IUT Nantes - Département Informatique

Rapport de stage : Restlet

APISpark

Etudiante : Marina Blin

Tutrice : Solen Quiniou

Maître de stage : Manuel Boillod

LaTeX

13 Avril 2015 — 19 Juin 2015

Remerciements à :

Manuel Boillod, mon tuteur de stage pour m'avoir permis de faire mon stage dans l'entreprise Restlet.

Constant Manteau pour m'avoir aider lors des différentes installations, notamment pour l'installation de l'environnement de développement.

Guillaume Blondeau pour m'avoir aider lors des différents problèmes que j'ai rencontrés avec APIspark.

Pascal Guilcher et **Nicolas Baptiste** pour avoir répondu à mes questions concernant le front end.

Thierry Boileau pour m'avoir expliqué le fonctionnement d'APISpark ainsi que pour m'avoir aidé sur le back end.

Jonathan Michaux pour avoir pris le temps de m'expliquer les spécifications attendues lors de mes missions.

Cyprien Quilici pour m'avoir aidé lorsque j'avais des problèmes avec la configuration d'Eclipse ainsi que pour avoir relu mon code et m'avoir aidé sur différents points du code d'APISpark.

Jean-Phillipe Le Goff pour m'avoir expliquer le système de tickets pour le Github.

L'équipe dans son ensemble pour son accueil et pour m'avoir intégré à l'équipe.

Je souhaite aussi remercier **Solen Quiniou** pour sa disponibilité et ses conseils tout au long de mon stage.

Tous les mots soulignés se trouvent dans le glossaire à la fin de ce rapport.

Résumé :

J'ai réalisé mon stage de fin de DUT dans l'entreprise Restlet. Cette entreprise a créé différents produits web. Dans ce rapport, je parlerai du seul produit sur lequel j'ai travaillé : APISpark. Le but final de ce produit est de permettre aux utilisateurs d'exporter leurs bases de données en API web. Ainsi, ils peuvent utiliser cette exportation en l'ajoutant et en y faisant appel dans leur site web.

Le but de mon stage était de m'occuper de la partie Google spreadsheet d'APISpark. Je devais aussi bien régler les problèmes qui s'y trouvaient qu'ajouter des nouvelles fonctionnalités. Ces différentes tâches étaient au préalable créées sur Github par l'intermédiaire de tickets.

Abstract :

To validate my DUT, I did my internship at Restlet. This company has created of different products. In this report, I will focus on APISpark, which is the product I worked on. The purpose of APISpark is to let the user export a database to a API web. This API can then be used in different applications, such as web sites or mobile applications.

During my time at Restlet, I worked on the Google spreadsheet wrapper, one of APISpark's key features. My contribution to this feature was to correct existing bugs and also to add new functionalities.

Table des matières

Introduction

Lors de ma deuxième année à l'IUT, j'ai réalisé un stage de 10 semaines dans une entreprise. J'ai souhaité faire ce stage dans le web. En effet, le développement web m'intéresse beaucoup car il regroupe plusieurs disciplines que j'apprécie tel que le développement front end et le développement back end ainsi que la proximité avec l'utilisateur. De plus, je souhaitais effectuer mon stage dans une petite entreprise pour la communication entre les différents salariés de l'entreprise.

Ce stage m'a permis de découvrir le monde de l'entreprise ainsi que le monde de l'informatique d'un point de vue professionnel avec, notamment, ses applications au quotidien.

Ma mission durant ce stage était d'améliorer la fonctionnalité Google spreadsheet, correspond au tableur de Google, de l'API APISpark. L'objectif de ce stage était l'amélioration des fonctionnalités dans APISpark. C'est-à-dire que mes missions portaient aussi bien sur la correction de problèmes rencontrés que sur la création de nouvelles fonctionnalités que ce soit du côté client que du côté serveur.

Dans la suite de ce rapport, je commencerai par présenter l'entreprise. Ainsi que de son cadre économique, humain et technique. Dans un deuxième temps, je décrierai la méthodologie mise en place par l'entreprise pour fonctionner. Puis je parlerai du projet sur lequel ma mission a porté et, enfin, je présenterai la mission que j'ai eu à réaliser durant mes 10 semaines de stage.

1 Présentation de Restlet

1.1 Présentation général

La société Restlet est une startup créée en 2008 par messieurs Jérôme Louvel, Thierry Boileau et Steve Sfartz. Cette entreprise est un éditeur de logiciels spécialisé dans les API web. Elle est aussi à l'origine d'APISpark qui a été développée avec Restlet Framework. APISpark vise les éditeurs d'applications mobiles et toutes les entreprises souhaitant partager et mettre à disposition ses données. Restlet framwork a été cré en 2005 et en 2013 APISpark.

Le siège de l'entreprise se situe dans les locaux d'ESSEC Ventures à La Défense. En 2013, un bureau a été ouvert dans la Silicon Valley et depuis un peu plus d'un an (en avril 2014), l'équipe recherche et développement s'est installée dans les locaux du Hub Créatic à Nantes. (Voir figure 1 et 2 ci-dessous)

Figure 1 – Le Hub Créatic

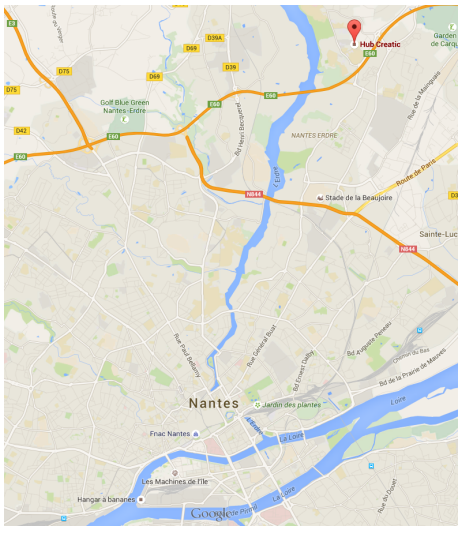

Figure 2 – Emplacement de Restlet

1.2 Cadre économique

L'entreprise Restlet vise en priorité le marché américain. C'est pour cela que les principaux clients sont des clients étrangers comme Smartbear (entreprise qui permet notamment de tester des API). Cependant, la société a aussi des clients français comme Canal+ par exemple.

1.3 Cadre humain

Restlet est décomposé en trois groupes. (Voir figure 3) Une partie de l'équipe est située à Paris, il s'agit de tout ce qui concerne l'administration. Le marketing, corespondant à la deuxième partie, se situe aux Etats-Unis. Et enfin, la troisième partie est l'équipe recherche et développement, qui se situe à Nantes.

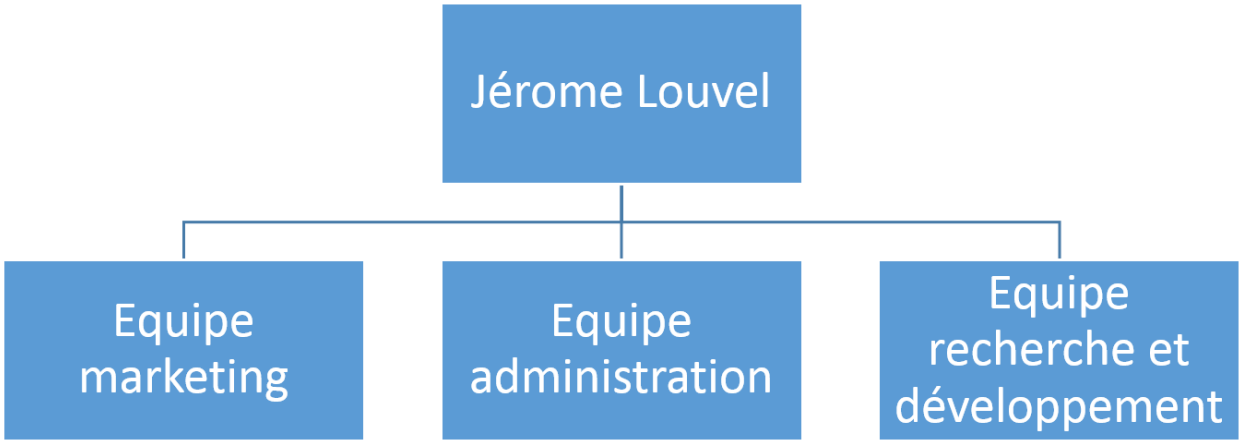

Figure 3 – Organigramme de l'entreprise

Dans l'équipe recherche et développement, il y a quatre équipes qui seront présentées dans les sections suivantes l'équipe développement, l'équipe système, l'équipe contôle qualité et enfin l'équipe documentation. La communication entre les différentes équipes est très forte. En effet, cela permet un meilleur résultat face à l'attente des utilisateurs.

