TP1 : Fork them all !

Episode 1 - Introduction

Vous allez voir à travers ce projet tous les aspects évoqués en cours vis à vis des techniques de tests. Pour cela nous allons réaliser un projet logiciel de petite taille, en suivant la roadmap suivante :

- Spécification / Création du projet
- Ecriture des tests unitaires
- Ecriture des mocks, et validation des tests
- Mise en place des outils d'intégration continue
- Développement
- Validation des implémentations par les tests

Le projet consiste à mettre en place un outil d'analyse statistique pour le jeu **Pokémon GO**. Si vous êtes de l'équipe **Bravoure**, félicitation vous venez de gagner 1 point[1]. Sinon ... Tant pis pour vous il fallait mieux choisir votre équipe.

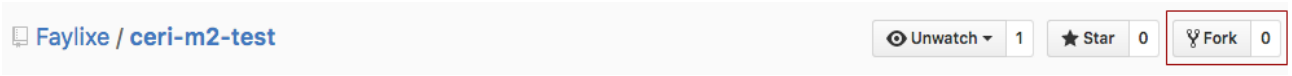
Durant cette série de TP, le gestionnaire de version **Git** sera utilisé à foison, à travers la plateforme **GitHub**. Si vous n'êtes pas à l'aise avec cet outil[2], voici un petit guide à garder sous la main : http://rogerdudler.github.io/git-guide/ .

[1] Sous réserve de présentation de votre profil lors de la première séance de TP, niveau 10 minimum requis.

[2] Si vous n'êtes vraiment pas à l'aise avec cet outil, je vous conseille quand même de vivement vous y mettre, on est en 2017.

Episode 2 - Récupération des sources

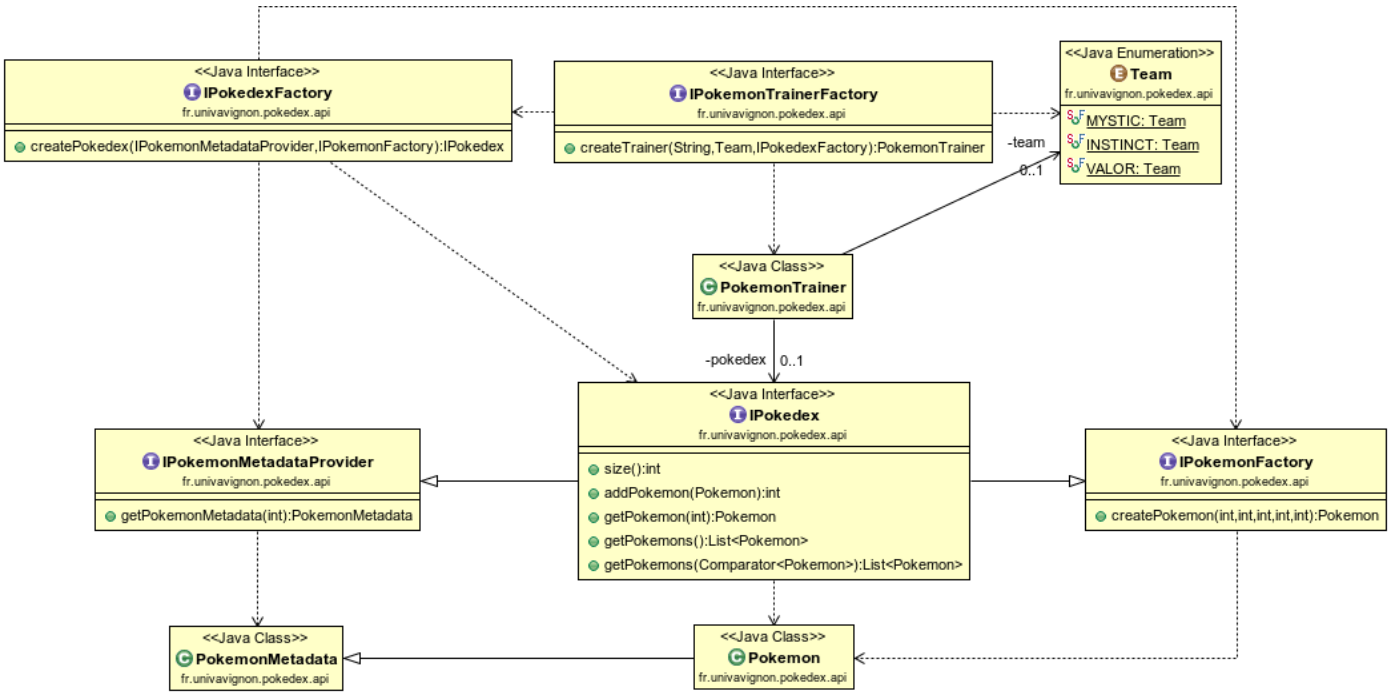
La partie spécification ainsi que l'écriture de l'architecture applicative n'est pas à faire dans le cadre de ce TP. Elle est donnée et doit être récupéré sur **GitHub**. Une fois connecté sur le site https://github.com, rendez-vous à l'adresse https://github.com/Faylixe/ceri-m1-test et cliquez sur le bouton **Fork** en haut à droite :



