Projet JXS - JXW :
Web côté client - serveur

Rédigé par
Julien DURAND
Louis LEDOUX
Olivier PEURICHARD

Vue d’ensemble de l’application

Architecture

Notre application se découpe en 2 parties :

- un serveur qui sert d’interface entre le client et les API des différents drives,
- un client qui sert d’interface avec l’utilisateur.

De plus, nous avons décidé d’implémenter les différentes fonctionnalités de l’application du côté du client.

Fonctionnalités existantes

bien que nous n'ayons pas eu le temps d'implémenter tout ce que nous nous étions fixé, plusieurs fonctionnalités sont disponibles :

- S’authentifier(Dropbox, Google Drive)
- Merge les dossiers communs(Dropbox, Google Drive)
- Supprimer des fichiers / dossiers (Dropbox, Google Drive)
- Créer des fichier / dossiers (Dropbox, Google Drive)
- Copier et coller des fichiers / dossiers (Dropbox)
- Partager un fichier (Dropbox)
- Renommer un fichier / dossier (Dropbox)
- Upload un fichier(Dropbox)
- Download un fichier(Dropbox)

Fonctionnalités non implémentées

Du fait de difficultées rencontrées que nous avions mal anticipé, certaine fonctionnalités n’ont pas eu le temps d’être implémentées :

- Copier / Coller des fichiers sur Google Drive
- Upload / download un fichier sur Google Drive
- Renommer sur Google Drive
- Créer un fichier ou un dossier sur Google Drive

Le front end

Pour le côté client, nous avons décidé d'utiliser le dernier langage vu en TP : Angular2/4.
Pour nous simplifier la tâche, et afin d'éviter de perdre trop de temps sur l'aspect esthétique, nous avons utilisé des composants Bootstrap.

- Architecture

Notre application se décompose en plusieurs composants. On peut les regrouper en plusieurs catégories : les vues et les services.
Les vues regroupent plusieurs éléments : le html qui décrira la page, et le fichier des fonctionnalités associées au html en question.
Les vues interagissent avec des services. Ces derniers sont l'interface entre le serveur et le client. On retrouve donc les différentes requêtes vers l'API du serveur.

De plus, à côté des précédents composants, on peut retrouver une classe "Element", qui est une représentation simplifiée des éléments manipulés : les fichiers et les dossiers.

Le backend

Démarche globale

En ce qui concerne le back-end, nous avons décidé de laisser sa majeure réalisation par la seule personne du groupe n'ayant fait que JXW et pas JXS pour garder de la cohérence. Le back-end dans ce projet est la passerelle entre le front-end et les services des drives que nous ciblons. Il faut donc créer ds services web sous forme d'api REST pour répondre au besoin du front, mais il faut également pouvoir requêter les api des drives ciblés (Google Drive, DropBox).

De nos jours plein de nos technologies permettent de réaliser ces aspects, il nous a donc fallu faire un choix. Comme le sujet parlait beaucoup du combo Java + Maven et comme nous avions utilisé ces technologies en TP nous nous sommes donc naturellement orientés vers ce couple Java + Maven.

Il a également fallu que nous choisissons une technologie pour héberger notre code sur un serveur, nous avons donc choisi Tomcat dans sa version 8.0. Ce choix n'a pas de raison particulière, nous avions testé avec cette version et cela marchait donc nous n'avons pas pris le temps de savoir s'il y avait de meilleurs solutions comme 8.5 ou 9.0 par exemple.

Une fois que nous avions cerné les technologies à mettre en place, le premier sprint côté back-end était de réussir à faire tourner du code java sur un serveur qui puisse répondre à des requêtes simples GET, POST et qui puisse requêter des pages web le tout en comprenant du JSON. Il faut effectivement rappeler que JSON n'est pas natif à Java, nous avons donc eu cette "barrière" en plus par rapport aux groupes qui ont opté pour un serveur orienté Node.JS.

