



M1 INFORMATIQUE AIGLE

**HMIN204**

CONDUITE DE PROJET

---

## Méta-Rapport du TER

---

### Groupe BAJONIM

**Bachar RIMA,**

bachar.rima@etu.umontpellier.fr

**Joseph SABA,**

joseph.saba@etu.umontpellier.fr

**Tasnim SHAQURA,**

tasnim.shaqura@etu.umontpellier.fr

*Responsable de l'UE :*

Eric BOURREAU

22 Mai 2019

# Table des matières

<b>1</b>	<b>Sujet</b>	<b>2</b>
<b>2</b>	<b>Planning</b>	<b>3</b>
2.1	Planning Prévisionel . . . . .	3
2.2	Planning Final . . . . .	4
2.3	Notes sur le planning final . . . . .	5
2.4	Comparaison des plannings . . . . .	6
<b>3</b>	<b>Données Quantitatives</b>	<b>7</b>
3.1	Commits . . . . .	7
3.2	Classes . . . . .	8
<b>4</b>	<b>Conclusion</b>	<b>9</b>

# Chapitre 1

## Sujet

Le logiciel constitue une partie très importante de nos connaissances scientifiques, culturelles, et technique. Les logiciels sont présents dans tous les aspects de notre vie quotidienne. Il est donc important d'archiver les logiciels. Des efforts ont déjà été fait pour la préservation des logiciels, tel que The Internet Archive et UNESCO Persist, mais ils se concentrent sur la préservation des exécutables. Software Heritage est un projet qui a comme but la préservation des codes source des logiciels disponibles publiquement. Les codes source sont importants parcequ'ils peuvent être facilement compris par des humains, et peuvent être facilement transformés en exécutables.

L'équipe de Software Heritage on crée une architecture qui permet de retrouver les sources codes d'un dépôt et de les placer dans l'archive. Les **Listers** sont une partie centrale à cette architecture. Ce sont des crawlers configurés pour parcourir des dépôt de code et retrouver leurs contenu. Les différents dépôts de code ont des structures bien différentes l'un de l'autre, ce qui nécessite la création d'un Lister dédié à chaque plateforme qu'on souhaite archiver. L'équipe de Software Heritage a déjà crée des Listers pour quelques dépôts populaires, tel que Github et Bitbucket, avec succès ; mais jusqu'à présent, aucune équipe externe a crée un Lister. Le dépôt que nous avons choisi est Launchpad.

Les objectifs de ce TER sont :

