Projet GL _ GSON

- **Yannick Mounguengui**
- **Nouha Boukili**

1] Présentation globale du projet

1.1. Utilité du projet

. Quel projet on a choisis?

On a choisi d'étudier la qualité logicielle du projet Gson qui se trouve dans le lien qui suit : https://github.com/google/gson

. Que fait le projet?

Gson est une libraire qu'on peut ajouter aux projets java dans le but de convertir des objets Java vers leurs représentations JSON. Ce n'est pas le premier projet open-source qui s'intéresse à la conversion des annotations Java vers Json, pourtant ce projet met en valeur deux points très importants contrairement aux autres, l'utilisation des Java Generics (with the aim of reducing bugs and adding an extra layer of abstraction over types) et aussi la possibilité de convertir vers JSON sans ajouter les annotations Java dont on a pas accès à leurs code source tout le temps.

. Quels sont les buts du projet?

- **fournir les méthodes toJson() et fromJson() pour convertir à JSON et vice-versa.**
- **permettre les objets Java qui définis précédemment comme étant non modifiables à se convertir vers JSON.**
- **utiliser les java Generics.**
- **permettre les représentations clients aux objets**
- **prise en charge d'objets arbitrairement complexes (avec des hiérarchies d'héritage profondes et une utilisation intensive de types génériques)**

. Comment lancer le projet?

Comme le projet est sous forme d'une bibliothèque Java, on peut l'installer en l'ajoutant aux dépendances en deux manières :

façon 1 : par les dépendances du pom.xml pour un projet Maven comme suit :

Maven:

```
<dependency>
  <groupId>com.google.code.gson</groupId>
  <artifactId>gson</artifactId>
  <version>2.10.1</version>
</dependency>
```

façon 2 : par les dépendances Gradle :

```
dependencies {
  implementation 'com.google.code.gson:gson:2.10.1'
}
```

. Quel est le résultat du projet : quelles sont les sorties du logiciel ?

Le but principal du projet c'est pouvoir utiliser les méthodes qui convertissent au Json.
La classe principale à utiliser est Gson qu'on peut simplement créer en appelant new Gson(). Il existe également une classe GsonBuilder disponible qui peut être utilisée pour créer une instance Gson avec divers paramètres tels que le contrôle de version, etc.

L'instance Gson ne conserve aucun état lors de l'appel d'opérations JSON. Ainsi, on est libre de réutiliser le même objet pour plusieurs opérations de sérialisation et de désérialisation JSON.

On peut donner comme exemple d'utilisation du projet ce qui suit:

```
// Serialization
Gson gson = new Gson();
gson.toJson(1);             // ==> 1
gson.toJson("abcd");        // ==> "abcd"
gson.toJson(new Long(10)); // ==> 10
int[] values = { 1 };
gson.toJson(values);        // ==> [1]

// Deserialization
int i = gson.fromJson("1", int.class);
Integer intObj = gson.fromJson("1", Integer.class);
Long longObj = gson.fromJson("1", Long.class);
Boolean boolObj = gson.fromJson("false", Boolean.class);
String str = gson.fromJson("\"abc\"", String.class);
String[] strArray = gson.fromJson("[\"abc\"]", String[].class);
```

1.2. Description du projet

. Le readme est-il pertinent?

Il y a bien un README dans ce projet, qui de plus est à jour (dernière modification effectuée le 6 janvier 2023 pour annoncer la sortie du gson-parent-2.10.1). Ce dernier présente globalement le projet sans entrer dans les détails. On peut ainsi retrouver une brève introduction pour se mettre dans le contexte, les objectifs du projet ou encore des liens utiles vers des documentations.

. La documentation du projet est-elle présente?

En plus de l'extrait de documentation retrouvée dans le README, il y a à disposition une documentation supplémentaire détaillée et complète du projet dans le fichier UserGuide.md. On y retrouve également des informations sur

- **l'installation**
- **l'utilisation : nous constatons qu'il y a suffisamment d'exemples permettant d'illustrer clairement comment utiliser Gson sans avoir besoin de consulter le code source**
- **ou encore l'utilité du projet.**

. Les informations en terme de lancement sont-elles suffisantes ?

Toutefois, les informations en termes d'installation et de lancement ne sont pas très explicites, on peut avoir des bugs lors de la première installation.