Pour pouvoir faire tout cela, il y a heureusement des .jar qui existent et qui nous offrent tout ce que nous avions besoin. Pour bien incorporer ces .jar au projet, nous avons utilisé Maven, ce qui nous a en plus permis de gérer les versions de ceux-ci. On retrouve entre autre parmis les .jar les plus important que nous avons utilisé :

- jersey-multipart
- jersey-server
- jersey-servlet
- jersey-client
- jersey-json

Les noms sont assez explicites, on remarque qu'il y a de quoi faire les opérations pour agir en tant que client, en tant que serveur, de quoi faire sérialisation / désérialisation JSON/JAVA, de quoi faire de l'upload/download de fichiers en multipart.

Durant la réalisation du back, plusieurs problèmes sont intervenus à cause de versions de plugins (de .jar), et ce ne sont pas des erreurs faciles à détecter car elles surviennent souvent au run-time seulement quand le serveur est lancé avec des messages

d'erreur qui n'indiquent pas du tout que cela peut venir à cause d'un problème de version. Par exemple au début le multi-part était dans sa version 1.8 au début, pour qu'on comprenne enfin qu'il fallait le mettre en 1.9. Je pense que cela peut être l'une des difficultés de travailler avec un gestionnaire de version comme Maven, mais qu'au moins une fait que tout est mis en place, il est d'une facilité extrême d'exporter le projet sans qu'une tierce personne ne rencontre de problèmes dûs au versionning.

Le code et l'environnement de travail

Avant de parler du code et de son organisation, il peut être intéressant de voir quels logiciels nous avons utilisé côté back-end.

Nous avons utilisé Eclipse J2EE qui était vraiment très approprié à la situation. Cette version d'eclipse permet de gérer des serveurs Tomcat, il suffit de cliquer sur Start/Debug/Stop au lieu de le lancer en ligne de commande. On peut également exporter celui en .war en un seul clic. Et on peut naturellement comme toute version d'Eclipse profiter d'un bon IDE java, et nous avons de plus gérer les dépendances maven avec eclipse contrairement aux TPs, ce qui est presque la même chose en fin de compte.

Pour ce qui est du code, nous avons une classe par drive que nous voulons accéder. Chacune de ces classes propose les services adéquat au front-end et s'occupe de requêter les drives correspondants.

Chaque classe méthode, et chaque méthode contient donc une annotation Path qui corresponde à une extension de base à l'URL du serveur REST.

```
<url-pattern>/myWebService/* </url-pattern>
```

On retrouve ce code dans web.xml, on comprend alors que notre serveur s'ocuppe de catch toutes les URL : http://localhost/ServerREST/myWebService/* .
C'est après dans les classes java grâce à l'annotation Path que les bonnes méthodes sont appelées :

```
@SuppressWarnings("unchecked")
@Path("/DropBox")
public class DropBox {

    private static Client client = Client.create();
    private static final String APP_SECRET = "[redacted]";
    private static final String APP_KEY = "[redacted]";
    private static final String REDIRECT_URI = "http://localhost:8080/ServerREST/myWebService/DropBox/getCode";
    private static String _code = null;
    private static String token = null;


    @GET
    @Path("/connection")
    @Produces(MediaType.APPLICATION_JSON)
    public Response connect() {

        MultivaluedMap<String, String> formData = new MultivaluedMapImpl();
        formData.add("response_type", "code");
        formData.add("client_id", APP_KEY);
        formData.add("redirect_uri", REDIRECT_URI);
```

Dans cet exemple on voit le début de la classe DropBox, elle a donc alors une annotation Path(“/DropBox”), mais on voit également le début de la première méthode qui permet de lancer la procédure de connection Oauth2 pour récupérer le code, le token et de rediriger sur le front-end localhost:4200.

Pour se connecter il faut faire une Requête GET à : http://localhost:8080/ServerREST/myWebWervice/DropBox/conection. Pour Google, il suffit de remplacer DropBox par Google, on a alors gardé cette logique de chemins pour tous les services différents.