1.3.1 L'équipe de développement

L'équipe de développement est composé d'une dizaine de personnes. Cette équipe est décomposée en deux parties : certaines personnes sont plus spécialisées dans le développement front end et d'autres dans le développement back end. Les personnes rattachées à cette équipe de développement sont ceux se trouvant à la figure 4 ci-dessous.

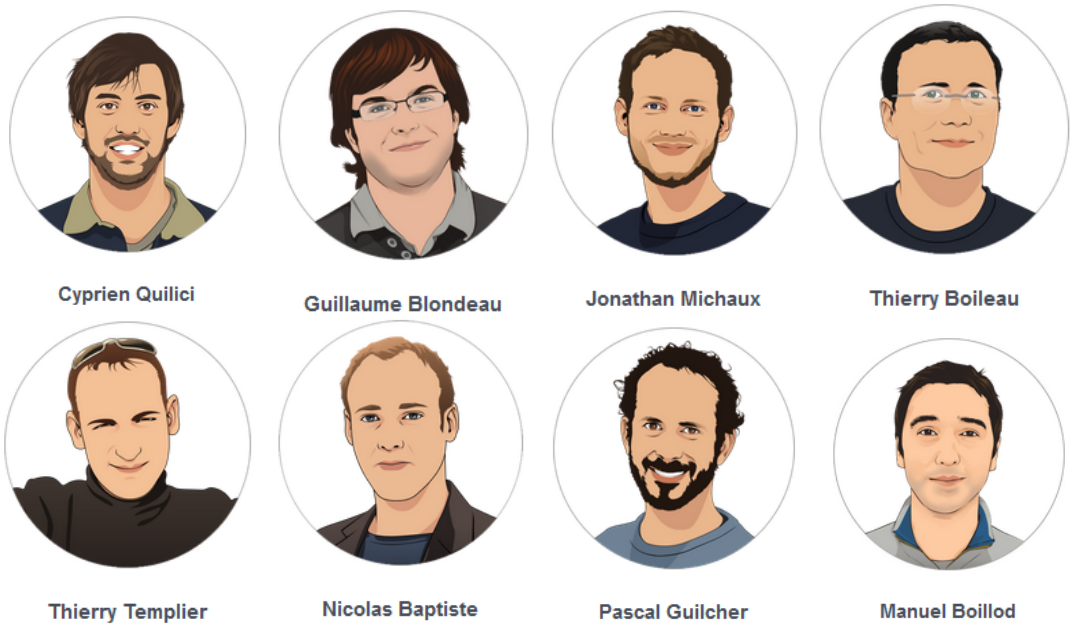

Figure 4 – Equipe de développement

1.3.2 L'équipe système

Cette équipe est composée d'une personne. (Voir figure 5) Elle s'occupe principalement de la mise en production des différents produits de la société.

Figure 5 – Equipe système

1.3.3 L'équipe contôle qualité

Cette équipe est composée de deux personnes. (Voir figure 6) Cette équipe testent les différentes fonctionnalités d'APISpark comme des utilisateurs. De cette façon, les tests sont faits sur le serveur de pré production. Si les tests passent, les nouvelles fonctionnalités sont alors mises en production. S'il y a des erreurs détectées, il y a un retour aux développeurs pour qu'une correction soit mise en place.

FIGURE 6 – Equipe contôle qualité

1.3.3.1 L'équipe documentation

Cette équipe permet de documenter APISpark sur ses différentes fonctionnalités. Il y a une personne dans cette équipe. (voir figure 7)

FIGURE 7 – Equipe documentation

1.4 Cadre technique

L'entreprise est basée au Hub Créatic. Les locaux possèdent quatre pièces. L'une d'entre elle sert de salle de réunion ainsi que de salle de pause. Les trois autres sont des salles de travail. (Voir figure 8) Dans l'entreprise, chaque employé a à sa disposition un ordinateur portable (Mac ou PC) ainsi qu'un ou deux écrans. L'ensemble de l'équipe recherche et développement utilise le système d'exploitation Linux. Certains ordinateurs sont équipés de Windows 7. Ces ordinateurs servent principalement à l'équipe contrôle qualité.

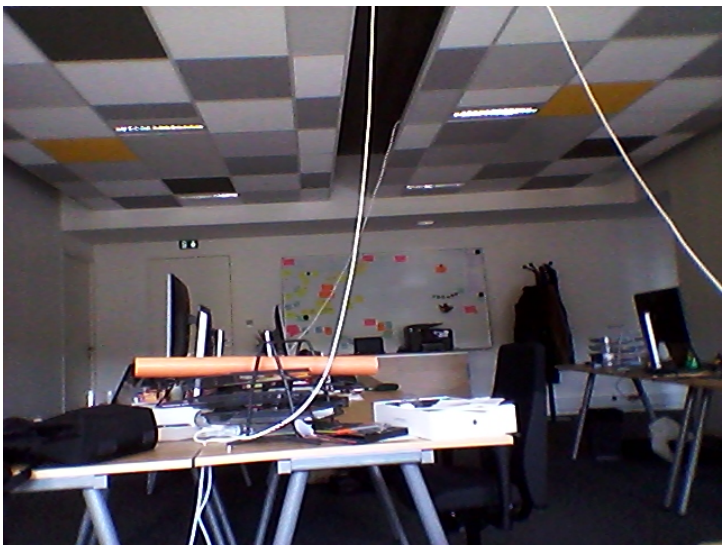

Figure 8 – Open space

2 Méthodologie de l'entreprise

2.1 Méthodologie scrum

Dans cette partie, je vais parler de la méthodologie de l'entreprise. En effet, il me semble important de parler de la méthodologie de travail, car elle est très importante dans la réalisation du projet auquel j'ai été affectée. La méthodologie utilisée par l'entreprise est l'une des méthodes agiles, appelée scrum. Les méthodes agiles sont basés sur quatre valeurs : l'équipe, l'application, la collaboration et l'acceptation du changement.Je vais donc présenter cette méthodologie en la déroulant dès son début, c'est-à-dire dès la mise en place d'un projet. Pour cela, la figure 9 résume l'explication qui va suivre.

