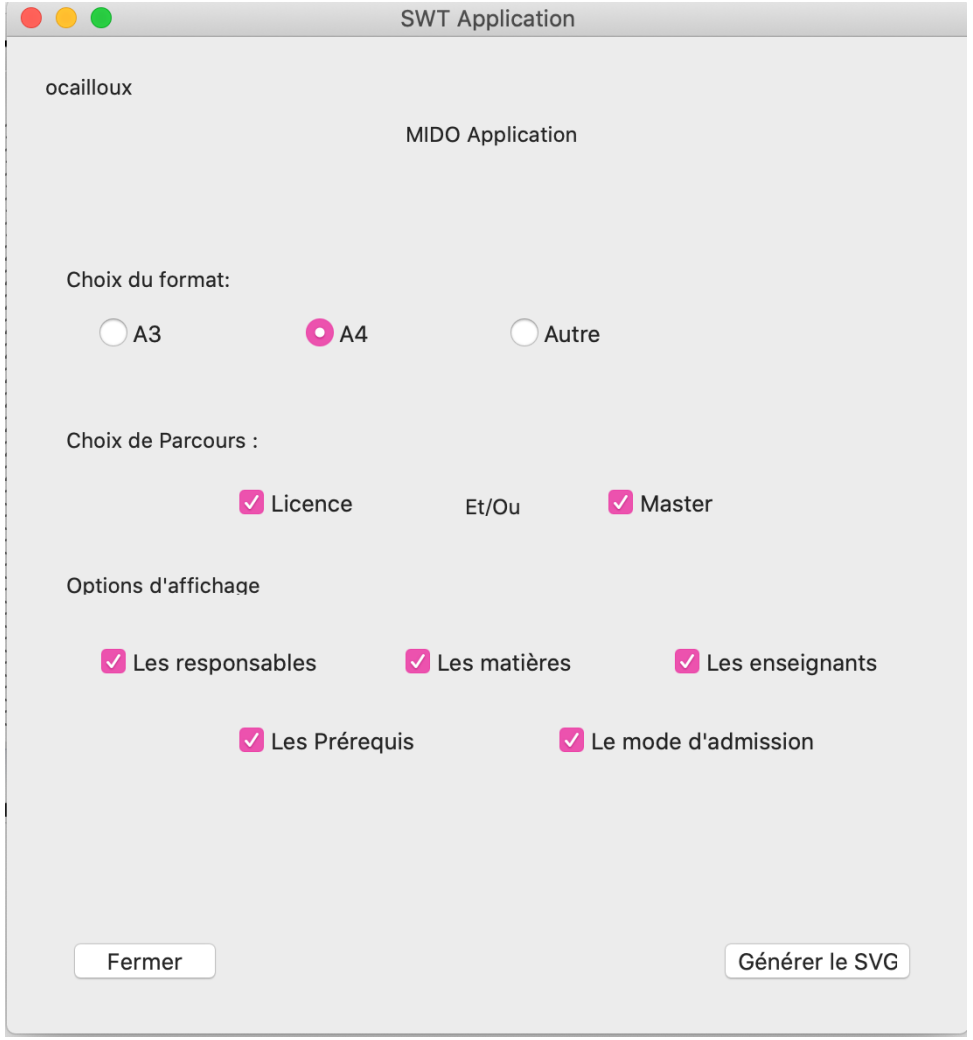
3- Génération d'un graphique sous format SVG





Dauphine | PSL UNIVERSITÉ PARIS

Concrètement, comment ça marche ?



Dauphine | PSL UNIVERSITÉ PARIS

Quels sont les outils utilisés ?