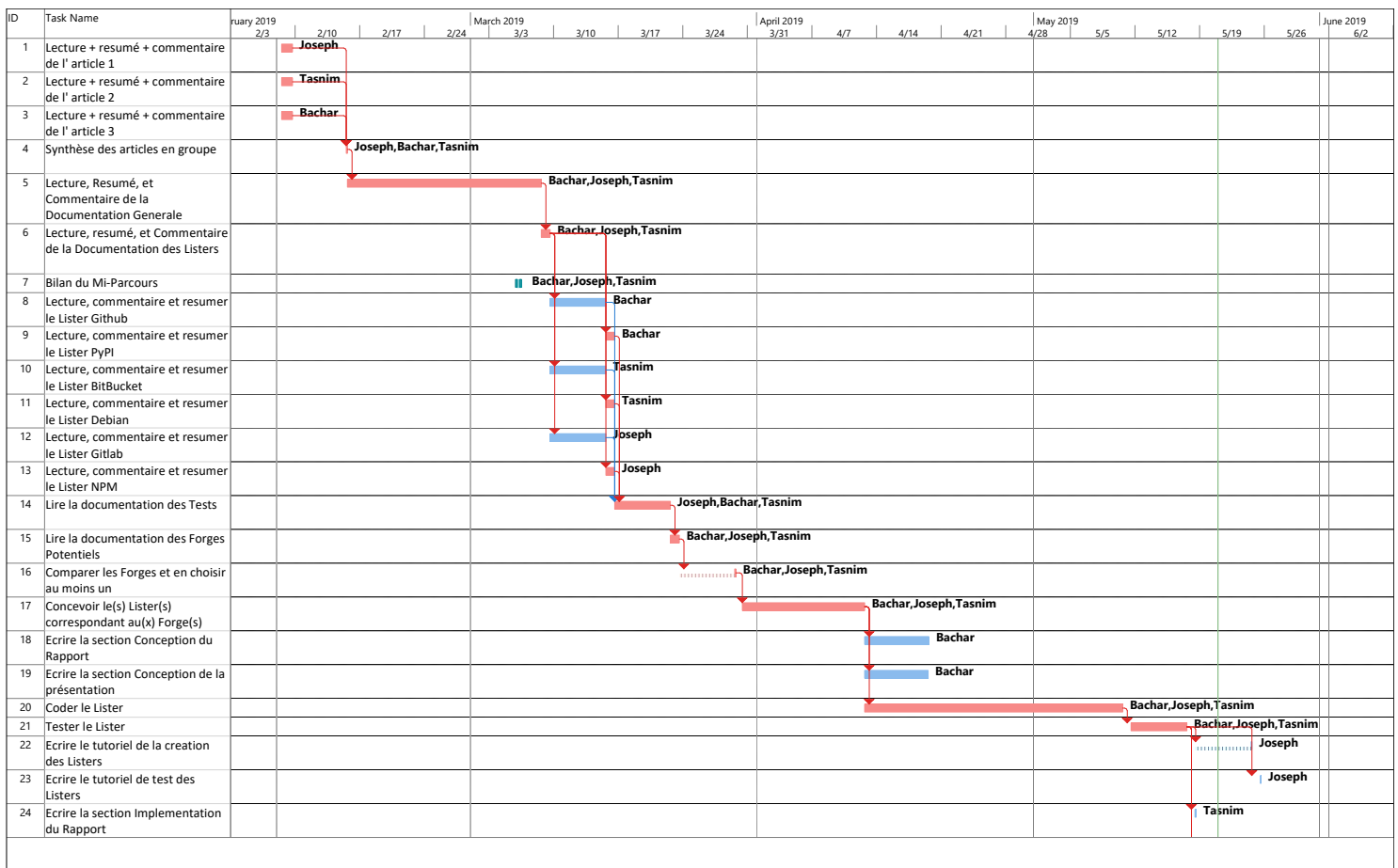
- Lire et comprendre les articles publiés par l'équipe de Software Heritage
- Lire les tutoriels écrits par l'équipe de Software Heritage
- Tester différents dépôt de code
- Écrire un Lister pour le dépôt choisis
- Répliquer localement l'environnement de Software Heritage et tester le Lister
- Faire une Pull Request pour ajouter le Lister à Software Heritage

# Chapitre 2

## Planning

Dans ce chapitre, nous montrons le diagramme de Gantt initiale et le diagramme de Gantt final, puis nous analyserons les différences entre les deux.

### 2.1 Planning Prévisionel



ID	Task Name	February 2019				March 2019				April 2019				May 2019				June 2019	
		2/3	2/10	2/17	2/24	3/3	3/10	3/17	3/24	3/31	4/7	4/14	4/21	4/28	5/5	5/12	5/19	5/26	6/2
25	Ecrire la section Implementation de la présentation																Tasnim		
26	Finaliser la critique des documentations de software heritage																Bachar,Joseph,Tasnim		
27	Finaliser les tutoriels																Joseph		
28	Finaliser le rapport																Bachar,Joseph		
29	Finaliser la présentation																Tasnim		
30	Préparer pour la soutenance																Bachar,Joseph		
31	Soutenance																Bachar,Joseph		