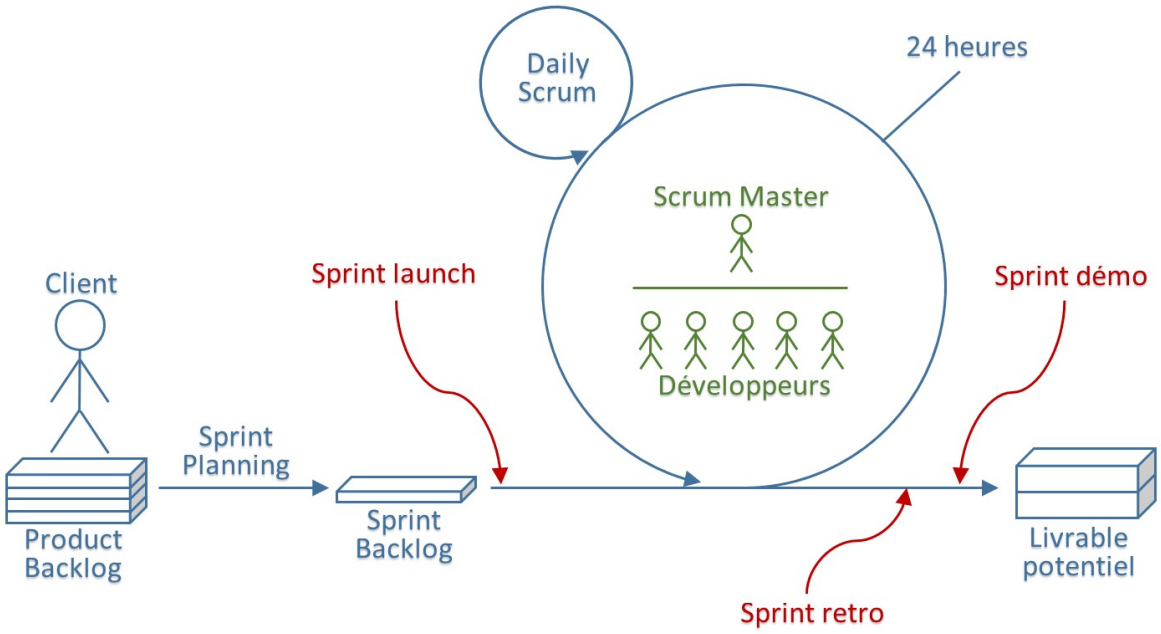

FIGURE 9 – Schéma de la méthode agile Scrum sur tout un projet

Lorsqu'un projet commence, le client définit ses attentes ou product backlog. Une fois les attentes du client fixées, un sprint planning est mis en place, c'est-à-dire que le projet est découpé en plusieurs parties. Une fois cela fixé, le sprint backlog est prêt à démarrer. Autrement dit, l'itération est prête à être lancé. Cependant, avant le lancement du sprint, une réunion appelée sprint launch est d'abord réalisée. Cette réunion permet d'informer l'équipe sur les fonctionnalités à mettre en place au cours du sprint. Lorsque le sprint launch est fini, le sprint peut alors commencer.

Généralement, un sprint dure entre deux et quatre semaines. A Restlet, un sprint a une durée allant de 10 jours à 14 jours. Il peut aussi arriver qu'il y ait des jours exclusivement réservés à la correction de problèmes. En général, ces journées se trouvent entre 2 sprints

Durant toute la durée du sprint, tous les matins, le scrum master gère une réunion d'environ 15 minutes qui s'appelle le daily scrum et qui permet à l'équipe de communiquer. Cette communication entre les différents membres de l'équipe permet à chacun de savoir où chaque personne est rendu dans sa tâche. Lors de cette réunion journalière, les tickets (ou tâches) affectés aux personnes sont déplacés sur un tableau, d'une colonne à une autre suivant leur état d'avancement. La première est "To do" ou à faire : ce sont les tâches qui n'ont pas été traitées. Vient ensuite la colonne "In progess" ou en cours : ce sont les tickets qui sont en cours de traitement. Une fois la tâche traitée, elle passe en "packaging". Autrement dit, la tâche est finie mais le développeur attend qu'on relise son code pour le valider. Une fois cette validation effectuée, le ticket passe à "done". Une fois que la tâche est validée, elle doit passer les tests de l'équipe contôle qualité qui va vérifier qu'il n'y a pas d'erreur au niveau de l'utilisation pour l'utilisateur. Si la tâche passe toutes les colonnes, elle peut alors partir en production.

Une fois le sprint terminé, une autre réunion est alors fixée : le sprint retro. Cette réunion permet à chaque membre de l'équipe de dire ce qui a été un moteur (ou engine), un frein (ou parachute), un pont (ou bridge) et un puits (ou abyss) lors du sprint qui vient de se passer. Cela permet de régler des problèmes pour les sprints qui suivront ou alors faire en sorte de maintenir les éléments moteurs du sprint. (Voir la figure 10)

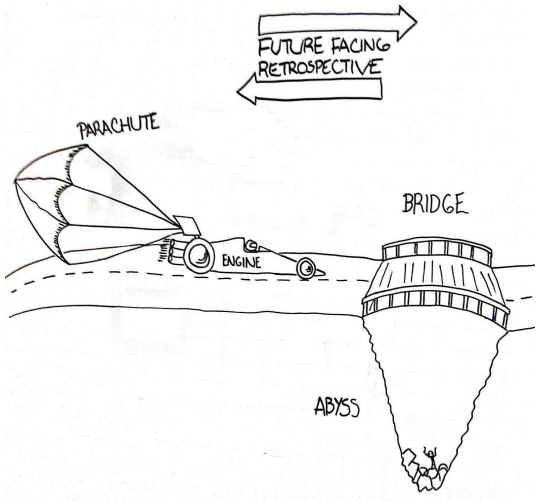

Figure 10 – Sprint retro

Après le sprint retro vient une autre réunion intitulé sprint démo. Cette réunion permet de faire une démonstration des fonctionnalités qui ont été ajoutées pendant le sprint.

Lors des sprints, la communication entre les différents salariés se fait principalement par skype. Mais lorsque plusieurs personnes travaillent sur la même tâche durant un sprint, il y a une réorganisation des bureaux afin qu'elles soient dans le même bureau, pour ainsi faciliter la communication.

2.2 Github

Pour le partage de fichiers et la mise en place du versionning, les salariés de Restlet utilise Github. Github permet aussi d'ouvrir des problèmes (ou issues) et de les affecter à un employé. Cela permet aussi de voir quand un ticket a été créé et le descriptif du problème rencontré. Une fois le problème traité, une demande d'ajout des modifications (ou pull request) est mise en place et une autre personne relit le code. Si le code convient, il est alors validé. Sinon, des commentaires sont placés aux endroits qui ne conviennent pas pour que le développeur qui a écrit le code puisse faire des modifications et refaire une pull request.

3 Présentation du projet

Le projet sur lequel j'ai travaillé se nomme APISpark. Je commencerai par faire une description de ce projet puis je présenterai son fonctionnement.

3.1 Description du projet

Le projet sur lequel j'ai travaillé au cours de ces 10 dernières semaines est un site permettant de créer des API web. Le principe est très simple. L'utilisateur a, par exemple, besoin d'avoir une base de données pour une application mobile. Pour cela, il se connecte à APISpark et arrive sur la page d'accueil de la figure 11.

Il crée alors un "entity store" du type "full stack". A partir de ce moment là, il pourra lui donner un nom, et ajouter une "entity" (qui correspond à une table en SQL) et des "properties" (qui sont l'équivalent des colonnes en SQL), dont il a besoin pour sa base de données. Il pourra ensuite ajouter des lignes qui seront les différentes valeurs des colonnes.