Développement

eclipse

Maven™

GitHub

Papyrus

Communication

Drive

Microsoft Teams

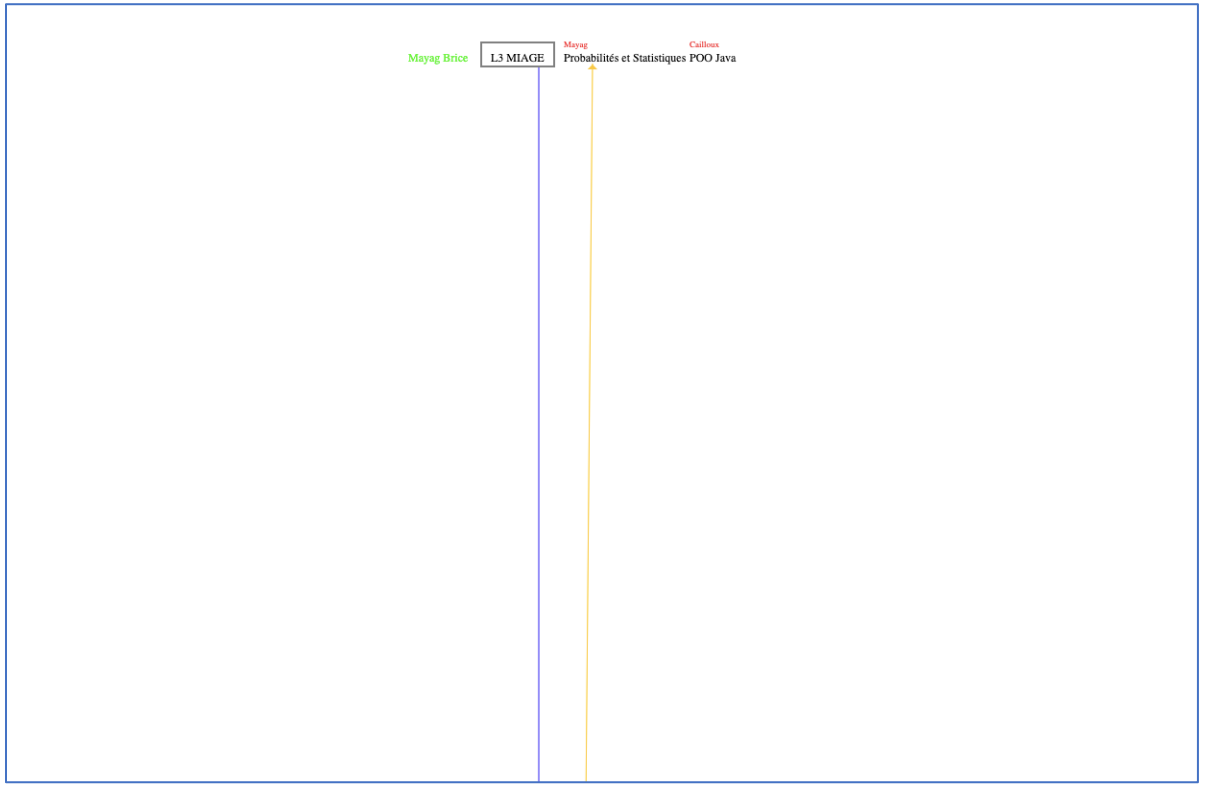
WhatsApp

La génération d'un fichier SVG

A la récupération du projet

Le SVG se génère mais est illisible :

- Il tient sur plus de quatre pages, chaque formation est très espacée des autres.
- Lorsqu'on ne sélectionne ni Licence, ni Master, le résultat parait aléatoire.
- Les cours sont écrits sur une même ligne, sans espace.
- Et beaucoup d'autres détails donnent un rendu médiocre.



Exemple : une partie du SVG obtenu avec le code du début d'année.

La génération d'un fichier SVG

Notre mission

Pour résoudre cela, nous nous sommes fixé plusieurs missions :