2] Historique du projet

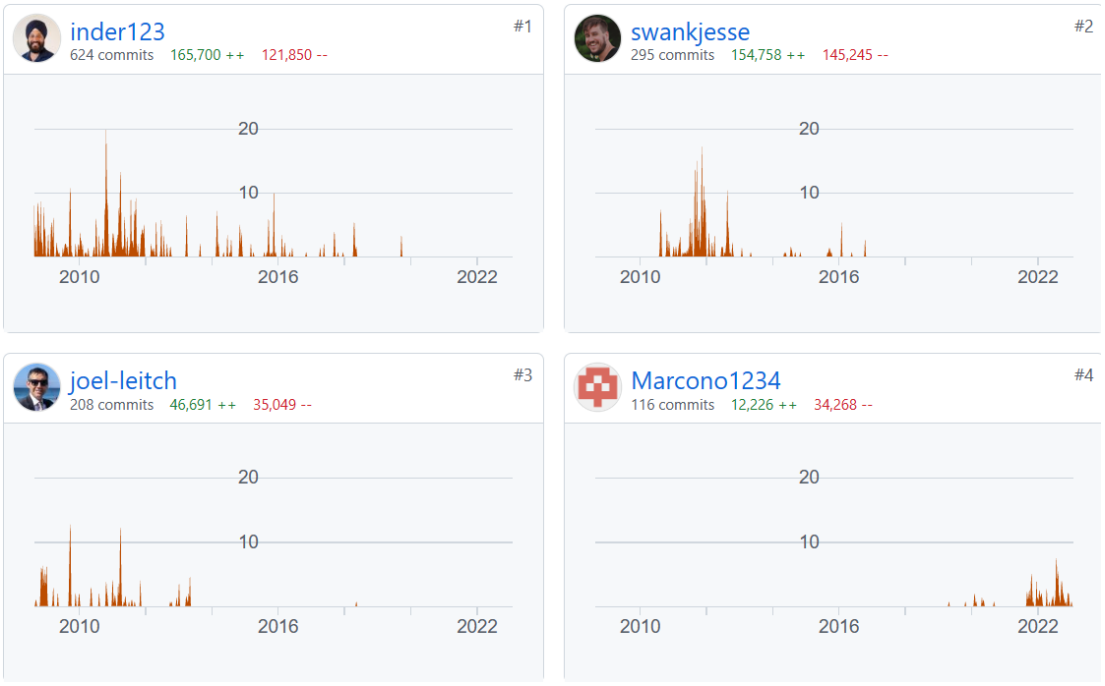
2.1. Analyse du git

. Quel est le nombre de contributeurs?

D'après le dépôt Github, 137 personnes contribuent pour l'instant au développement du projet.

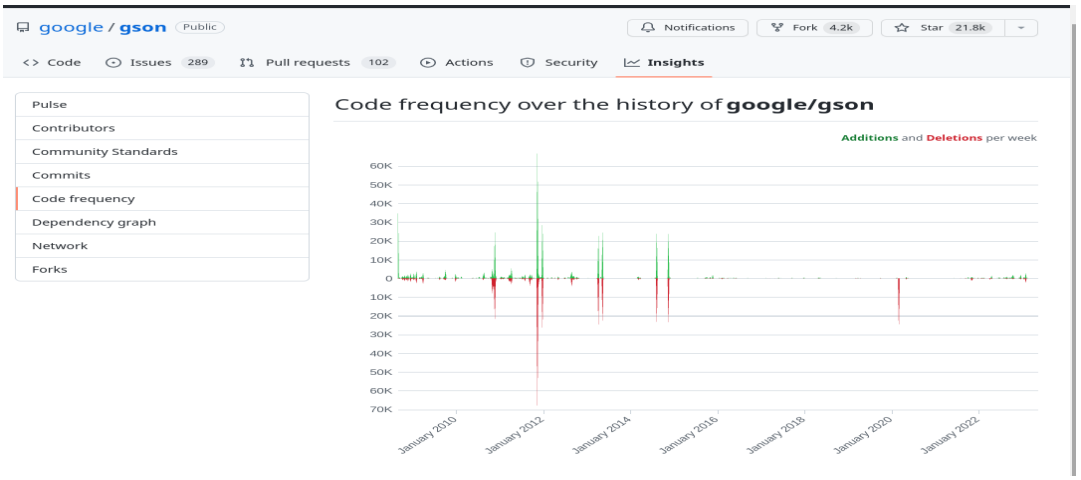
. Est ce qu'il ont contribué de façon équilibré sur le projet? et dans le temps?

Non, on peut constater clairement que les contributions ne sont pas équilibrées, on trouve certains qui ont fait 624 commits et d'autres qui ont fait qu'un seul.
Aussi, on trouve une majorité de personnes à travailler en 2010, contrairement à d'autres qui ont commencé en 2022.
On donne un exemple des contributions d'utilisateurs dans l'image qui suit:



. Le projet est-il toujours actif? l'activité est-elle répartie régulièrement sur le temps?

Le projet est toujours actif, on trouve des modifications faites le jour même ou on vérifie, mais l'activité n'est pas régulière, on trouve que les ajouts et les suppressions en 2012 sont beaucoup plus nombreux par rapport à ces dernières années.



Combien de branches sont définies? et combien parmi elles sont utilisées?

Le projet est constitué de 15 branches dont 6 sont closed et le reste est open.

. Le mécanisme de pull requests est-il utilisé?

Il y a 102 demandes de pull qui sont en attente et 627 demandes qui sont closed

3] Architecture Logicielle

3.1. Utilisations des bibliothèques extérieurs

. Quel est le nombre des bibliothèques extérieures?

(à faire..)

4] Analyse Approfondi

4.1. Tests

.Combien de classes de tests existent-ils? combien de méthodes de tests? Combien d'assertions? Que peut-on en déduire sur la structuration des tests?

Ils existent 128 classes de tests.