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

1.1 Description de Software Heritage

1.2 Contexte du TER

1.3 Plan du rapport

Chapitre 2

Problématique

Les logiciels sont actuellement omniprésents dans tous les aspects de notre vie quotidienne ; archiver leurs codes source paraît ainsi une tâche primordiale. 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 fichier exécutables ; allant jusqu'à offrir des émulateurs pour permettre l'éxecution des logiciels présents dans leurs archives. Par comparaison, Software Heritage s'interesse au code source des logiciels, pas à leurs éxecutables. En effet, le code source d'un logiciel constitue un artefact logiciel fondamental dans le domaine des connaisances scientifiques, culturelles, et techniques. Le code source est écrit sous une forme compréhensible par les humains, et peut façilement être transformé en une forme éxecutable par une machine. Le code source est muable et évolue celon les besoins. Ca préservation nous permet d'accéder à l'historique du developement d'un logiciel.

Malgrés son importance dans notre vie quotidienne, il est façile de voir que nous prennons pas soin correctment du code source. Cela est dû à trois raisons principales.

2.1 La diaspora du code source

Le quantité de projets open sources a vu un énorme accroissement pendant les deux dèrnieres décennies. Le code source de ces projets sont souvent développés sur des plateformes d'hébergement publiques(comme *Github* et *BitBucket*), ou sur des divers forges institutionelles. Beaucoup d'options s'offrent aux développeurs pour distribuers leurs logiciels. La distribution peut se faire sur des plateforms comme *Github*. Elle peut se faire via des archives liés à des ecosystemes spécifiques, comme *CTAN*, qui distribue des

logiciels pour TeX. Les développeurs peuvent aussi choisir de publier leurs logiciels sur des distributions comme *Debian* et *Fedora*, ou via un gestionaire de paquets comme *npm* et *pip*.

2.2 La fragilité du code source

Le code source est une entité fragile. Elle peut être façilement détruite ou perdue si elle n'est pas fréquement sauvegardée. Les plateforms d'hébergement ne guarantient pas forcement la préservation de leurs contenus ; des grandes plateformes d'hébergement ont déja arreter leurs services.

2.3 Software Heritage en tant que solution

Software Heritage a été créer pour relever ces défis. Software Heritage vise a fournir une infrstructure qui permet la collection, l'organisation, le préservation, et l'accés à tout code source publique. L'archive doit avoir la capacité d'accomplir ses objectifs pour toutes plateformes de dévelopement et de distribution, et doit pouvoir persister les codes source sur le long terme.

Current status et roadmap de SWH

2.4 Les défis

Identifier les plateformes d'hébergement : Les projets peuvent être hébergés sur des plateformes bien connues, comme sur des plateformes obscures. Il faut construire un catalog des plateformes. **Supporter differents protocoles :** Software Heritage doit pouvoir recupérer leurs projets et doit pouvoir maintenir les modifications faites sur ces projets. Vu qu les plateformes d'hébergements sont hétérogènes, Software Heritage essayera de promouvoir des bonnes pratiques pour le préservation. **Parcourir les historiques de développement :** Les plateformes d'hébergements supportent differents logiciels de gestion de versions qui n'ont pas les même modèles de données. Software Heritage construira un tel modèle unifiant.

2.5 Les principes de base

2.5.1 Transparence et gratuité

Pour pouvoir assurer la préservation de l'archive sur le long terme, il faut que tout les élements formant l'archive soient open source et accessibles au

publique.

2.5.2 Réplication compréhensive de l'entiereté du système

Un tel archive est soumis à differents types de risques. Ces risques étant inévitables, le système doit pouvoir les tolérer. Le système sera répliqué sur differents niveaux : differentes localisations géographiques, differents matériels de stockage , etc...

2.5.3 Multitude des partenaires, sans profits

Pour atteindre ses objectifs, Software Heritage ne doit pas dépendre sur une seule entité qui cherche d'en profiter, et ne doit pas être créer pour la géneration de profits. Ce projet doit apporter de la valeur au public en large, et non seulement pour les organisations qui le supporte.

2.5.4 Pas de présélection

Il est impossible de savoir quel projets vont finir par être les plus importants. Il faut doc préserver tout les logiciels disponibles sans présélection, surtout que la capacité technique de faire cela est disponible.

2.5.5 Source Code First

Bien qu'il est intéressant de garder le contexte du code source (comme les Wiki, l'environement où le programme est éxécuté, etc), une telle tâche nécessite une énorme quantité de resources, surtout qu'il n'y a pas de préselection. Software Heritage se contente d'archiver les code sources ainsi que leur historique de développement capturé par les logiciels de gestion de versions. Cela permet de guarder des informations importantes qu'on retrouve dans les messages de commit.