FIGURE 2.1 – Planning prévisionnel

## 2.2 Planning Final

ID	Task Name	019	March 2019					April 2019					May 2019					June 2019									
0	Software Heritage TER	6	11	16	21	26	3	8	13	18	23	28	2	7	12	17	22	27	2	7	12	17	22	27	1		
1	Etude préalable																										
18	APIs																										
26	Développement du Lister																										
27	Installation de l'environnement SWH	Joseph, Bachar, Tasnim																									
28	Installation de l'environnement SWH	Bachar, Joseph, Tasnim																									
29	Formalisation de l'hiérarchie des listers	Joseph, Bachar, Tasnim																									
30	Modélisation des Listers	Bachar, Joseph, Tasnim																									
31	Creation du Lister Launchpad	Joseph																									
32	Creation de la classe Proxy pour Launchpad	Bachar																									
33	Préparation des tests																										
34	Coder le Lister	Joseph																									
35	Coder la Superclass pour les Proxy Listers	Bachar																									
36	Résoudre les problèmes d'installation	Bachar, Joseph																									
37	Tester et modifier le programme	Bachar, Joseph, Tasnim																									
38	Pull Request	Joseph, Tasnim, Bachar																									
39	Documentation																										
40	Bilan de Mi-Parcours	Joseph, Tasnim, Bachar																									
41	Squelette du Rapport Final	Joseph, Tasnim																									
42	Ecrire le Meta Rapport (Sujet + Planning + Données quantitatives)	Joseph																									
43	Ecrire le Meta Rapport (Données qualitatives)	Bachar																									
44	Ecrire le rapport du TER - Introduction	Tasnim																									

