Projet

Une bibliothèque d'analyse de données en Java

Ce projet est une opportunité de travailler à plusieurs sur un projet d'envergure. Pour assurer la qualité du logiciel produit, vous mettrez en place une procédure d'intégration continue pour votre logiciel. Le projet s'intéresse à la mise en œuvre d'une bibliothèque de manipulation et d'analyse de données.

1 Informations importantes

- Le projet est à faire par groupe de deux à quatre personnes.
- **Les solutions ne pourront pas être partagées entre les groupes**.
- **Note** : Ce TP est noté.
- **Date limite** : Le TP est à terminer pour le **28 avril**.

2 Évaluation

Le projet sera noté en fonction de la qualité du code produit, du travail collaboratif (Pull/Merge Requests, revue de code), des fonctionnalités mises en œuvre, ainsi que de la maîtrise des outils de développement logiciel.

Veuillez noter que :
- La qualité du code produit sera évaluée notamment au travers du taux de couverture de code par les tests.
- L'évaluation des fonctionnalités tiendra compte du nombre de personnes travaillant sur le projet.

Mettre en oeuvre de manière correcte tous les attendus obligatoires en termes de développement logiciel et de fonctionnalités permet d'obtenir une *bonne* note pour ce projet (environ 13/20).

Veuillez noter aussi que le nombre de réalisations attendues dépend de la taille du groupe soumettant un projet.

Pour vous *aider* à vous orienter dans les réalisations optionnelles proposées en terme de développement logiciel et d'intégration/livraison continue, nous avons associé un niveau de difficulté à chacune d'entre elles (de 1 à 4, 4 étant le niveau le plus difficile). Bien sûr, les réalisations avec un plus haut niveau de difficulté rapportent plus de points.

3 Déroulement du TP

3.1 Résultats attendus

Le sujet du TP est très ouvert. Quelques-unes des fonctionnalités que vous pouvez explorer sont brièvement décrites dans la section 5. Cependant cette liste est loin d'être exhaustive. Si d'autres fonctionnalités vous semblent plus intéressantes, vous êtes libre de vous orienter vers celles-ci. Veuillez cependant vérifier auprès des enseignants que la fonctionnalité peut l'intéresser, avant de vous lancer dans sa réalisation.

De la même manière, plusieurs pistes vous sont proposées dans la section 4 pour enrichir votre projet sur les aspects intégration/livraison continue. Nous vous encourageons fortement à en explorer plusieurs.

Indication importante : Ceci n'est pas un projet uniquement centré sur le développement de fonctionnalités. Les aspects en lien avec la qualité du code, la collaboration entre développeurs, et les mécanismes d'intégration et de livraison continue sont au coeur de ce projet, et représenteront donc une part importante de votre note finale.

3.2 Rendu

Le rendu pour ce TP consistera en un simple fichier `txt` qui contiendra l'URL de votre projet sur `Github` (ou Gitlab), et si applicable, l'URL de votre dépôt sur `Docker Hub`.

Le fichier est à déposer sur Moodle en utilisant le lien prévu à cet effet.

3.3 Contenu de votre dépôt sur Github/Gitlab

Votre dépôt devra contenir au minimum, en plus du code source de votre projet :
- Un fichier `AUTHORS` avec la liste des contributeurs au projet.
- Un ficher `README` qui fournira une documentation aux utilisateurs, c'est-à-dire, une présentation de l'ensemble des fonctionnalités fournies et une explication de leur utilisation. Quelques remarques complémentaires sur le contenu du fichier `README` sont fournies dans la section 3.4.

3.4 Le fichier README

Le fichier `README` fera office de compte-rendu pour votre TP. Ainsi, il est attendu qu'il contienne au moins les informations suivantes :
- Une description des fonctionnalités fournies par votre service.
- Une description des choix d'outils que vous avez fait.
- Une description du workflow git que vous avez mis en place pour votre projet, et de la procédure de validation des Pull/Merge requests que vous avez adoptée.
- Le cas échéant, une liste et une courte description des images Docker produites et un lien vers leur dépôt.
- Une partie `feedback` dans laquelle vous donnerez votre retour d'expérience sur les différents outils utilisés durant le projet.

4 Développement logiciel et intégration continue

Avant de décrire le logiciel que vous allez devoir développer, nous donnons ici les principales instructions à suivre concernant le développement de votre projet.