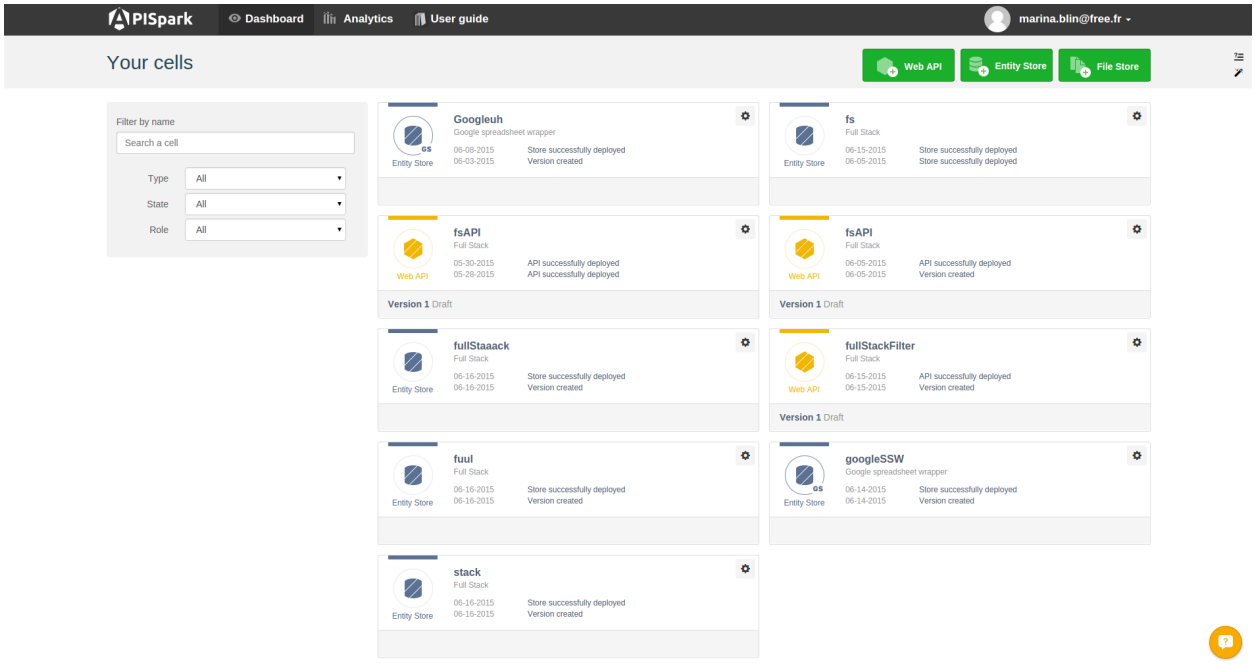

FIGURE 11 – Page d'accueil d'APISpark

Une fois les valeurs entrées, l'utilisateur doit déployer le store qu'il vient de créer. Cette action permettra la génération de code. Une fois le déploiement effectué, il faut l'exporter en API web. Il pourra alors donner un nom à cette API web. Lorsque cette API web est créée, l'utilisateur devra la déployer. Une fois réalisé fait, il pourra y faire appel pour son application mobile.

L'utilisateur peut aussi exporter un fichier json, par exemple, qui lui permettra de faire des appels sur sa base de données à partir de ce fichier, pour un site web par exemple.

De plus, si l'utilisateur possède la base de données voulue sur un tableau Google (ou Google spreadsheet), il peut directement y faire appel pour récupérer les données. Il en va de même s'il utilise Parse ou Firebase ou même s'il a déjà un fichier SQL existant.

La mascotte d'APISpark s'appelle Sparky : il s'agit d'une abeille. Tout vient du latin. En effet, abeille en latin est apis. Le but d'APISpark, est de pouvoir relier toutes les API entre elles à "l'infini". C'est pour cela que lorsqu'une API web est créée, elle a la forme d'un hexagone. Cette forme rappelle aussi une alvéole. (Voir figure 13) Le fait d'en rattacher plusieurs donne l'idée d'une ruche. (Voir figure 13)

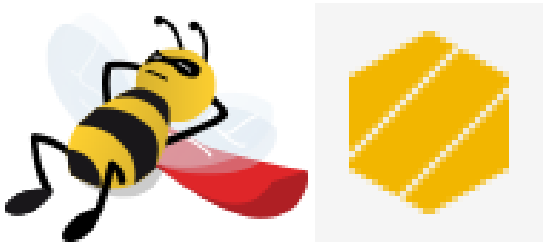

Figure 12 – Sparky - API web

Figure 13 – Ruche

3.2 Description du fonctionnement

Dans cette partie, je vais décrire le fonctionnement du projet APISpark. Je présenterai tout d'abord le fonctionnement du REST et ensuite je décrirai du fonctionnement de l'appel d'une application d'APISpark.

3.2.1 Fonctionnement du REST

Pour expliquer le fonctionnement du REST, je vais m'appuyer sur le schéma de la figure 14. Il est possible de voir deux parties : une entourée de rouge, qui correspond au projet APISpark, et une partie non entourée, qui correspond aux support physiques et aux stores et API web que nous avons créés.

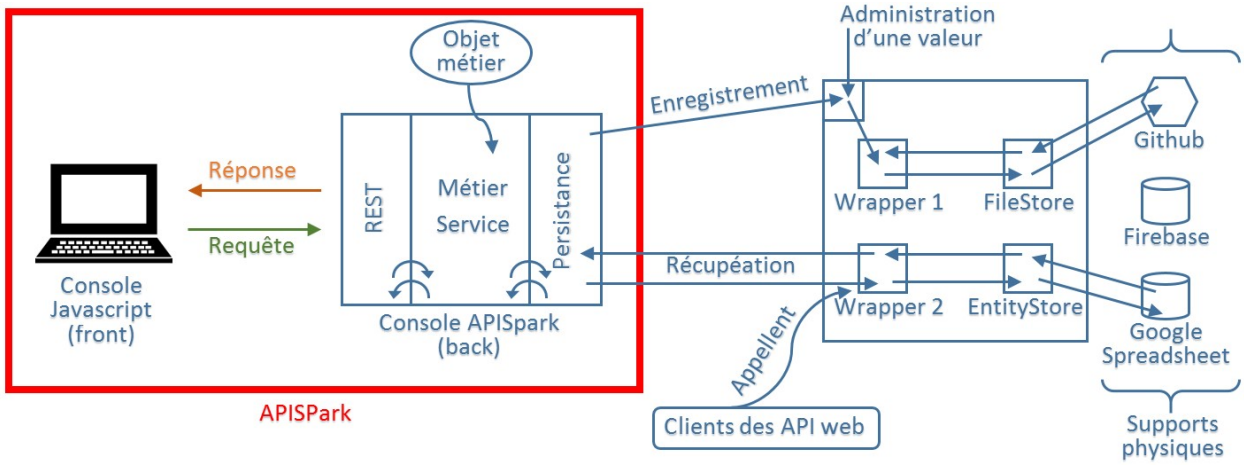

FIGURE 14 – Fonctionnement du REST

Le REST permet de communiquer des informations de la console Javascript d'APISpark à l'APISpark worker (conteneur de store et de wrapper). Pour passer les informations d'un côté à l'autre, nous utilisons des objets métiers. En d'autres mots, lorsque nous récupérons des valeurs dans le wrapper2 du schéma, nous les récupérons grâce à la persistance de la console APISpark. Une fois que nous avons récupéré ces valeurs dans la persistance, nous ne pouvons pas les renvoyer directement au front end. C'est ici que nous utilisons alors les objets métiers. Ces objets métiers nous permettent de passer à la partie REST qui communique avec la console Javascript.

REST n'est pas un protocole. Il s'agit d'un système architectural qui s'applique à la réalisation d'architectures orientées service destinées à la communication entre machines. L'un des avantage de REST est, dans un contexte web comme cela a été pour mon stage, l'utilisation du protocole HTTP en utilisant son enveloppe et ses en-têtes. De plus, avec le REST, l'URI est utilisé pour représenter une ressource, ce qui permet d'avoir un système universel d'identification des éléments de l'application. Cependant, il y a aussi des inconvénients comme, par exemple, la nécessité pour le client de conserver toutes ses données localement. De ce fait, la consommation de la bande passante réseau est plus importante.

3.2.2 Fonctionnement d'un appel d'une application à APISpark

Dans cette partie, je vais expliquer le fonctionnement d'un appel à une API web par une application mobile. Pour cela, je vais considérer qu'un store a été créé au préalable et que nous avons pris les données qui étaient dans un Google spreadsheet. Nous avons également déployé ce store et exporté une API web.

