**Documentation technique Serveur**

L’application « dap » est la couche applicative permettant de récupérer directement les informations demandées vers l’API Google (emails non lus, calendrier, contacts) ainsi que vers l’API Microsoft (emails non lus, corps des 3 premiers mails, calendrier ainsi que les contacts).

On retrouve 9 packages différents qui peuvent paraître énorme pour un projet, mais qui enfaite permet de découper la modèle MVC pour chaque API (Google Et Microsoft) ainsi qu’un modèle MVC pour une partie permettant de regrouper les résultats des deux API (Google et Microsoft).

L’avantage de ce système est que si demain, nous avons besoin de faire le lien vers une nouvelle API pour récupérer des données, il nous suffirait de créer un nouveau package et intégrer le modèle MVC à l’intérieur. Ceci pour toujours garder une architecture claire et précise pour n’importe quel développeur venant à intégrer le projet en cours.

* *Pour Google*
  + **fr.ynov.dap.configuration** : Regroupe toutes les classes en lien avec la configuration de l’application ou bien des configurations de certains fichiers comme le dossier « tokens » ou le « storedCredential » (concernant Google).
  + **fr.ynov.dap.controller** : Regroupe tous les controllers pour Google
  + **fr.ynov.dap.service** : Regroupe tous les services pour Google
  + **fr.ynov.dap.repository** : Regroupe tous les repository pour Google
  + **fr.ynov.dap.model** : Regroupe les différents model pour les différents API. Dans notre cas, EventModel permet de comparer un évènement entre Google et Microsoft.
  + **fr.ynov.dap.data** : Regroupe les différents data pour la bonne interaction avec l’API Google (Entités, classe pour stocker le retour Json, etc.). Dans notre cas, ce sont 2 entités.
* *Pour Microsoft*
  + **fr.ynov.dap.microsoft** : Regroupe tous les fichiers permettant d’interagir avec Microsoft, c’est-à-dire en récupérant les tokens, refresh token, ainsi que le AuthHelper pour la mise en place de l’authentification d’un compte Microsoft.
  + **fr.ynov.dap.controller** : Regroupe tous les controllers pour Microsoft
  + **fr.ynov.dap.service** : Regroupe tous les services pour Microsoft
  + **fr.ynov.dap.repository** : Regroupe tous les repository pour Microsoft
  + **fr.ynov.dap.data** : Regroupe les différents data pour la bonne interaction avec l’API Microsoft (Entités, classe pour stocker le retour Json, etc.).
* *Global (Microsoft + Google)*
  + **fr.ynov.dap.controller** : Regroupe tous les controllers pour le Global
  + **fr.ynov.dap.service** : Regroupe tous les services pour interagir avec les services de Microsoft et Google

**Les routes du serveur**

Lancer le serveur puis taper dans l’URL : <http://localhost:8080> (si aucune modification a été faite côté Config)

*PS : userKey = nom de l’utilisateur et accountName = pseudo donné pour lier un compte à un userKey.*

ROUTES GOOGLE

**La route pour ajouter un utilisateur :**

/user/add/{userKey}

**La route pour ajouter un compte Google à l’utilisateur :**

/google/account/add/{userKey}/{accountName}

**Les routes pour contacter l’API Google :**

Nombre de mails non lus : /google/mail/{userKey}

Prochain évènement : /google/event/{userKey}

Nombre de contacts : /google/contact/{userKey}

ROUTES MICROSOFT

**La route pour ajouter un compte Microsoft à l’utilisateur :**

Ajout utilisateur : /account/add/microsoft/{userKey}/{accountName}

Succès après ajout : /user/success

**Les routes pour contacter l’API Microsoft :**

Nombre de mails non lus de l’utilisateur actuellement connecté : /microsoft/mail

Nombre de mails non lus par l’userKey passé en paramètre : /microsoft/mail/{userKey}

Prochain évènement de l’utilisateur actuellement connecté : /microsoft/events

Prochain évènement de l’userKey passé en paramètre : /microsoft/events/{userKey}

Nombre de contacts de l’utilisateur actuellement connecté : /microsoft/contacts

Nombre de contacts de l’userKey passé en paramètre : /microsoft/contacts/{userKey}

Se déconnecter : /logout

Index : /

**Les routes pour retourner le cumul des données de Google et Microsoft :**

Admin pour afficher le tableau des credentials : /admin

Nombre de mails non lus par l’userKey passé en paramètre : /mail/global/{userKey}

Prochain évènement de l’userKey passé en paramètre : /event/global/{userKey}

Nombre de contacts de l’userKey passé en paramètre : /contact/global/{userKey}

**Base de données**

La base de données utilisé est une base MySQL. Pour ma part, j’utilise WampServer et je peux atteindre la base de puis cette URL : <http://localhost/phpmyadmin>

Nom de la BDD : JavaProject

Username : root

Password : (vide)

*Tous ces paramètres peuvent être directement changés dans le fichier : src/main/resources/application.properties*

**ZeroConf**

De base :

- Le "storedCredential" se trouve de base dans : C:\Users\dap\tokens

- Le "authProperties" pour Microsoft se trouve de base dans : C:\Users\dap\microsoft

Pour modifier l'emplacement de storedCredential :

- Faire un setteur sur l'objet Config pour la propriété tokensDirectoryPath pour pouvoir mettre le dossier tokens ailleurs voire même le mettre à jour si on l’a renommé manuellement.

- Faire un setteur sur l'objet config pour la propriété credentialsFilePath pour pouvoir mettre le fichier storedCredential ailleurs que dans le dossier "tokens"

Pour modifier l'emplacement de authProperties :

- Faire un setter sur l'objet AuthHelper pour la propriété authConfigFile pour pouvoir mettre le fichier authProperties à un autre emplacement.

**Javadoc**

La Javadoc est présente dans le dossier *doc* qui est situé à la racine du projet intégrant toute la javadoc de l’application « dap ».

**Problème rencontré**

Pour ma part, lorsque je voulais ajouter un compte microsoft, seul les comptes étant dans le domaine « @ynov.com » avait le champ « user\_email » renseigné dans la base de données. Tous les autres comptes avec un domaine différent à celui du dessus, étaient à null ce qui me bloquait par la suite.