4.1 Développement

Pour le développement de votre bibliothèque, vous utiliserez une méthode basée sur l'intégration continue. Pour cela, vous devez impérativement utiliser les outils suivants :
- `Git` pour versionner votre code source,
- `Maven` pour construire les différentes phases de votre projet,
- `JUnit` pour les tests unitaires,
- Un outils d'évaluation de le couverture de code intégré à `Maven` (e.g., EclEmma),
- Un dépôt `Github` ou `Gitlab` pour héberger votre code source versionné.

Remarque : Pour Gitlab, vous avez la possibilité d'utiliser l'instance publique (`gitlab.com`) ou l'instance hébergée à l'université (`https://gricad-gitlab.univ-grenoble-alpes.fr/`).

4.2 Mise en place Github / Gitlab

Voici comment procéder pour travailler à plusieurs sur un projet sur Github ou Gitlab :

1. Chaque membre du groupe doit se créer un compte sur le site de la plateforme,
2. Un des membres du groupe crée le projet (*repository*) sur son compte. Avec Github, il devient l'administrateur du projet.
3. Il ajoute ensuite les autres membres du groupe en tant que collaborateurs au projet. Sur la page principale du *repository* :
   - Pour Github : cliquez sur l'onglet *Settings*, puis sur l'onglet *Manage access*,
   - Pour Gitlab : cliquez sur *Members*, choisissez le rôle *Maintainer*.

NB : Sur Github, tous les collaborateurs peuvent participer au projet, mais seul l'administrateur peut associer le projet à un service externe (e.g., `Docker Hub`).

4.3 Intégration continue

Il vous est demandé de mettre en place un mécanisme d'intégration continue qui valide le bon fonctionnement de votre bibliothèque. Pour cela, vous utiliserez un workflow (`Github Actions` ou `Gitlab CI/CD pipelines`) qui lance les tests de votre bibliothèque et reporte les résultats de ces tests. De manière optionnelle, vous pouvez aussi configurer votre workflow pour reporter les résultats de la couverture de code.

NB : Il existe d'autre services dédiés à l'intégration continue (tels TravisCI ou CircleCI). Pour les besoins de ce projet, il vous est demandé, à minima, d'utiliser un workflow mais vous pouvez bien sûr utiliser ces autres solutions en complément.

4.4 Travail Collaboratif

Il est attendu de vous que vous utilisiez un méthode de travail collaboratif similaire aux méthodes discutés en cours (i.e., un workflow git).

Au minimum, vous devez adopter le principe des `features branches` avec revue de code. Concrètement, cela signifie que les changements significatifs ne seront pas directement poussés sur la branche principale. Ces changements devront passer par des features branches qui sont fusionnées dans la branche principale via une `Pull/Merge Request` par un **autre** collaborateur **après revue de code**.

De manière optionnelle, vous pouvez aussi configurer votre intégration continue pour qu'elle soit exécutée automatiquement pour chaque création/modification d'une Pull/Merge Request et que le résultat de cette exécution soit intégré dans la Pull/Merge Request. Vous en trouverez un exemple ici sur le dépôt de Pandas (il est nécessaire d'être connecté avec son compte Github pour voir l'intégration continue dans cet exemple). Vous pouvez aussi intégrer la couverture de code et ajouter une vérification du fait que le taux de couverture de vos tests est suffisamment élevé. Ces indicateurs supplémentaires (rapports de tests, taux de couverture) peuvent ainsi être utilisés par la personne en charge de valider la Pull/Merge Request pour prendre sa décision.

4.5 Livraison continue (Maven) [Niveau de difficulté : 2]

Github et Gitlab intègrent nativement un support pour la livraison continue des bibliothèques via leur outils de workflow et les dépôts Maven intégrés aux dépôts de code source (`Github Packages` et `Gitlab Package Registry`).

De manière optionnelle, vous pouvez déployer les versions compilées de votre bibliothèque dans le dépôt Maven associé au dépôt de votre code source. Ce déploiement devrait survenir à chaque fois que le code source est mise à jour dans le dépôt (version `SNAPSHOT`) et que tous les tests unitaires s'exécutent avec succès. Vous pouvez même envisager de conditionner la publication d'une nouvelle version de votre bibliothèque à un taux de couverture de code suffisant.