La suite des explications est en relation avec la figure 15.

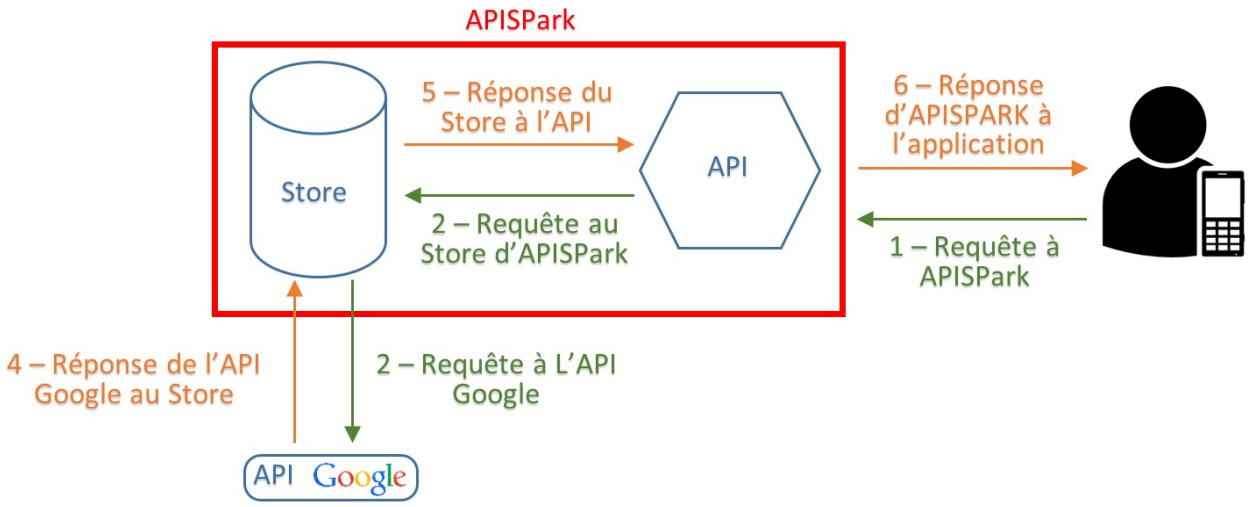

Figure 15 – Fonctionnement d'un appel d'une application à APISpark

Supposons que notre utilisateur a mis en place une application mobile lui permettant de gérer son carnet d'adresses. Les données de ce carnet d'adresse sont stockées dans un Google spreadsheet que nous avons importé pour le store que nous avons créé. L'utilisateur souhaite alors trier ses contacts. Pour cela, il fait une requête à l'API d'APISpark. L'API fera elle-même un appel au store auquel elle est reliée. Une fois arrivée au store, une requête sera envoyée à l'API Google. Une réponse sera alors renvoyée au store, qui lui-même donnera la réponse à l'API d'APISpark qui répondra enfin à la requête de l'application. (Voir figure 15)

4 Présentation de la mission

Dans cette partie, je vais présenter la mission. Pour cela, j'identifierai d'abord les outils utilisés puis je présenterai le projet.

4.1 Les outils

Pour la réalisation du projet, j'ai eu besoin de différents outils. Ces outils utilisés se séparent en deux catégories : ceux qui existaient déjà avant le projet et ceux développés par l'entreprise.

4.1.1 Les outils déjà existants

En ce qui concerne le front end, autrement dit la partie client, j'ai dû utiliser Webstorm. Il s'agit d'un éditeur de code. qui permet de formater le code de façon à ce que toutes les personnes travaillant sur le code côté front end ait le même formatage. La raison principale pour que tout le monde ait le même formatage de code est que lorsqu'une modification est faite, les différences sont directement visibles. Or, si chacun avait son propre formatage, il y aurait très vite des conflits et la relecture du code serait alors plus difficile car les changements moins visibles.

Pour ce qui est de la partie back end, ou serveur, c'est l'IDE Eclipse qui est utilisé. Là aussi, il y a un formatage au niveau du code pour les mêmes raisons que pour le front end.

Pour ma mission, j'ai dû utiliser l'API spreadsheet de Google.

4.1.2 Les outils développés par l'entreprise

APISpark est basé sur Restlet Framework, un produit développé par l'entreprise. Ce framework permet de créer facilement des API web en Java ainsi que d'utiliser le REST. Ce framework a été créé en 2005 par messieurs Jérome Louvel, Thierry Templier et Thierry Boileau. Ces personnes ont écrit un livre sur ce framework intitulé "Restlet in action" édité par l'édition Manning.

4.2 La mission

Le but de ma mission était la correction et l'amélioration de la partie Google spreadsheet d'APISpark. Cette fonctionnalité permet à l'utilisateur de récupérer ses données se trouvant dans son Google spreadsheet, et d'en faire une API.

4.2.1 La partie front

J'ai d'abord commencé par me familiariser avec l'API en ajoutant des fonctionnalités du côté front end. Comme dit précédement, le côté front end est réalisé avec angular JS.

4.2.1.1 Compatibilité entre navigateur

Sur le navigateur firefox, les listes déroulantes étaient présentes de la même façon que sur la figure 16. Alors que sur chrome, ces mêmes listes déroulantes étaient représenter comme sur la figure 17.

Figure 16 – Rendu d'une liste déroulante sous firefox

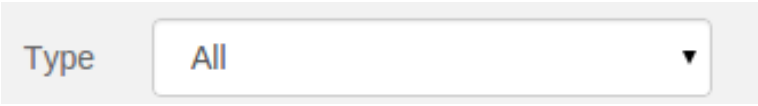

Figure 17 – Rendu d'une liste déroulante sous chrome

Cependant, il y a certaines listes déroulantes qui ont le même rendu sous firefox que sous chrome. (Voir figure 18)

Figure 18 – Rendu d'une liste déroulante identique sous firefox et sous chrome

La différence se trouve au niveau du code. En effet, pour les figures 16 et 17, une balise native, autrement dit la bailse HTML select, qui est utilisée.

Pour comprendre la différence de rendue suis suivant les navigateurs, j'ai cherché sur leur site officiel respectif. Sur le site de mozilla, il est dit que le rendu de la liste déroulante ne peut pas être changée pour obtenir le même résultat que sous chrome.

Cependant, il est possible de constater d'après la figure 18 que certaines listes déroulantes peuvent avoir le même rendu suivant le navigateur. Cela vient du fait que ce n'ai pas une balise html qui est utilisée mais la balise ui-select d'angularJS.

Or, il y a environ une centaine de balise native select. Changer autant de balise native en balise angularJS est très couteux. Les listes n'ont pas été changées et sont restées avec le rendu que nous pouvons voir sur les figures 16 et 17. Cendant, pour la suite du code, nous allons faire attention lorsqu'une liste déroulante sera créée.