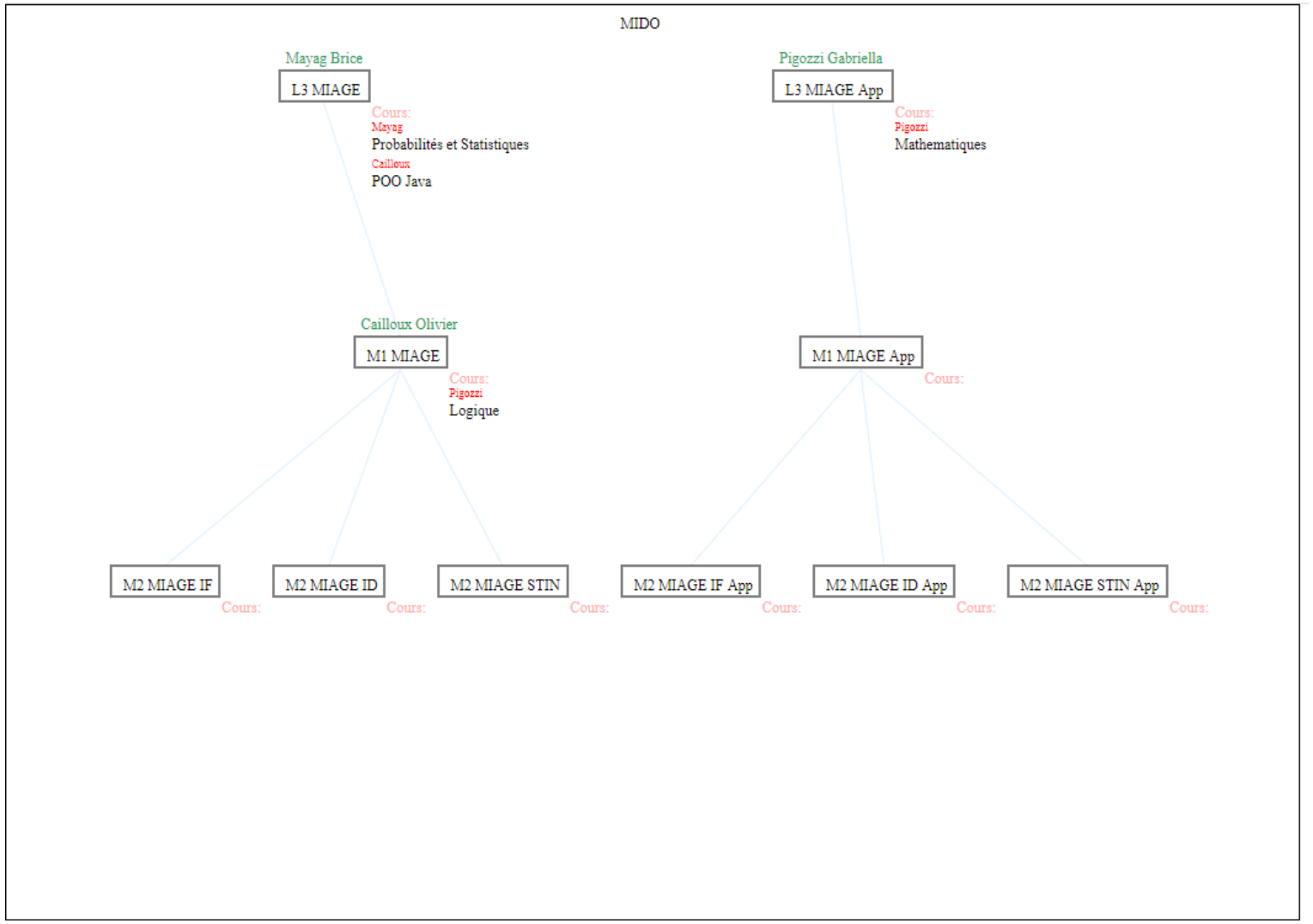
→ Respecter le format choisi par l'utilisateur (A3, A4…)

→ Rendre l'affichage des cours plus lisible

→ Rendre le SVG plus «responsive », en adaptant la positions des formations en fonction des critères choisis.

→ Résoudre d'autres détails afin de parfaire l'aspect visuel de notre application.

La génération d'un fichier SVG



La génération d'un fichier SVG

Nos difficultés rencontrées

1. Reprise et compréhension d'un code déjà **existant**.

2. Recherche du **calcul** des positions assez **complexe** et laborieux car il doit prendre de nombreux critères en compte.