4.6 Livraison continue (Docker) [Niveau de difficulté : 3]

De manière optionnelle, vous pouvez mettre en pratique vos connaissances de Docker acquises lors des séances précédentes pour enrichir votre procédure de livraison continue par la construction et le déploiement d'une (ou plusieurs) image(s) Docker. Une bibliothèque n'étant pas très approprié pour créer une image Docker, nous vous proposons de travailler sur le scénario suivant :

- une image qui à l'exécution du conteneur déroulera un scénario de démonstration des fonctionnalités de votre bibliothèque.

Pour la construction et l'hébergement des images Docker, plusieurs solutions sont à votre disposition :

- Vous pouvez ajouter les étapes de construction et de déploiement de l'image Docker à votre workflow et déployer l'image dans le dépôt Docker intégré à Github (resp. Gitlab) : `Github Packages` (resp. `Gitlab Container Registry`).

- Vous pouvez créer un compte gratuit sur Docker Hub pour avoir votre propre dépôt public d'images Docker. En utilisant le service `Docker Hub`, vous avez la possibilité de laisser Docker Hub construire votre image (c.f., `Docker Hub automated build`) plutôt que d'utiliser un workflow.

4.7 Infrastructure-as-code et Cloud [Niveau de difficulté : 4]

Dans le contexte du cours DevOps, vous pouvez obtenir $50 de crédit sur Google Cloud (cf messages publiés précédemment).

Le site web `https://roparst.gricad-pages.univ-grenoble-alpes.fr/cloud-tutorials/` regroupe un ensemble de ressources vous expliquant comment utiliser ces crédits.

Sur ce site, vous trouverez notamment une description de l'outil `Terraform` permettant d'approvisionner de manière automatique des machines virtuelles dans le Cloud : `https://roparst.gricad-pages.univ-grenoble-alpes.fr/cloud-tutorials/terraform/`.

De manière optionnelle, vous pouvez envisager d'utiliser et d'adapter la procédure décrite pour déployer automatiquement des conteneurs correspondant aux images que vous aurez crées sur Google Cloud. En plus de `Terraform`, nous vous conseillons d'utiliser `Ansible` pour configurer automatiquement les machines virtuelles approvisionnées.

4.8 Insertion de Badges [Niveau de difficulté : 1]

Pour mettre en évidence la qualité du code de votre projet, il est possible d'insérer des *badges* au fichier `README` décrivant votre projet. Ces badges permettent, entre autres choses, d'afficher l'évaluation de votre projet du point de vue de l'intégration continue (workflow ou service externe). Le dépôt `Github` de la bibliothèque Pandas en python offre un bon exemple d'utilisation de ces badges.

Vous pouvez de manière optionnelle ajouter des badges pour *certifier* la qualité de votre projet. Au delà du badge sur l'intégration continue, vous pouvez envisager d'intégrer d'autres badges (couverture du code, qualité du code, etc.).

4.9 Valorisation de votre bibliothèque [Niveau de difficulté : 2]

De manière optionnelle, vous pouvez mettre en place un site web, hébergé à l'aide de `Github Pages` / `Gitlab Pages` pour valoriser votre bibliothèque. Plusieurs solutions s'offrent à vous pour le contenu de ce site web, vous pouvez :

- créer une page Web (succincte, le Web ne faisant pas parti des objectifs de l'enseignement),
- utiliser votre fichier README (converti automatiquement au format html),
- utiliser le plugin Maven `maven-site-plugin` qui génère automatiquement un site web,
- générer et héberger la documentation (javadoc) de votre bibliothèque (cette dernière option peut aussi être combinée avec l'une des trois précédentes).

Dans tous les cas, vous pouvez intégrer la génération et le déploiement de ce site web à votre workflow pour qu'ils soient réalisés automatiquement.

NB : Avec Github, il est nécessaire de rendre le projet publique pour pouvoir utiliser (gratuitement) `Github Pages`.

4.10 Utilisation d'autres services Github

De nombreux autres services sont accessibles gratuitement depuis Github au delà des services d'intégrations continue. Dans le cadre ce projet, vous êtes encouragés à expérimenter certains de ces services, par exemple pour la couverture de code ou la revue automatique de code. Vous trouverez la liste des services disponibles depuis le marketplace Github.

5 Bibliothèque d'analyse de données

La capacité de générer de la connaissance, et donc de la valeur, à partir d'un ensemble de données brutes est centrale dans notre monde *connecté* où des quantités gigantesques de données sont produites à chaque instant. Dans ce contexte, les langages et bibliothèques permettant de traiter simplement et efficacement de grandes quantités de données deviennent très populaires.