Manuel d’utilisateur

1. L'authentification

Il est possible pour l'utilisateur de se connecter sur les drives de son choix (dans la limite des propositions). Cette authentification est d'ailleurs, et il est important de le noter, une fonctionnalité clé de notre application. En effet, elle nous permet de récupérer les fichiers de l'utilisateur ainsi que toutes les métadonnées nécessaires aux autre fonctionnalités.

Lorsque l'utilisateur se connecte sur l'application, il arrive tout d'abord sur la page d'accueil (voir Fig.1). Cette page affiche à l'écran les différents drives auxquels il est possible de se connecter (ici Google Drive et Dropbox).

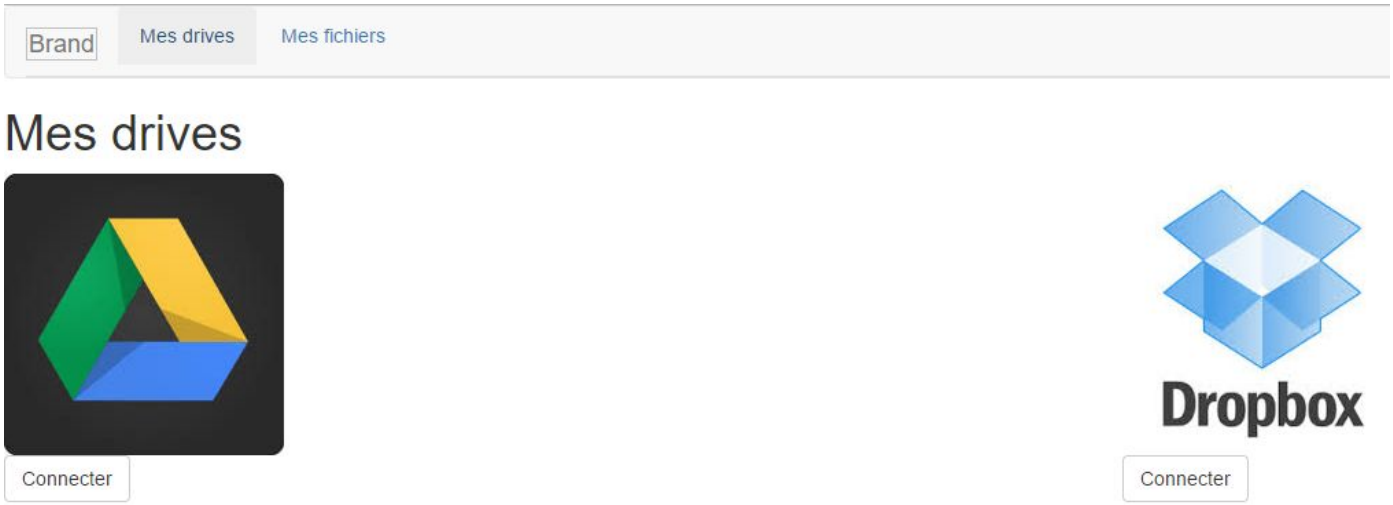
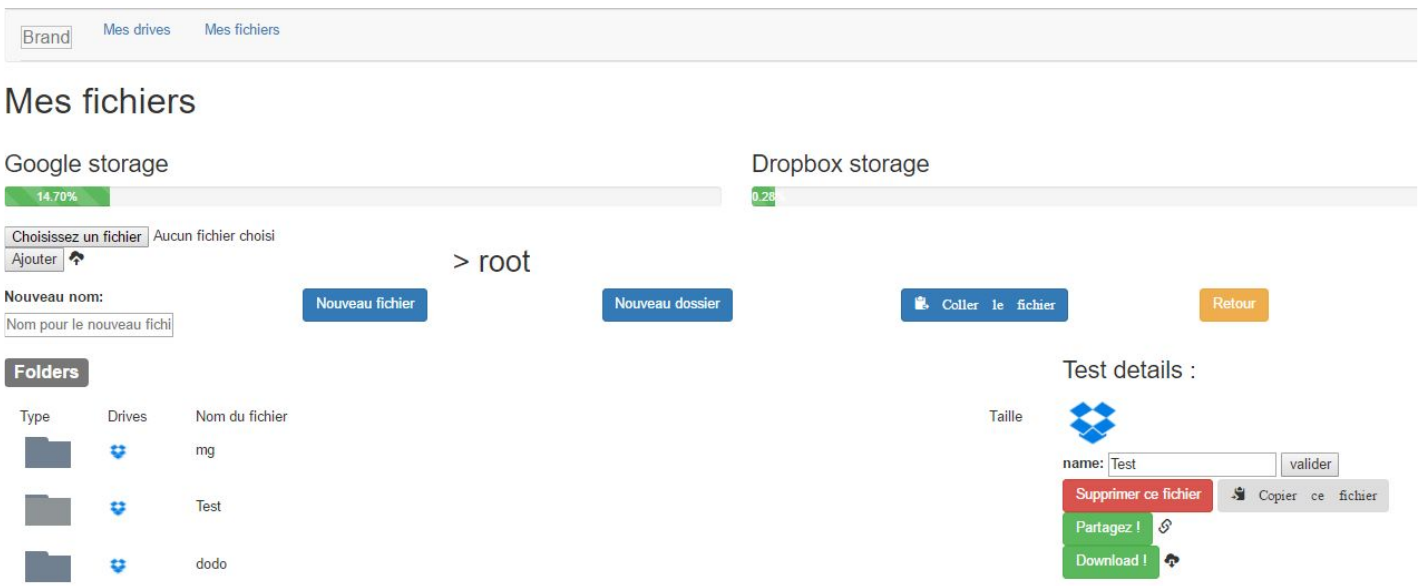