4.2.1.2 Système de recherche

Ma deuxième mission a été de permettre aux utilisateurs de pouvoir chercher un Google spreadsheet dans la liste de Google spreadsheet que nous récupérons. (Voir figure 19)

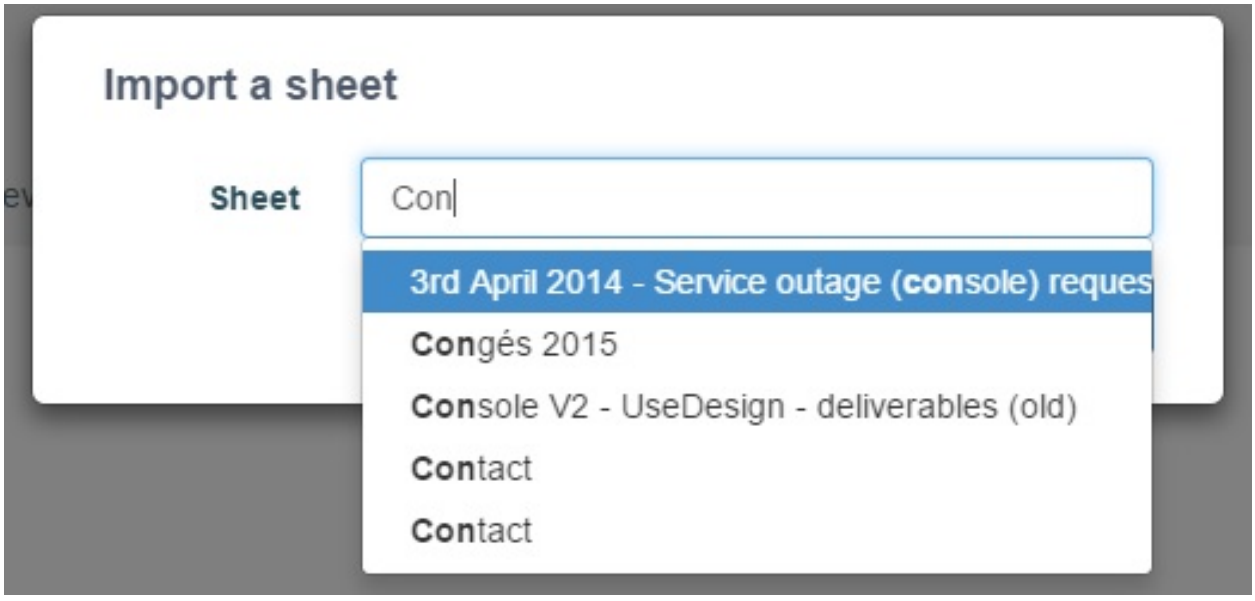

Figure 19 – Rendu de la recherche dans la liste déroulante

Pour cela, il a fallu faire une analyse du code afin de voir s'il était moins couteux d'utiliser une balise native ou un balise angular JS. Après discussion avec les salariés, nous nous sommes rendu compte qu'il était mieux d'utiliser une balise angularJS. La balise en question est la balise ui-select. Le code fourni à la figure 21 permet l'affichage de la figure 20. Il est possible de voir la balise ui-select utilisée. Ensuite, il y a la balise ui-select-match qui est un equivalent de la balise input. Dans la balise div, il est possible de voir que l'attribut HTML "title" permet d'afficher le titre de la feuille Google selectionnée par l'utilisateur. La recherche quant à elle, est réalisée dans la balise ui-select-choices. L'attribut "repeat" a la même fonction que l'itération en java. Autrement dit, ici, nous avons une liste de "spreadsheets" et à chaque passage, une variable "spreadsheet" est créée. Ensuite, le "filter" permet de garder uniquement les éléments cherchés par l'utilisateur.

```
<ui-select ng-model="selection.spreadsheet"
           ng-disabled="!spreadsheets.length"
           focus-on="focus-import-sheet"
           style="min-width: 300px;"
           reset-search-input="true">

  <ui-select-match placeholder="Enter a spreadsheet title"
                   allow-clear="false"
                   ui-lock-choice="true">
    <div title="{{ $select.selected.title }}"
         class="ellipse"
         style="width: 290px;">
      {{ $select.selected.title }}
    </div>
  </ui-select-match>

  <ui-select-choices repeat="spreadsheet in spreadsheets | filter: $select.search">
    <div ng-bind-html="spreadsheet.title | highlight: $select.search"
         title="{{ spreadsheet.title }}"></div>
  </ui-select-choices>
</ui-select>
```

Figure 20 – Utilisation de la balise ui-select pour la recherche dans une liste déroulante

Enfin, afin de rendre le résultat de la recherche plus visible aux yeux de l'utilisateur, grâce à l'attribut ng-bind-html la balise div permet de mettre en gras les caractères ou chaîne de caractères que l'utilisateur est en train de rechercher.

4.2.2 La partie back

Une fois cette mission réalisée, je suis passée sur le côté backend. Pour cela, j'ai utilisé les méthodes de l'API Google spreadsheet.

4.2.2.1 Changement d'importation

L'une des missions que j'ai dû réaliser concerne la récupération d'un Google spreadsheet suivant sa clé unique et non plus par rapport à son nom. La raison principale à cela est que si deux Google spreadsheet ont le même nom, quelque que soit le Google spreadsheet selectionné, le Google spreadsheet qui était récupéré, était le premier trouvé. Et donc, ce n'était pas forcement celui voulu par l'utilisateur.

J'ai donc changé le fichier du coté back end afin de permetre la génération du code. Ainsi, la génération ne se faisait plus par rapport au nom du Google spreadsheet mais par rapport à sa clé unique. Justqu'à maintenant, seul le nom était envoyé au back end. J'ai donc aouté la récupération de la clé unique du côté front end. Ainsi, le nom et la clé unique étaient envoyés au back end. (Voir figure 21)

```
var sheetToImport = {
  spreadsheetKey: $scope.selection.spreadsheet.key,
  spreadsheetTitle: $scope.selection.spreadsheet.title
};
```

Figure 21 – Permettre la récupération de la clé unique côté front

Cependant, dans la base de données permettant de stocker les information des Google spreadsheet, rien ne me permettait de garder la clé unique. J'ai donc ajouter une colonne afin de pouvoir les stocker. (Voir figure 22)

```
CREATE TABLE "EntityStoreGoogleSheetDependency" (
    "id" varchar,
    "sheetName" varchar,
    "sheetId" varchar,
    "sheets" list<varchar>,
    "implementation" varchar,
    PRIMARY KEY ("id")
);
```

Figure 22 – Ajout de la colonne sheetId dans la table EntityStoreGoogleDependency

Cet ajout dans la base de données se fait grâce à un script de migration. (Voir figure 23)

```
alter table apispark."EntityStoreGoogleSheetDependency" add "sheetId" varchar;
```

Figure 23 – Script de migration

De plus, avant il existait une liste qui contenait tous les noms des Google spreadsheet. En intégrant le système de clé unique, il a fallu changer de structure. J'ai donc changé les listes par des Map<String,String>. Ainsi, j'ai pu relier le nom de la Google spreadsheet avec sa clé unique.

La suite de cette mission a été d'optimiser cette recherche. En effet, maintenant, nous ne récupérons plus un nom de Google spreadsheet mais nous le récupérons par sa clé unique. Comme cette clé est unique, il nous est donc possible de ne plus passer par une liste mais de directement intégrer la clé unique dans l'adresse URL. Cela permet donc de récupérer toutes les Google spreadsheets ayant cette clé unique. Comme chaque Google spreadsheet à une clé unique, nous ne récupérons qu'une seule Google spreadsheet. (Voir figure 24)

```
if (sheetId != null) {
    URL spreadSheetFeedUrl = new URL("https://spreadsheets.google.com/feeds/spreadsheets/private/full/" + sheetId);
    // Look by sheet identifier
    SpreadsheetEntry spreadsheetEntry = service.getEntry(spreadSheetFeedUrl, SpreadsheetEntry.class);

    if (LOGGER.isDebugEnabled()) {
        String format = "Spreadsheets retrieved [cell,version]=[%s,%s]";
        logEndTime(beginTime, String.format(format, cellId, versionMajorNumber));
    }
    return spreadsheetEntry;
```

Figure 24 – Amélioration de la recherche

4.2.2.2 Les filtres

Une autre mission portait sur le système de filtre [1] d'un Google spreadsheet wrapper. C'est-à dire-que lorsque l'utilisateur fait appel à la API web qu'il a créé, il peut désormais choisir de filtrer ses données. Par exemple, il peut choisir toutes ses données comportant le même nom de famille. Cela lui retournera donc la liste correspondante.