Parmi ces outils pour le traitement de grande quantités de données, le langage Python et plus particulièrement le package `Pandas` est une solution très appréciée des informaticiens. Il n'existe pas de solution équivalente au package `Pandas` pour le langage Java. Durant ce projet nous vous proposons de mettre en oeuvre une petite sous-partie des fonctionnalités offertes par `Pandas` en Java.

L'objectif sera de développer votre propre bibliothèque d'analyse de données en Java en suivant les consignes et recommandations décrites dans la section 4 pour assurer la qualité du code produit.

Dans la suite, nous décrivons quelques fonctionnalités qu'il nous semble intéressant d'intégrer à votre bibliothèques. Vous êtes bien entendu autorisés et invités à en intégrer d'autres.

5.1 Fonctionnalités obligatoires

Dans cette partie, nous décrivons quelques fonctions que vous avez l'obligation d'intégrer à votre bibliothèque dans le contexte de ce projet.

Dataframe : Le type d'objet principal manipulé par Pandas sont les Dataframe. Les Dataframes sont des tableaux en deux dimensions où chaque colonne est identifié par un label et chaque ligne par un index. Chaque colonne stocke des données d'un seul type. Cependant deux colonnes différentes peuvent stocker des types différents.

Votre bibliothèque devra supporter la création de dataframes selon au moins 2 méthodes différentes :

- A partir d'un constructeur prenant en paramètre le contenu de chaque colonne sous forme d'une structure de données simple (par exemple un tableau).
- A partir d'un constructeur prenant en paramètre le nom d'un fichier `CSV`[1] qui devra être *parsé* au sein de ce constructeur. Dans ce cas, vous devrez déduire du format des données dans le fichier CSV, le type de données à créer pour chaque colonne.
  - Suggestion : Si déduire automatiquement les types vous semble trop compliqué, vous pouvez envisager d'adapter le format des fichiers que vous utiliser en disant que la première ligne du fichier permet de définir le type de chaque colonne.

Affichage d'un dataframe Vous devrez fournir à l'utilisateur des méthodes pour afficher un dataframe avec plusieurs variantes :

- Afficher tout le dataframe
- Afficher seulement les premières lignes
- Afficher seulement les dernières lignes

Sélection dans un dataframe Vous devez fournir à l'utilisateur des méthodes pour créer un nouveau dataframe en sélectionnant un sous-ensemble des données d'un dataframe existant. Vous devez au moins permettre de sélectionner :

- Un sous-ensemble de lignes à partir de leur index (voir ici pour plus d'informations)
- Un sous-ensemble de colonnes en utilisant les labels (voir ici pour plus d'informations)

En plus de ces deux types de sélection simples, vous devez intégrer au moins un mécanisme de sélection *avancé*. Référer vous à la documentation de `Pandas` pour choisir un mécanisme avancé qui vous semble pertinent.

1. A propos du format CSV, voir `https://fr.wikipedia.org/wiki/Comma-separated_values`

Statistiques sur un dataframe Vous devez fournir à l'utilisateur la possibilité d'exécuter des calculs de statistiques de base sur les colonnes d'un dataframe, tels que calculer la moyenne des valeurs, trouver le minimum/maximum, etc.

5.2 Fonctionnalités optionnelles

Vous trouverez ci-dessous quelques suggestions concernant d'autres fonctionnalités à ajouter à votre bibliothèque.

- Pour toutes les catégories de fonctionnalités décrites dans la section 5.1, vous avez bien sûr la liberté d'ajouter d'autres fonctionnalités que celle déjà mentionnées si elles vous semblent intéressantes.
- Pour une analyse avancée des données, il est intéressant de pouvoir regrouper les données selon des critères pour ensuite appliquer des opération sur ces groupes. Dans le package Pandas, la fonction `GroupBy` permet de regrouper les données d'un Dataframe selon un critère. La fonction `Aggregate` permet ensuite d'appliquer une opération sur les éléments de chacun des groupes créés. Travailler sur ces manipulations avancées de Dataframes peut être une direction de travail pour l'extension de votre bibliothèque.
- Plus généralement, vous pouvez vous référer à la documentation du package Python `Pandas` pour trouver d'autres fonctionnalités que vous aimeriez mettre en œuvre.