Le **fork** du projet fait une copie intégrale du **repository** sur votre compte. C'est ce projet forké que vous utiliserez comme base de travail. Vous pouvez maintenant récupérer le projet sur votre machine, pour cela, ouvrez un terminal et tapez la commande suivante[3] :

```
git clone https://github.com/your_username/ceri-m2-test.git
```

Le projet ne contient qu'un package Java `fr.univavignon.pokedex.api`, sous le répertoire **src/main**. Ce package expose l'API de notre application :



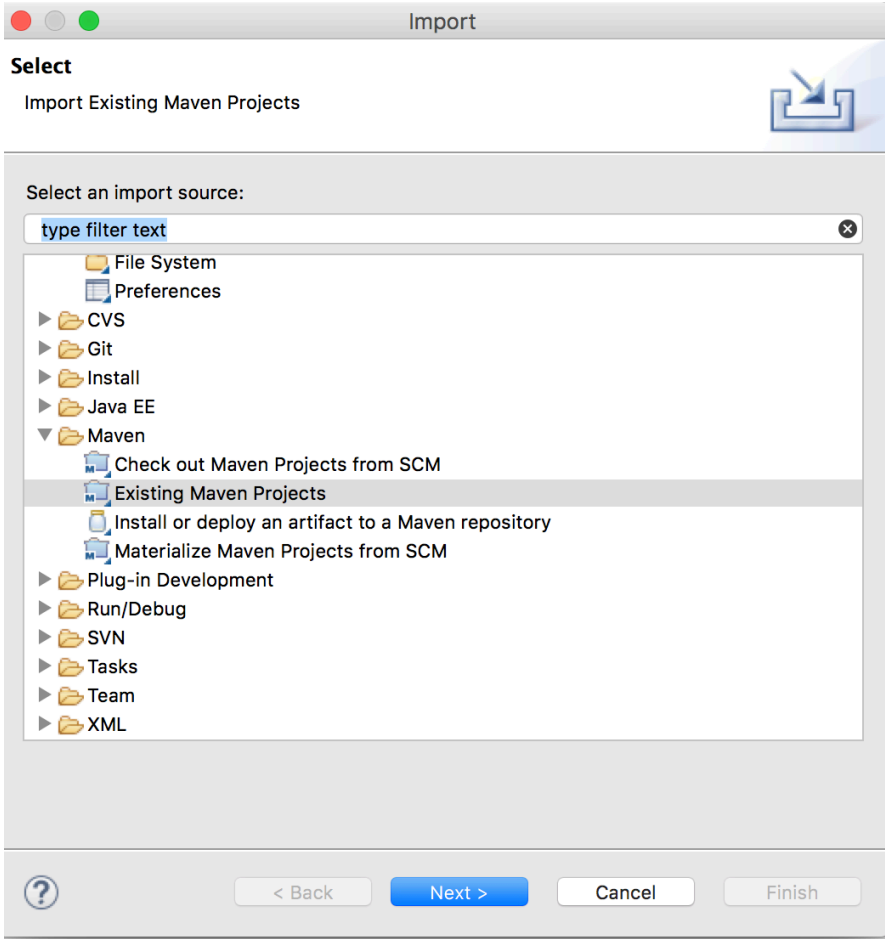
[3] N'oubliez pas de remplacer *your_username* par votre nom d'utilisateur **GitHub**.