1. Voir les figures 1, 2, 3, 4 et 5 de l'annexe

Pour cela, j'ai utilisé les paramètres de query qui sont à notre disposition grâce à l'API Google. (Voir figure 25)

```
String listCondition = "";
com.google.gdata.client.Query feedQuery = new com.google.gdata.client.Query(listFeedUrl);

Iterator<Condition> iteratorCondition = query.getConditions().iterator();
Condition condition;
while (iteratorCondition.hasNext()) {
    condition = (Condition) iteratorCondition.next();
    String conditionFieldName = condition.getFieldName().toLowerCase();
    String conditionValue = (String) condition.getValue();
    listCondition += conditionFieldName + "=" + conditionValue;
    if (iteratorCondition.hasNext()) {
        listCondition += " and ";
    }
}

if (!listCondition.equals("")) {
    feedQuery.getCustomParameters().add(new CustomParameter("sq", listCondition));
}
```

FIGURE 25 – Création des filtres

De plus, pour réaliser cette mission, j'ai utilisé la classe CustomParameter qui prend en paramtre deux valeurs lors de sa création. La première, "sq", permet de dire que nous souhaitons filtrer une valeur. Le second paramètre permet de dire sur quelle valeur ce filtre sera affecté. Or, nous donnons la possibilité à l'utilisateur de rentrer plusieurs valeurs de filtre. Il a donc fallut créer un objet string pour pouvoir stocker toutes ces valeurs car il est impossible d'appliquer plusieurs fois le paramètre "sq" dans une requête.

Une fois toutes les conditions réunies, nous regardons si l'objet possédant nos conditions est vide ou non. Si l'objet ne possède aucune condition, alors nous effectuons aucune condition au feedQuery sinon, nous lui passons les paramètres dont il faudra tenir compte si une requète est effectuée sur les filtres. (Le résultat est visible en annexe)

4.2.2.3 Système de tri

Une fois que le système de filtre a été mis en place, j'ai pu réaliser un système de tri. (Voir figure 26) La méthode est exactement la même que pour le système de tri [2]. Seul le paramètre de query change.

2. Voir les figures 6 et 7 de l'annexe

En effet, cette fois, nous avons besoin de récupérer le nom de la colonne selon laquelle le tri sera effectué ainsi que le sens du tri. C'est-à-dire si nous voulons que les éléments soient triés de façon croissante ou de façon décroissante.

```
Iterator<OrderBy> iteratorOrder = query.getOrdersBy().iterator();
OrderBy orderBy;
String listOrder = "";
while (iteratorOrder.hasNext()) {
    orderBy = (OrderBy) iteratorOrder.next();
    if (!feedQuery.getCustomParameters().contains(orderBy)) {
        if (orderBy.getType() == OrderByType.DESC) {
            feedQuery.getCustomParameters().add(
                    new CustomParameter("reverse", "true"));
        } else {
            feedQuery.getCustomParameters().add(
                    new CustomParameter("reverse", "false"));
        }
    }

    listOrder += orderBy.getFieldName().toLowerCase();

    if (iteratorOrder.hasNext()) {
        listOrder += "&";
    }
}

if (!listOrder.equals("")) {
    feedQuery.getCustomParameters().add(new CustomParameter("orderby", "column:" + listOrder));
}
```

Figure 26 – Création du tri

Cependant, pour le moment, nous ne pouvons trier que sur un seul élément. En effet, Google ne prend pas en compte les autres paramètres de tri une fois qu'il y en a un.

Le code est légèrement différent. En effet, ce n'est plus le paramètre "sq" que j'ai utilisé mais le paramètre "reverse" pour savoir si je dois changer l'ordre de la colonne ou non. Ensuite, pour savoir sur quelle colonne le tri doit porter, j'ai utilisé le paramètre "orderby". (Voir figure 20)

Cependant, jusqu'ici, pour les cas où l'utilisateur effectuait une mauvaise requête, il n'y avait aucun message d'erreur qui lui était envoyé. Le tri n'était pas pris en compte. J'ai donc changé le template qui gère ce problème. (Voir figure 27)

J'ai alors vérifié que l'utilisateur ne passait pas trop de paramètre [3]. Pour cela, j'ai pris le paramètre et j'ai "coupé" ce paramètre suivant les espaces.

3. Voir figure 8 de l'annexe

Je me retrouve donc avec le paramètre qui me permet de savoir sur quelle colonne porte le tri et un second paramètre qui permet de savoir si le tri se fait d'une manière croissante ou décroissante.

```
if (sortParams != null) {

    String[] sortParamList = sortParams.split(",");

    for (String sortParam : sortParamList) {

        String[] splitSortParam = sortParam.trim().split("\\s+");
        String sortParamName = splitSortParam[0];

        if (!fields.contains(sortParamName)) {
            throw new org.restlet.resource.ResourceException(org.restlet.data.Status.CLIENT_ERROR_BAD_REQUEST,
            "Invalid $sort value: '" + sortParamName + "' is not a property name");
        }

        if (splitSortParam.length == 1) {
            OrderBy ob = new OrderBy();
            ob.setFieldName(sortParamName);
            ob.setType(OrderByType.ASC);
            orders.add(ob);
        } else if (splitSortParam.length == 2) {
            OrderBy ob = new OrderBy();
            ob.setFieldName(sortParamName);
            String type = splitSortParam[1];
            if ("desc".equalsIgnoreCase(splitSortParam[1])) {
                ob.setType(OrderByType.DESC);
            } else if ("asc".equalsIgnoreCase(type)) {
                ob.setType(OrderByType.ASC);
            } else {
                throw new org.restlet.resource.ResourceException(org.restlet.data.Status.CLIENT_ERROR_BAD_REQUEST,
                "Invalid sort " + sortParam + ", " + type + " is not supporterd");
            }
            orders.add(ob);
        } else {
            throw new org.restlet.resource.ResourceException(org.restlet.data.Status.CLIENT_ERROR_BAD_REQUEST,
            "Too many arguments have been passed to sort");
        }
    }
```

Figure 27 – Ajout de vérifications du tri dans un template java

De cette manière, je peux regarder si la requête possède plus de deux paramètres, une erreur est envoyé à l'utilisateur. Il en va de même s'il utilise un second paramètre invalide [4] ou si le nom de la colonne n'existe pas [5].

Il est aussi possible d'utiliser des filtres et le tri dans la même requête [6].

4.2.2.4 Versionnage

Cette mission traite de la pagination. En d'autres mots, cela permet à l'utilisateur d'avoir un certain nombre de données par page.

4. Voir figure 9 de l'annexe
5. Voir figure 10 de l'annexe
6. Voir figure 11 de l'annexe

Cette mission m'as permis de voir le système de version des wrappers qui sont créés. En effet, lorsqu'un Google spreadsheet wrapper est créé, une version lui est attribué. Avant d'ajouter la pagination, ce type de wrapper était en version 1. Lorsque la fonctionnalité de la pagination a été ajouté, les Google spreadsheet wrapper sont passés à la version 2.