3. Difficultés techniques sur des notions non vues en cours (affichage…)

La récupération d'une base de données externe

A la récupération du projet

Après avoir fait un état des lieux rapide sur l'existant de la « base de données », nous nous sommes rapidement rendu compte que la génération d'un graphe SVG se base sur des données inscrites « **en dur** » dans l'application.

Issu de la classe « *Database* », nous y trouvons un ensemble d'objets composant une université (un *Department*, une liste de *Formations*, une liste de *Subjects*, une liste de *Teachers*,…), ensemble d'objet que l'on retrouve dans le package *University.*

L'instanciation d'un objet *Database* se fait par un appel constructeur :

```
/** we create the DataBase **/
DataBase datas = new DataBase(settings);
```

```
// List of formations
this.formations = new LinkedList<>();
this.formations.add(L3MIAGE);
this.formations.add(L3MIAGEApp);
this.formations.add(M1MIAGE);
this.formations.add(M1MIAGEApp);
this.formations.add(M2MIAGEIF);
this.formations.add(M2MIAGEID);
this.formations.add(M2MIAGESTIN);
this.formations.add(M2MIAGEIFApp);
this.formations.add(M2MIAGEIDApp);
this.formations.add(M2MIAGESTINApp);

// Available formation
L3MIAGE.addAvailableFormation(M1MIAGE);
L3MIAGEApp.addAvailableFormation(M1MIAGEApp);
M1MIAGE.addAvailableFormation(M2MIAGESTIN);
M1MIAGE.addAvailableFormation(M2MIAGEIF);
M1MIAGE.addAvailableFormation(M2MIAGEID);
M1MIAGEApp.addAvailableFormation(M2MIAGESTINApp);
M1MIAGEApp.addAvailableFormation(M2MIAGEIDApp);
M1MIAGEApp.addAvailableFormation(M2MIAGEIFApp);

this.formationsMap.put("L3MIAGE", L3MIAGE);
this.formationsMap.put("M1MIAGE", M1MIAGE);
this.formationsMap.put("L3MIAGEApp", L3MIAGEApp);
this.formationsMap.put("M1MIAGEApp", M1MIAGEApp);
```

De plus, le remplissage des données se faisait par *simple* ajout d'élément dans une liste.

Ici, nous avons relevé 3 axes sensibles :

- Une BdD fragile, non sécurisée de part l'aspect « hard-coded ».
- Une BdD rigide, peut être sujet à une maintenance conséquente et redondante en cas d'évolution BdD
- Une BdD incomplète, n'apparaît qu'un faible échantillon de données

A noter que ces l'existant témoigne certainement d'une volonté aux prédécesseur à poser réflexion sur d'autres aspects du projet. (ex: GUI, XML, …)

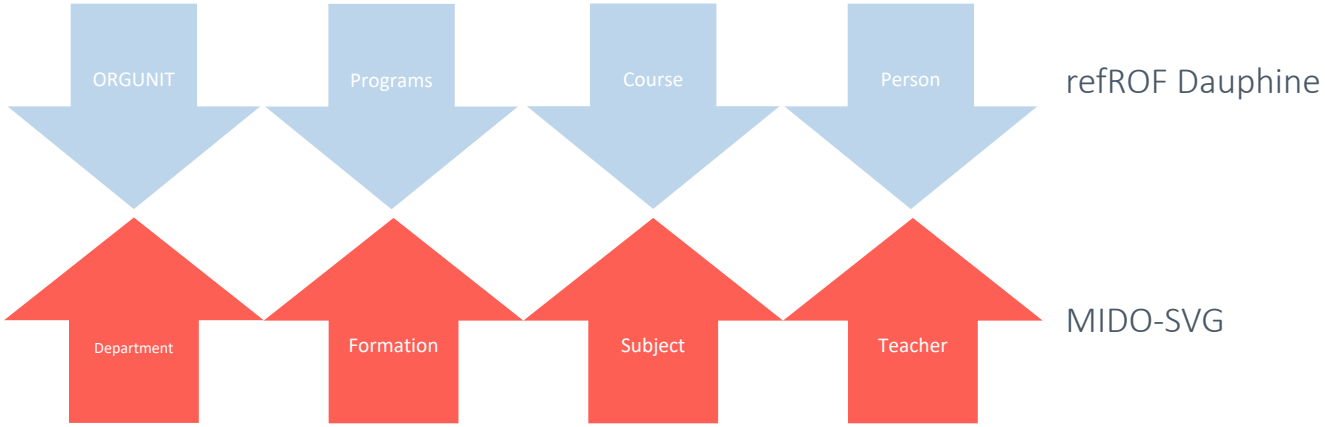
La récupération d'une base de données externe