Episode 3 - Création du projet Maven

Comme vu en cours, la création d'un projet d'un projet **Maven** consiste uniquement à écrire un fichier nommé `pom.xml` et d'y décrire les cycles de vie de notre application. Dans le cadre de projet il nous faut définir :

- Les méta informations (nom, version[4], etc …)
- Les cycles de vie, à minima pour le build et les tests
- La liste des dépendances, incluant les librairies suivante : **JUnit, Mockito**

Une fois le fichier `pom.xml` écrit, vous pouvez importer le projet dans Eclipse, à l'aide du menu `File > Import :`



Ensuite sélectionnez le dossier racine de votre projet et finalisez l'import.

[4] Pour les numéros de version, on suit les standards du **sementic versionning** : http://semver.org

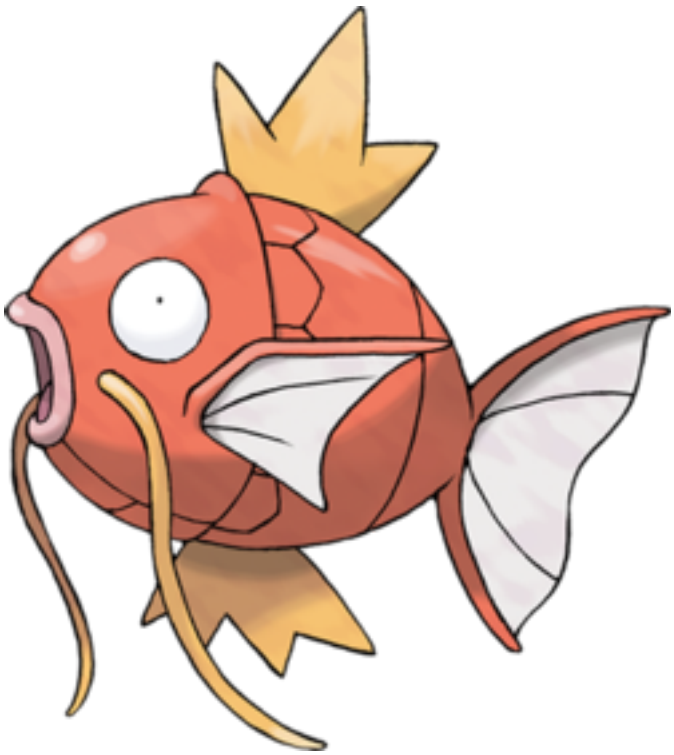
Episode 4 - Synchronisation avec Git

Nous avons maintenant un projet opérationnel pour travailler, il faut maintenant commuter les changements sur **GitHub**. Deux options :

- Ligne de commande
- Directement depuis **Eclipse** avec le plugin **eGit**.

Pour la ligne de commande vous pouvez vous servir de la documentation fourni en introduction. Pour **Eclipse**, il suffit de faire un clic droit sur le projet dans la vue **PackageExplorer**, puis `Team > Commit`.

Félicitation tu as gagné le badge suivant :



Magicommit[5]

[5] Plus que 400 commits, et tu seras enfin utile.