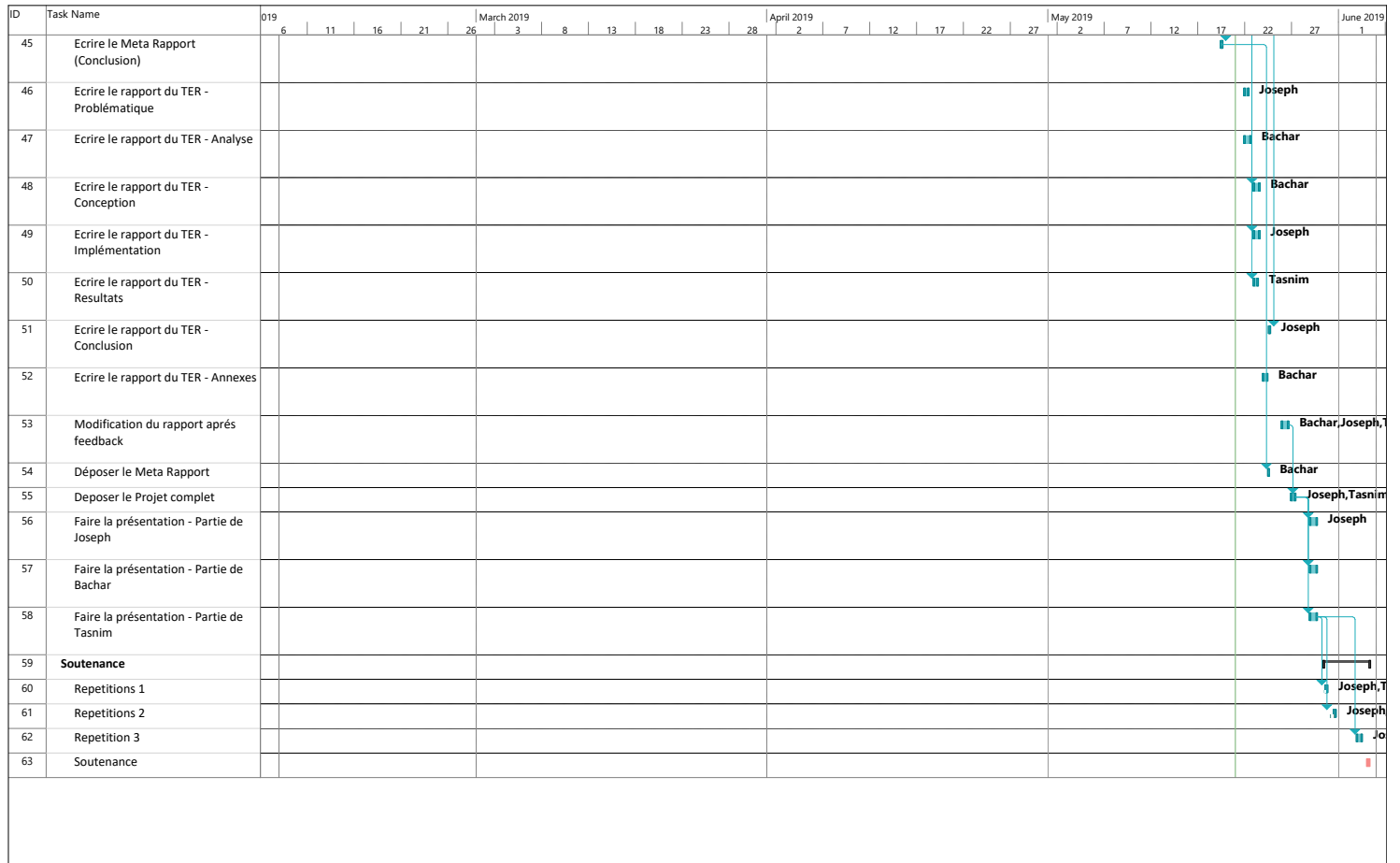


FIGURE 2.2 – Planning final

## 2.3 Notes sur le planning final

Après avoir déposer le planning prévisionnel, nous avons refait le planning et nous l'avons découpé en cinq parties pour mieux refléter la structure du projet :

- **Étude préalable** : Cette section forme une grande partie du projet. Elle inclut toutes les tâches de lecture et recherche qui nous ont permis de se familiariser avec l'architecture et l'environnement de Software Heritage, ainsi que les différents Listers préexistants.
- **APIs** : Cette section regroupe les tâches qui concernent la recherche d'un dépôt de code avec une API adaptée, qui va être exploitée dans

les tâches de développement.

- **Développement du Lister** : Ce groupe contient les tâches liées au développement du Lister, donc les tâches de modélisation et conception, de coding, et de testing.
- **Documentation** : Pour organiser la création des rapports et de la présentation.
- **Soutenance** : Pour organiser les répétitions et la soutenance.

## 2.4 Comparaison des plannings

Bachar

# Chapitre 3

## Données Quantitatives

Ce chapitre se concentre sur les détails quantitatifs concernant le TER. Le projet est disponible sur le lien Git suivant : <https://github.com/joe11093/Software-Heritage-TER>

### 3.1 Commits

Depuis la date de la création du Git repo, jusqu'au XX/XX/XXXX, nous avons effectué XX commits.

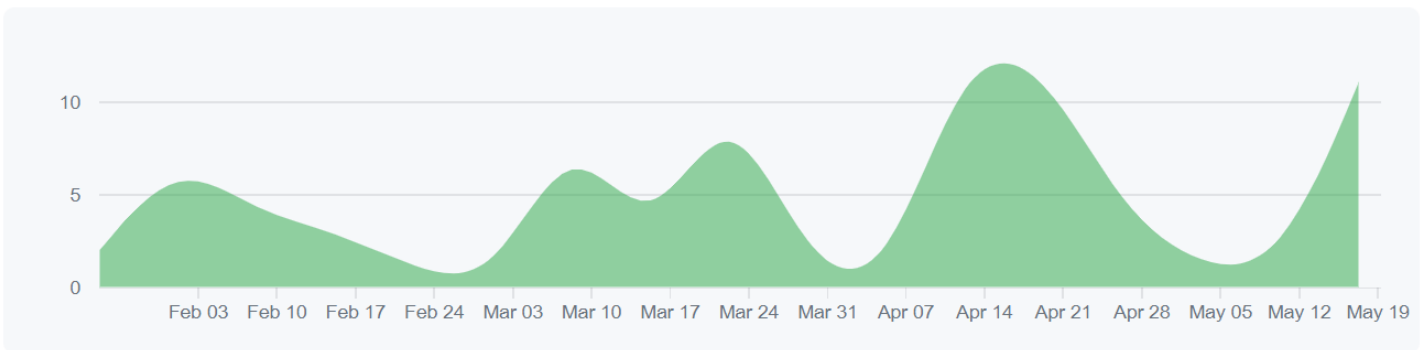


FIGURE 3.1 – Graph des commits

Nous observons qu'en premier temps, le nombre de commits est bas. Cela correspond à la période de l'étude préalable qui consistait surtout de tâches de lecture. Les deux points bas suivants (aux alentours de fin Mars et début Mai) sont dû à la période de rattrapage et la période des révisions des examens finaux. Les commits s'élèvent à la fin du TER parce que nous ajoutons au fur et à mesure le code sur lequel nous travaillons, ainsi que les rapports et leurs documents correspondants.