Notre solution

Dans le but de pallier à la fragilité, la rigidité et la partialité de la base de données, nous avons missionné la tâche « Base de données » à :

L'implémentation d'une une classe orientée objet, similaire à la classe Database issue de données récupérées sur l'API refROF Dauphine.

Le premier exercice avant d'implémenter cette extension et de déterminer si la base de données détient les informations recherché.
Une période d'exploration a été nécessaire à la bonne exécution de la mission.
Par ce biais, nous avons pu ressentir des correspondances structurelles entre la base de données refROF Dauphine et l'application MIDO-SVG.



Procédure de récupération d'une base de données externe :

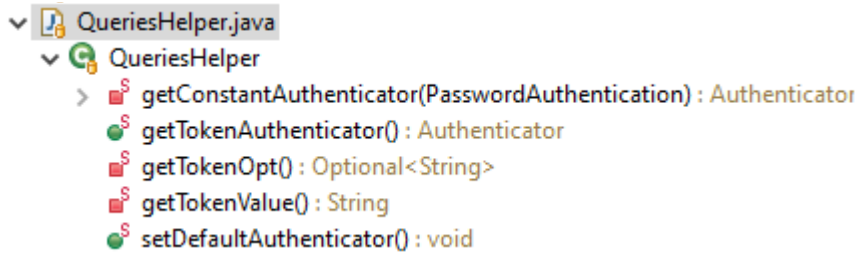
La récupération d'une base de données externe

Nos missions : La connexion

Après avoir finaliser l'exploration de l'API BdD Dauphine, la première étape dans l'implémentation d'une connexion flexible de notre application à cette dernière est bien évidemment la mise en place d'une procédure d'authentification.

C'est avec l'aide de notre enseignant encadrant du projet, que nous avons pris connaissance d'un projet existant procédant à notre volonté de connexion : Plaquette-MIDO (url : https://github.com/Dauphine-MIDO/plaquette-MIDO)

Un identifiant et un mot de passe utilisateur ont été essentiel dans la connexion à l'API refROF Dauphine.
Pour ce faire, une classe *QueriesHelper* a été nécessaire à ce processus d'identification.



Dans cette classe, par sécurité, la méthode getTokenValue() permet de récupérer le mot de passe inscrit dans un fichier externe nommé « API_password.txt » placé à la racine du projet.

L'identifiant de l'utilisateur est quant à lui renseigné en dur dans la méthode getTokenAuthenticator().

La connexion à la base de données se fait par : `QueriesHelper.setDefaultAuthenticator();`

La récupération d'une base de données externe

Nos missions : La récupération + transformation

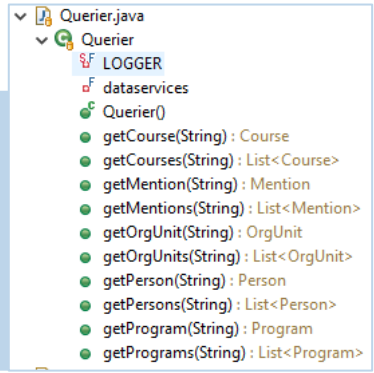
A la suite d'une implémentation de la classe Database avec connexion API, vient l'implémentation des méthodes de récupération des données et de sa transformation en objet compatible avec le projet MIDO-SVG.