2.5.6 Identifiants Intrinsèque

Les identifiants des objets stockés ne doivent pas dépendre des sources externes et doivent pouvoir être calculés à partir des objets qu'ils identifient. Ils sont étroitement liés à ces objets. Cela permet de verifier que l'objet obtenu correspond à l'objet demandé, et permet la détection les modifications sur l'objet.

2.5.7 Informations de provenance et informations factuelles

Software Heritage va stocké les informations de provenance qui décrivent le ou, le quoi, et le quand des objets dans l'archive. Ces informations vont être verifiés et les méthodes de verification vont être stockées aussi.

2.5.8 Minimalisme

Software Heritage se contentera de construire l'infrastructure essentielle et rien de plus.

2.6 Notre travail

Dans le cadre de ce projet, encadré par Jessie Carbonnel, nous avons cibler la plateforme d'hébergement Launchpad afin de collectioner les codes sources qui y sont hébergés, et les stocker dans l'archive de Software Heritage. Ainsi, les objectifs de ce TER peuvent être énumérés de la manière suivante :

- Lire/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.

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 Listers de Software Heritage et les codes qui les accompagnent sont écrits en Python.

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

Chapitre 6

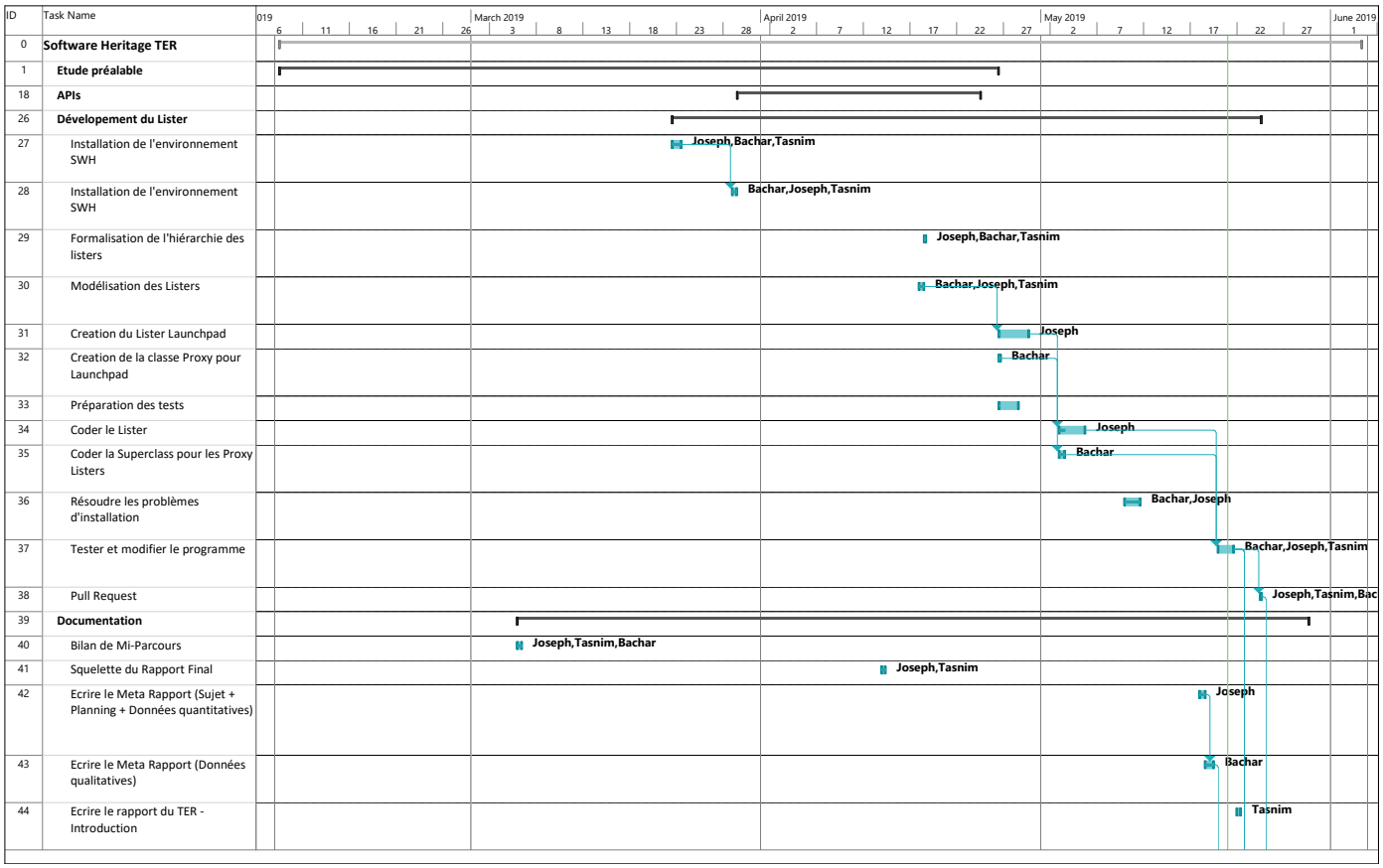
Résultats

pull request ?

