M1 Informatique AIGLE

HMIN201

M1 TER

TER Software Heritage

Rapport Final

Groupe Bajonim

Bachar Rima,
bachar.rima@etu.umontpellier.fr
Joseph Saba,
joseph.saba@etu.umontpellier.fr
Tasnim Shaqura,
tasnim.shaqura@etu.umontpellier.fr

Encadrant :
Jessie Carbonnel

Responsable de l'UE :
Mattieu Lafourcade

27 mai 2019

Table des matières

Chapitre 1

Introduction

Les logiciels sont actuellement omniprésents dans tous les aspects de notre vie quotidienne ; ils constituent l'un des piliers de l'héritage humain et doivent être préservés contre toute suppression et tout endommagement. Archiver leurs codes source s'avère ainsi une tâche primordiale. En effet, le code source d'un logiciel constitue un artefact logiciel essentiel dans le domaine des connaisances scientifiques, culturelles, et techniques. D'autre part, le code source est facilement lisible et compréhensible par les humains, et peut être transformé en fichiers exécutables. À ce titre là, des plateformes ont déjà été proposées telles que `The Internet Archive` et `UNESCO Persist`. Toutefois, ces plateformes se concentraient plutôt sur la préservation des fichiers exécutables au lieu du code source [1][2].

1.1 Description de Software Heritage

`Software Heritage` est une initiative lancée par **INRIA** [1], soutenue par l'**UNESCO** et visant « la collecte, la conservation et le partage de code source de tous les logiciels publiquement accessibles depuis n'importe quelle plateforme d'hébergement de code source »[3].

Son architecture consiste en un *framework* permettant de retrouver le code source des logiciels susmentionnés et de les ingérer au sein de l'archive universel de `Software Heritage`. En particulier, les **Listers** en constituent une partie centrale : il s'agit de *crawlers* configurés pour parcourir des dépôts de code source, « *mapper* » leurs modèles à des modèles intégrables à l'infrastructure, et reporter l'ingestion de leur contenu à d'autres composants du *framework*. L'ingestion du contenu d'un dépôt « *listé* »au sein de l'archive est effectuée par des composants spécifiques, les **Loaders**. Enfin, la planifi-

1. **Institut National de Recherche en Informatique et Automatique**

cation des tâches du *listing* et du *loading* est régulée par un **Scheduler**, un composant interagissant avec une queue de tâches asynchrones opérée par un serveur `Celery`.

Il faut préciser que les plateformes d'hébergement embarquent chacune des dépôts de code source à structures différentes, ce qui nécessite la création d'un **Lister** dédié pour chaque plateforme. Par ailleurs, les différentes versions d'un logiciel et leurs métadonnées associées sont gérées par un gestionnaire de version, ce qui nécessite la création d'un **Loader** dédié pour chaque gestionnaire. Actuellement, tous les **Listers** et **Loaders** ont été créés uniquement par l'équipe de `Software Heritage`. Les **Listers** développés l'ont été pour les plateformes d'hébergement les plus populaires (`Github`, `Bitbucket`, ...). De même, les **Loaders** développés l'ont été pour les gestionnaires de version les plus populaires (`Git`, `SVN`, `Mercurial`, ...).

1.2 Contexte du TER

Dans le cadre de ce projet, encadré par Jessie Carbonnel, du module **HMIN201** désignant le TER, encadré par Mathieu LaFourcade, notre objectif final consiste à créer un **Lister** pour une plateforme de développement ciblée. Ainsi, les tâches nécessaires à effectuer afin d'accomplir ce but peuvent être énumérées de la manière suivante :

- Lire et comprendre les articles et tutoriels écrits par l'équipe de `Software Heritage` ;
- Analyser différentes plateformes d'hébergement afin d'en cibler une ;
- Concevoir et développer un **Lister** pour la plateforme choisie ;
- Répliquer localement l'environnement de `Software Heritage` afin de tester le **Lister** développé ;
- Faire une *Pull Request* afin d'intégrer le **Lister** testé au dépôt de développement de `Software Heritage` sur `GitHub`.

1.3 Plan du rapport

Nous initions ce rapport par une introduction de notre projet consistant d'une une courte description de `Software Heritage`, suivie par la spécification du contexte du stage. Ensuite, nous détaillerons la problématique générale traitée par `Software Heritage` et la sous-problématique particulière adressée par notre projet.

Après, nous commencerons la section d'analyse par une explication technique détaillée de l'infrastructure de `Software Heritage` et de son fonction-

nement, l'étape fondamentale sur laquelle se base notre méthodologie. Nous terminerons cette section par le partage du planning prévisionnel du projet.

Par la suite, nous rentrerons dans le vif du sujet en détaillant nos approches pour la conception et l'implémentation d'un **Lister**, suivie des résultats obtenus.

Pour la conclusion, nous comparerons les versions prévisionnelle et finale du planning, puis nous discuterons les difficultés rencontrées et les perspectives du projet. Pour finir, nous listerons un bilan du projet en citant les apports techniques et personnels assimilées.

Chapitre 2

Problématique

2.1 La diaspora du code source

2.2 La fragilité du code source

2.3 Software Heritage en tant que solution

Current status et roadmap de SWH

2.4 Notre contribution

Chapitre 3

Analyse

3.1 Fonctionnement de Software Heritage

3.1.1 Modèle des données

Plateformes d'hébergement

Artéfacts logiciels

Informations sur la provenance des données

Structure des données

3.1.2 Architecture et flot des données

Flot d'ingestion des données

Listing

Loading

Scheduling

3.1.3 L'archive

Stockage des noeuds BLOB de l'archive

les IDs, les fichiers

Stockage des autres noeuds des archives

chemins, repertoires, snapshot, revisions, releases
postgres, etc

Stockage haché des objets

Mise en mirroir des noeuds

Politique de rétention

Récupération automatique des objects corrompus

3.2 Méthodologie

Sourceforge sitemap, api
Launchpad api, client
Analyzing the listers (bitbucket, gitlab, github, eclipse, LIRMM, OpenHub, Assembla, GNU savannah
heritage, injection de dependances
conclusion : on adoptera une strategie pour definir un lister, loader ou autre

3.3 Planning Prévisionnel

Chapitre 4

Conception

design de la solution proposée (diagrammes + explications)

Chapitre 5

Implémentation

les technos qu'on a utilisé
bibliotheques
Outils (e.g. XML parsers)
Launchpad client

Chapitre 6

Résultats

pull request ?

Chapitre 7

Conclusion

7.1 Planning final

7.2 Difficultés rencontrées

7.3 Perspectives

7.4 Bilan et apports du TER

annexes
resumés
code

Bibliographie

[1] The internet archive software collection. https://archive.org/details/software&tab=about. Accessed : 2019-05-23.

[2] About persist : Unesco persist programme. https://unescopersist.org/about/. Accessed : 2019-05-23.

[3] Roberto Di Cosmo and Stefano Zacchiroli. Software Heritage : Why and How to Preserve Software Source Code. In *iPRES 2017 - 14th International Conference on Digital Preservation*, pages 1–10, Kyoto, Japan, September 2017.