Pour la récupération des données, il a été nécessaire d'importer une bibliothèque externe renseignant la structure de la base de données de l'API.
C'est par le biais d'une dépendance MAVEN que nous avons mis en place l'intégration cette structure dans notre projet.

```
<build>
    <plugins>
        <plugin>
            <groupId>com.sun.xml.ws</groupId>
            <artifactId>jaxws-maven-plugin</artifactId>
            <version>2.3.2-1</version>
            <executions>
                <execution>
                    <goals>
                        <goal>wsimport</goal>
                    </goals>
                    <configuration>
                        <wsdlUrls>https://rof.testapi.dauphine.fr/ebx-dataservices/tables/pvRefRof/RefRof?WSDL</wsdlUrls>
                        <xauthFile>WSDL_access.txt</xauthFile>
                        <vmArgs>
                            <vmArg>-Djavax.xml.accessExternalSchema=all</vmArg>
                        </vmArgs>
                    </configuration>
                </execution>
            </executions>
        </plugin>
```

Remarquons qu'il a été indispensable de renseigner, ici aussi, un identifiant et un mot de passe utilisateur.
Ce fichier est disponible aussi à la racine du projet.

C'est à l'aide de la classe Querier.java que nous avons pu récupérer les données et ses attributs dans l'objectif de créer un objet d'un composant University à partir des données API.



La récupération d'une base de données externe

Nos missions : L'instanciation de l'objet database

```
public class RofDatabase {

    private Department department;
    private ImmutableSet<Formation> formations;
    private ImmutableSet<Subject> subjects;
    private ImmutableMap<Subject,String> tags;
    private ImmutableMap<String, Teacher> teachers;

    static public final String MENTION_MIDO_IDENT = "?";

    /**
     * Initialize an immutable Database with ROF informations.
     * @return RofDatabase
     * @throws Exception
     */
    public static RofDatabase initialize() throws Exception {
        return new RofDatabase();
    }

    private RofDatabase() throws Exception{

        QueriesHelper.setDefaultAuthenticator();
        this.department = fetchDepartment();
        this.formations = fetchFormations();
        this.teachers = fetchTeachers();

    }
```

Se basant sur l'existant, les attributs de la classe Database ont été allégés, ne gardant que les attributs utiles à son attendu.

Appliquant la méthode factory, l'instanciation de l'objet database se fait comme suit :

```
RofDatabase rof = RofDatabase.initialize();
```

L'instanciation d'un objet Database suit les étapes précédemment définies.

1. La connexion
2. La récupération des données ainsi que sa transformation

Par effet cascade, la récupération des formations englobe la récupération des cours.

La récupération d'une base de données externe

Nos difficultés rencontrées

1

Accès et compréhension de la base au format JSON ⇨ **JSON Viewer** a beaucoup simplifié la phase d'exploration

2

La base à laquelle on a accès est en **environnement de test** rendant ainsi nos tests fragiles puisque des données peuvent être supprimées, ce qui a d'ailleurs était le cas. Les formations de MIDO ont disparu en cours de route et nous avons dû travailler sur d'autres formations. Une façon de pallier à cela est de récupérer toutes les *Mention* et de chercher dans leurs noms des mots-clés correspondant à MIDO notamment Informatique, Mathématiques… Sauf que les méthodes de Querier.java nous permettent de récupérer des données uniquement à partir d'une clé et il reste à identifier cette clé qui correspondra aux mentions.

3

Ecart de données entre API et objet dans MIDO-SVG: quelques valeurs sont à *null* alors que nous voulonq éviter à tout prix de se retrouver avec de telles valeurs dans notre base de données. Notamment les tags, on avait compris que ça correspondait aux *searchword* des *Course* dans ROF mais ce champ est toujours *null.*

⇨ Tester si c'est *null* avant de récupérer les données sinon mettre à vide ou encore à *nan* pour dire que ce n'est pas mentionné.

Conclusion

Difficultés rencontrées:

- Organisation (Temporel, papyrus, partage d'info)
- La compréhension du périmètre (Java et packages liés, versioning)

Points positifs :

- Développement de compétences organisationnelles
- Développement de compétences techniques

Merci à tous et à très bientôt !