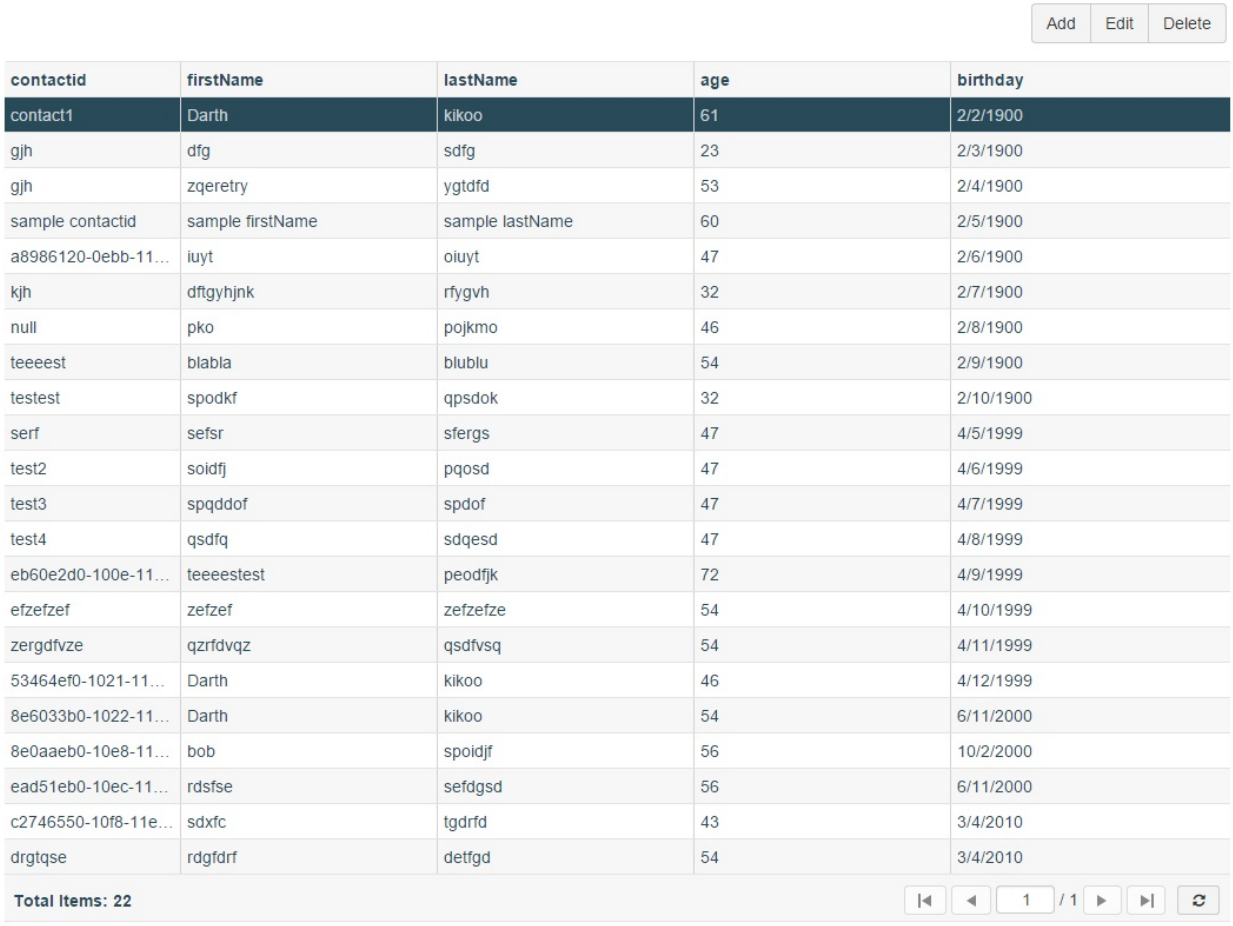

Figure 28 – Pagination

Cela a impliqué les causes suivantes. Les wrapper du type Google spreadsheet créés avant cette mise en place ne devait pas avoir l'affichage de la pagination temps qu'ils n'étaient pas déployé. Lorsqu'un de ces anciens wrapper était déployé, il devait passer à la version 2. Cela impliquait alors l'apparition de la pagination lors de l'affichage des valeurs qu'il contenait.

En ce qui concerne la pagination en elle-même, il a fallu affecter des valeurs aux paramètres de query concernant le nombre de valeurs par page, le nombre de pages ainsi que le nombre total de valeurs.

Conclusion

Pendant ces 10 semaines de stage, j'ai appris à travailler en entreprise. Cela implique le travail d'équipe, la communication et aussi une nouvelle méthodologie de travail.

De plus, j'ai pu apprendre à utiliser des utils ou des framework que je n'avais encore jamais utilisé comme Webstorm et AngularJS. J'ai aussi pu en apprendre plus sur l'IDE Eclipse comme le mode de debug et les breakpoints. Le fait d'utiliser Java pour le côté serveur était aussi très enrichissant car, jusqu'à maintenant, nous n'avions utilisé que du PHP ou du Python pour le back end. J'ai donc du faire des recherches pour bien comprendre comment circulent les informations, surtout avec le fonctionnement avec l'architecture REST du projet.

En ce qui concerne la mission qui m'a été attribuée, je l'ai trouvée très intéressante car elle était ciblée sur une seule fonctionnalitée. J'ai donc pu bien comprendre son fonctionnement. De plus, même si elle était sur une seule fonctionnalitée, les tâches m'ont permis de voir un large panel de cas, que ce soit du côté client avec le problème de compatibilité des différents navigateurs ou bien la mise en place d'une fonction de recherche, ou du côté serveur avec la génération de code java, la gestion d'une nouvelle version ou encore l'amélioration de la recherche.

Glossaire

API Application Programming Interface ou interface de programmation en français est un ensemble normalisé de classes, de méthodes ou de fonctions qui servent de façade par laquelle un logiciel offre des services à d'autres logiciels.. 1–3, 11, 14–16, 18, 21

API web API utilisé pour le web. 2, 10–13, 15, 20

APISpark Il s'agit du produit sur lequel j'ai travaillé durant mes 10 semaines de stage.. 1, 2, 5, 10–12, 14–16

back end C'est le côté serveur d'un site web.. 1, 4, 15, 19, 25

daily scrum Petite réunion réalisée tous les matin pour savoir ce qui a été fait la veille. 8

front end C'est le côté client d'un site web.. 1, 4, 12, 15, 16, 19

Google spreadsheet Il s'agit du tableur Google. 1, 2, 11, 13, 14, 16–20, 24

itération Chaque sprint correspond à une itération. Lors qu'un sprint est finit, l'itération est incrémentée.. 7

product backlog Produit finit voulu par le client.. 7

pull request Demande d'un développeur pour ajouter son code avec l'ensemble du code. 9

REST C'est un système architectural. 12, 13, 15, 25

Restlet Framework Il s'agit d'un produit de Restlet sur lequel APISpark est basé. 2, 15

scrum Il s'agit d'une des méthode agile.. 7

scrum master Personne en communication avec le product manager. 8

sprint Il s'agit du découpage d'un projet en une timeboxing ou boîte de temps en français. Cette boîte de temps est une période volontairement limitée pour une tâche donnée. 7–9

sprint backlog Fonctionnalités qui seront mises en place à la fin du sprint.. 7

sprint démo Démonstration des fonctionnalités mises en place lors de la dernière itération. 9

sprint launch Réunion permettant de lancer une itération. 7

sprint planning Découpage du product backlog par tâche.. 7

sprint retro Réunion permettant au employé de faire le point sur l'itération qui vient de passer. 8, 9

store Litéralement, un store désigne un magasin. Ici, nous pouvons concidérer qu'il s'agit d'un magasin de données. 11–14

URI Uniform Resource Identifier ou identifiant uniforme de ressource, est une courte chaîne de caractères identifiant une ressource sur un réseau. 13

versionning Permet de voir les différences entre le code déjà présent et celui qui est mis en place. Cela permet aussi de revenir en arrière si jamais il y a eu un problème. 9

Webographie

- Restlet in action - J. Louvel, T. Templier, T. Boileau : recherche sur Restlet framework
- Les cours de Programmation objet pour la partir serveur
- Les cours de web pour la partie client
- http ://enicolashernandez.blogspot.fr/2011/03/redaction-dun-rapport-de-stage.html pour la rédaction du rapport
- Le cours de culture et communication pour la rédaction du rapport
- Les cours de Gestion de Projet Informatique de l'IUT pour la rédaction du cahier des charges
- Le cours des mines portant sur les méthodes agiles pour la compréhension de la méthodologie de l'entreprise

Rapport de Stage

Université de Nantes

Institut Universitaire de Technologie

Département Informatique

Nantes, le 22 juin 2015