Rapport de projet:

Spécifications techniques et implémentation en application Java : “Package Viewer”

I. Étude de conception **1**

II. Synthèse **1**

III. Bilan technique **2**

1. RoadMap 2
2. Api utilisées 3
3. Test unitaires / CI 4

IV. Problèmes rencontrés **5**

V. Écarts avec les prévisions **5**

VI. Mesures d’amélioration **5**

I. Étude de conception

Notre projet a pour finalité de créer un programme qui permet de rechercher et de visualiser un paquet linux sous plusieurs distributions : Arch Linux, Ubuntu et Fedora. Ce programme permettait de visualiser les métadonnées liée au paquet, s'il est trouvé, pour la distribution donnée, c'est-à- dire la version disponible dans les dépôt officiel de la distribution, une description si la distribution en fouine une ainsi que l'arbre des dépendances nécessaires à l' installation du paquet. Ce programme fournira deux manières d'interagir avec un utilisateur, le premier est via une interface CLI (Command Line Interface) très classique sur les systèmes unix. L'utilisateur utilisera les différents paramètres mis à disposition pour exécuter ses requêtes et pourra, s'il le souhaite, intégrer notre programme a ses scripts à sa guise. La partie CLI sera inspirée du programme "YAY" (Yet Another Yogurt) qui étends les fonctionnalités du gestionnaire de paquet d'Arch Linux "pacman", bien évidemment notre programme ne va pas installer les paquets et se contentera uniquement d'afficher les métadonnées trouvées. La seconde manière d'interagir avec notre programme serait via une interface TUI (Terminal User Interface) utilisant la bibliothèque "ncurse" tout autant répandue dans les système unix, elle permettrait à l'utilisateur d'effectuer les même opérations que via le CLI mais de manière plus graphique tout en restant dans le terminal.

II. Synthèse

Le programme que l'on a fourni, avec le peu temps qu'on a eu. satisfait les espérances que l'on a posées. Certes le programme est loin d'être fini mais nous n'imaginons pas réaliser toute l'application en si peu de temps. Le programme respecte le contrat basique que l'on a posé : pouvoir rechercher un paquet et voir ses informations et ses dépendances.

III. Bilan technique

1. RoadMap

Voici une road map des fonctionnalité initiales que nous voulons implémenter ainsi que la répartition du travail :

Parseurs:

- **[X]** Parseur des dépôts Arch Linux [Marla]
- **[ ]** Parseur des dépôts Ubuntu
- **[X]** Parseur des dépôts Fedora [Djalim]

Interface CLI:

- **[X]** Afficher le paquet via son nom exact [Marla]
- **[X]** Afficher un arbre des dépendances [Marla]
- **[X]** Rechercher les paquets via un pattern dans les dépôts [Marla]
- **[ ]** Affichage des résultats de maniere a etre parser dans un script [Marla]

Interface TUI:

- **[ ]** Menu de recherche
- **[ ]** Menu de résultats
- **[ ]** Menu de détail sur un paquet
- **[ ]** Arbre navigable des dépendances d'un paquet

Autre:

- **[X]** Recherche des api et leurs documentations [Djalim]
- **[X]** Documentation [Djalim]
- **[X]** Requête concurrente [Thomas]
- **[X]** Tests unitaire et leur automatisation (CI) [Thomas]
- **[ ]** Cache pour utilisation hors ligne
- **[X]** Dossier [Djalim]

2. Api utilisées

Pour implémenter notre programme nous avons utiliser les apis suivantes:

- Arch Linux (dépôts officiels)

Pour Arch linux nous avons utilisé l'api officiel pour la recherche de métadonnées et de fichier des paquets, la documentation est disponible est disponible sur leur wiki.
Cette api nous permet de rechercher un paquet dans les dépôts d'arch linux et ,si l'on possède son nom exact, tous les détails sur ce paquet.