Figure.1

L'utilisateur peut alors s'authentifier sur les drives de son choix.
Dans certains cas où l'utilisateur serait déjà connecté à l'un des drives, l'authentification n'est pas requise. En effet, l'utilisateur étant déjà considéré comme connecté, une redirection automatique est effectuée.

2. L'accès aux fichiers

Une fois l'authentification effectuée, il suffit maintenant de cliquer sur « Mes fichiers » dans la barre de navigation située en haut de la page. Une nouvelle vue apparaît alors.



Les fichiers et les dossiers apparaissent en bas de la page.
Des icônes apparaissent à côté du nom du fichier. Elles représentent les drives auxquels appartient l'élément.
On peut retrouver les valeurs relatives à l'espace de stockage de chacun des drives grâces aux barres de remplissages situées en haut de l'écran. Le vert indique l'espace déjà utilisé.

Pour accéder aux détails des différents éléments, il suffit de cliquer sur l'élément en question. L'ensemble des informations relatives à l'éléments en question apparaîtront sur la droite de la page.

1) Créer un dossier ou fichier

Il est possible de créer un fichier ou un dossier grâce aux boutons "Nouveau fichier" et "Nouveau dossier". Il faudra tout d'abord avoir renseigné un nom dans le champ prévu à cet effet.

2) Copier / coller un fichier

Pour créer une copie d'une fichier, il faut tout d'abord sélectionné un élément. Cela fait, il faut cliquer sur le bouton "copier", situé dans les détails. Une fois le fichier copié, il suffit tout simplement de se déplacer le dossier de destination, puis de cliquer sur le bouton "coller". Une copie du fichier sera alors créée dans le dossier courant.

3) Partager un élément

Via le bouton "partager", situé dans les détails, un utilisateur peut récupérer le lien de partage du fichier en question.

4) Renommer un fichier / dossier

Il est possible de renommer un fichier ou un dossier du drive. Pour cela, il suffit de modifier le nom du fichier en question, puis de cliquer sur le bouton "Valider"

5) Upload un fichier

L'utilisateur peut à tout moment upload un fichier sur ses drives via le bouton "Ajouter".

6) Download un fichier

L’utilisateur peut télécharger un fichier présent sur le drive grâce au bouton prévu à cet effet : “Download”.Celui-ci est situé les détails de l’élément.

Il est aussi possible d’accéder à un menu raccourci, en effectuant un clic droit sur l’élément voulu. Une fenêtre apparaît, et proposer de copier ou supprimer l’élément.