Nous observons que les commits ont été souvent effectués les Vendredis. En effet, nous avons décidé de travailler sur le TER les Vendredis, qui est la seule journée où les trois membres du groupe n'ont pas de cours. Les commits de la dernière période du projet sont une exception ; cela est dû à l'arrêt des cours.



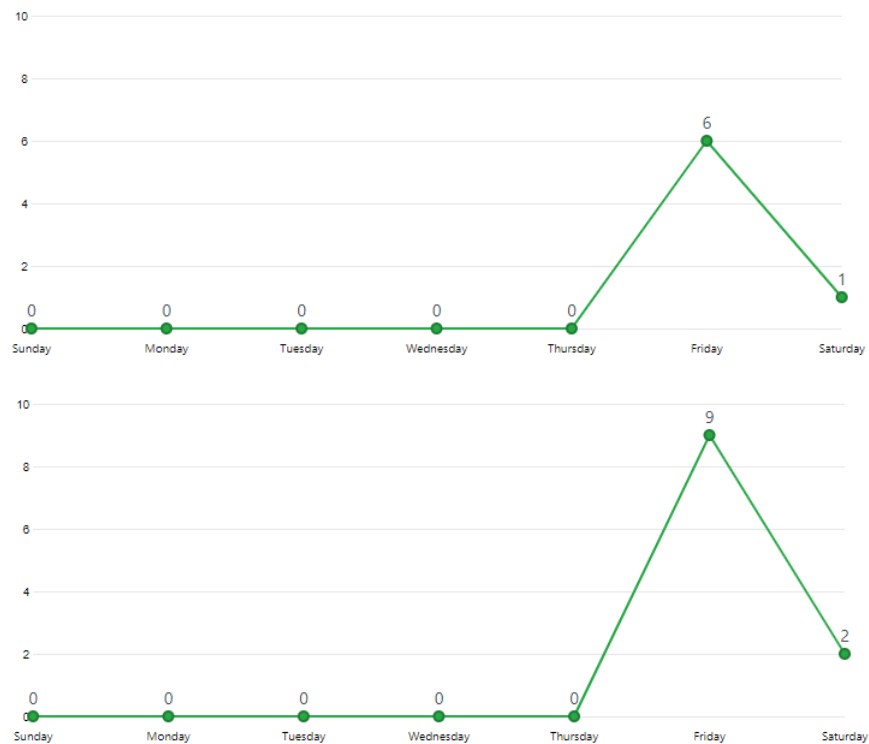


FIGURE 3.2 – Moyenne des commits par jours pour les semaines du 3 Février et du 24 Mars

## 3.2 Classes

- **LaunchpadProxy** : Cette classe exploite l'API de Launchpad pour récupérer tous les projets Git disponibles publiquement.
- **LaunchpadGitModel** : Elle définit le modèle d'un repository en Git hébergé sur Launchpad.
- **LaunchpadGitLister** : La classe du Lister qui implemente toute les fonctions requises pour énumérer les origines hébergés sur Launchpad.
- **ProxiedLister** : Classe abstraite qui pourra être implementée dans un autre Lister qui a les exigences que Launchpad.
- **WebApiProxy** : Une interface qui contient les déclarations des fonctions requise dans une classe Proxy utilisée par un Lister.
- **LaunchpadListerTester** : Une classe utilisée dans les tests

### 3.3 Temps de Travail

En ce qui concerne le temps de travail, nous estimons :

- Approximativement 110 heures sur l'étude préalable du projet.
- Approximativement 60 heures pour la recherche et les tests des APIs.
- Approximativement 110 heures pour le developpement du code
- Approximativement 20 heures pour la documentation et les rapports.
- Approximativement

Nous estimons encore autour de 30 heures de travail, qui portera surtout sur l'écriture du rapport du TER ainsi que la présentation.

Chapitre 4

Conclusion