Exemple : Rechercher les paquet contenant bash dans leur nom :

Requete: GET https://www.archlinux.org/packages/search/json?q=bash

Réponse:


```
{"version": 2,
      "limit": 250,
      "valid": true,
      "results": [
            {
                  "pkgname": "argbash",
                  "pkgbase": "argbash",
                  "repo": "community",
                  [...]
            },
            {
                  "pkgname": "bash",
                  "pkgbase": "bash",
                  "repo": "core",
                  [...]
            },          {
                  "pkgname": "bash-bats",
                  "pkgbase": "bash-bats",
                  "repo": "community",
                  [...]
            }
            [...]
      ]
}
```


- Fedora

Afin d'implémenter Fedora nous avons utilisé deux api différentes, une pour la recherche de paquet, une pour l'obtention des métadonnées liés aux paquets. Le recherche de paquet se fait via l'api de Pagure, un clone de git sur lequel fedora héberge les sources des paquets disponibles dans leur dépôts. La documentation est disponible sur leur site. La seconde api

est la MDAPI (MetaData API), comme son nom l'indique elle fournit les métadonnées sur les paquets hébergés sur pagure. A notre grand désespoir cette api et sa documentation (disponible a la racine des endpoints) est très mal référencée par les moteur de recherches, nous avions déjà commencé à créer un parseur pour les fichier ".spec" se trouvant à la racine de toute les sources sur pagure avant de trouver cette api qui fait exactement ce que l'on voulait faire.

Exemple : Information sur le paquet bash
Requete : GET https://mdapi.fedoraproject.org/rawhide/pkg/bash
Réponse :


```
{
	"arch": "x86_64",
	"epoch": "0",
	"version": "5.2.9",
	"release": "3.fc38",
	"summary": "The GNU Bourne Again shell",
	"description": "The GNU Bourne Again shell (Bash) is a shell or
command language\ninterpreter that is compatible with the Bourne shell
(sh). Bash\nincorporates useful features from the Korn shell (ksh) and
the C shell\n(csh). Most sh scripts can be run by bash without
modification.",
	"basename": "bash",
	"provides": [
		{
			"name": "bash",
			"epoch": "0",
			"version": "5.2.9",
			"release": "3.fc38",
			"flags": "EQ"
		},
		[...]
	]
}
```


3. Test unitaires / CI

Nous avons réalisé des test unitaires qui testent le bon fonctionnement interne de notre projet, afin de s'assurer qu'il y ait toujour une version fonctionnel dans la branche main ainsi qu'il n'y ait pas de régression lorsqu'on merge des pull request, nous avons créé une CI github qui test si l'application se build et si elle passe toujours les tests.

IV. Problèmes rencontrés

Nous avons rencontré plusieurs problèmes lors de la conception de ce programme, le premier étant que l'une des trois distribution majeure que l'on voulait implémenter ne dispose pas d'une api facile à implémenter dans un programme java. Initialement nous voulions partir sur l'api de launchpad, qui héberge les paquets ubuntu. Mais certaines des requêtes nécessaires à notre implémentation d'un parseur ne renvoyait pas de réponses interprétables en java. Le second problème fut l'architecture complète du programme, nous avons passé beaucoup de temps à discuter sur la forme interne du programme avant de commencer à programmer. Après il y a eu le fait que l'api de fedora ne fournissait pas de métadonnée sur leurs paquet et qu'il fallait écrire manuellement un parseur de fichier ".spec" tres peu fonctionnel qui avait beaucoup de problèmes, avant de simplement supprimer le code et d'adapter le code pour utiliser une meilleur api.

V. Écarts avec les prévisions

Le seul écart notable que l'on peut noter est le fait que l'on a pas pu programmer de parseur pour les dépôts d'ubuntu.

VI. Mesures d'amélioration

Après toutes les étapes de réflections que l'on a eu, nous pensons avoir convergé vers une solution optimale. Actuellement nous ne voyons pas de manière d'améliorer le code.