Chapitre 7

Conclusion

7.1 Planning final



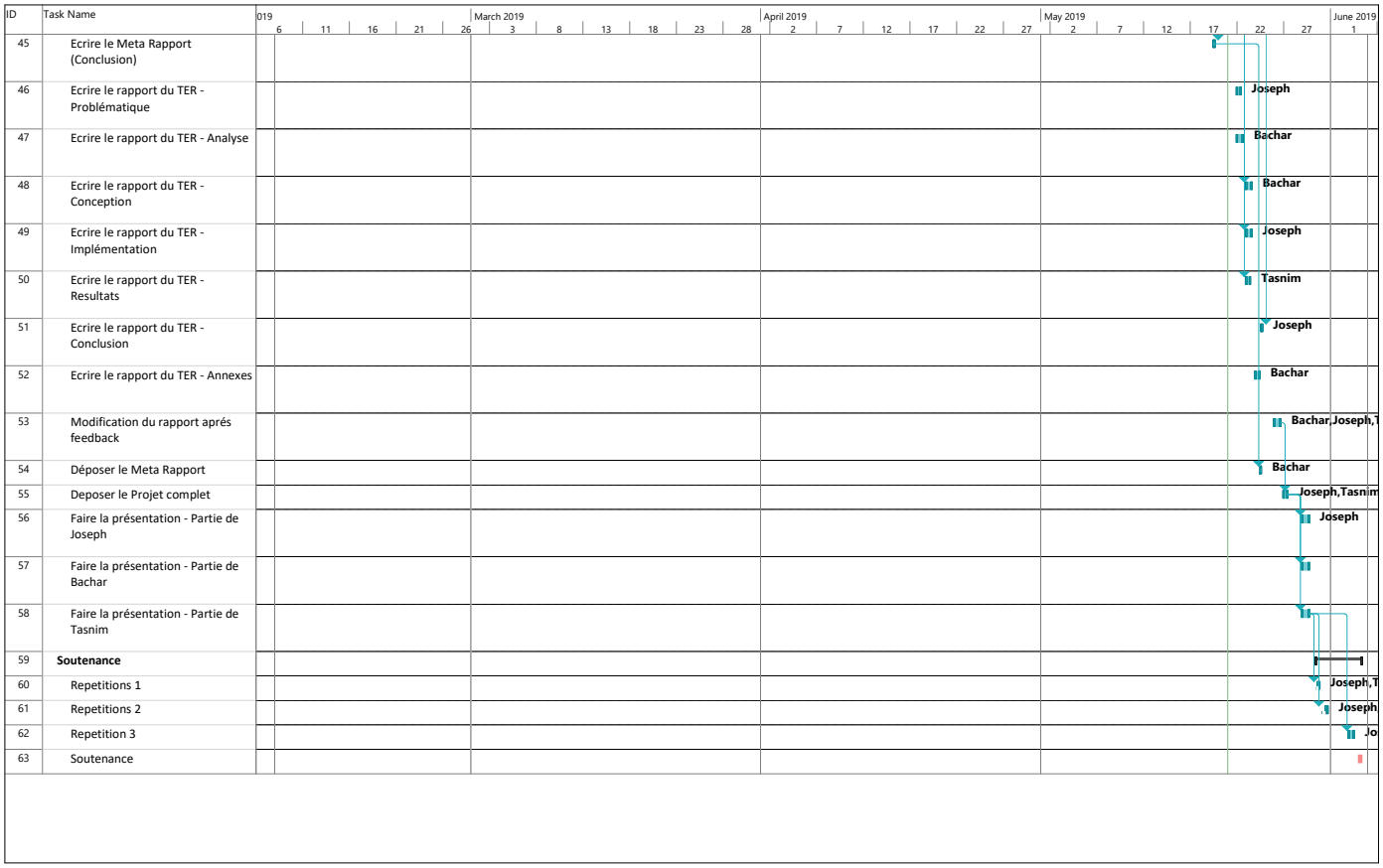


FIGURE 7.1 – Planning final

7.2 Difficultés rencontrées

Au cours de se projet, nous avons rencontrer des difficultés auxquelles nous ne nous attendions pas.

Bibliographie

1)Software Heritage : Why and How to Preserve Software Source Code - Roberto Di Cosmo, Stefano Zacchiroli 2) Identifiers for Digital Objects : the Case of Software Source Code Preservation - Roberto Di Cosmo, Morane Gruenpeter, Stefano Zacchirol