Licence Informatique

La Rochelle Université

GALACTIC Market

Développement d'une interface en ligne de commande avec Click

Cyprien ROBINAUD

2025

Contents

Liste des figures

Remerciements

Je tiens à exprimer mes remerciements à toutes les personnes qui ont contribué au bon déroulement de mon stage.

Je remercie du fond du cœur **mes parents** et mon **grand frère** pour leur soutien constant et leur confiance tout au long de mon parcours.

Je remercie sincèrement Monsieur **Christophe DEMKO**, mon maître de stage, pour sa bienveillance, son encadrement et sa grande disponibilité tout au long de mon stage. Je lui suis particulièrement reconnaissant de m'avoir offert cette opportunité d'intégrer son équipe et de m'avoir accueilli avec confiance. Les formations qu'il m'a dispensées, ses conseils avisés ainsi que les nombreux apprentissages que j'ai pu acquérir à ses côtés ont été précieux.

Je remercie également Monsieur **Renaud PETERI**, mon tuteur pédagogique, d'avoir accepté cette mission, puis d'avoir suivi et veillé au bon déroulement de mon stage.

Je souhaite aussi remercier Madame **Karell BERTET**, ainsi que de nouveau Monsieur Christophe DEMKO, pour la présentation claire et structurée du projet GALACTIC lors de la réunion à laquelle j'ai eu le privilège d'assister. Cette présentation m'a permis de mieux comprendre les objectifs du projet et ma contribution au sein de celui-ci.

Je tiens à remercier tout particulièrement Madame **Djénaba BARRY**, ma collègue de travail, pour sa bonne humeur quotidienne, sa disponibilité et son aide précieuse. Sa présence a grandement facilité l'avancement du projet et a contribué à rendre cette expérience d'autant plus agréable.

Enfin, ma reconnaissance s'adresse également à Monsieur **Yacine GHAMRI-DOUDANE**, directeur du L3i, pour m'avoir accueilli au sein de son établissement, ainsi qu'à **l'ensemble de l'équipe du L3i**, pour leur accueil chaleureux.

Abstract

Résumé en français

Le projet **GALACTIC *Market*** s'inscrit dans le cadre du développement d'un écosystème numérique autour du cadriciel **GALACTIC**, un outil d'analyse de données basé sur l'**Analyse Formelle des Concepts**. Ce projet, conduit au sein du **L3i** de La Rochelle Université, vise à concevoir une plateforme de gestion de paquets Python sous forme de proxy respectant la spécification **PEP503** (Simple Repository API).

L'objectif est de créer un **marché virtuel de paquets *Python*** spécifiques au cadriciel **GALACTIC**, facilitant la consultation, la recherche et l'interaction avec des paquets hébergés sur divers dépôts.

Le projet se structure en deux axes : un ***backend* commun** et des **interfaces**. Le *backend* permet d'interagir avec des dépôts *Python* compatibles, de traiter les métadonnées des paquets, de gérer leur indexation dans un cache. L'ensemble des métadonnées collectées permet une recherche via des classifiers ou autres critères, offrant une expérience utilisateur plus fluide et pertinente.

Une première contribution spécifique concerne le développement d'une **interface en ligne de commande (CLI)** à l'aide de la bibliothèque **Click**, dont les fonctionnalités incluent la navigation dans le marché, la recherche de paquets, la consultation des métadonnées, le téléchargement de paquets, ainsi que la gestion de dépôts locaux avec ou sans authentification. Une attention particulière est portée à l'ergonomie de la **CLI**, aux options d'affichage, et à la robustesse du code via des tests unitaires. Ce travail s'accompagne d'une **documentation** utilisateur complète.

Ce projet s'inscrit dans une démarche pédagogique forte : il permet d'approfondir les compétences en développement *Python*, en gestion de **projets collaboratifs**, en structuration d'*API*, et en gestion de versions avec *GitLab*.

GALACTIC, en tant que cadriciel transparent et interactif, favorise un **numérique responsable** et est adapté à des contextes variés, comme en témoigne son implication dans le projet **DA3T** portant sur la valorisation touristique à La Rochelle. Ainsi, le **GALACTIC Market** représente une extension technologique pratique, visant à offrir des outils de gestion de données ouverts, flexibles et puissants a ces utilisateurs.

Abstract in English

The **GALACTIC *Market*** project is part of the development of a digital ecosystem around the **GALACTIC** framework, a data analysis tool based on **Formal Concept Analysis**. This project, carried out within the **L3i** laboratory at La Rochelle University, aims to design a Python package management platform in the form of a proxy that complies with the **PEP503** (Simple Repository API) specification.

The goal is to create a **virtual marketplace for *Python* packages** specific to the **GALACTIC** framework, making it easier to browse, search, and interact with packages hosted on various repositories.

The project is structured around two main components: a **shared *backend*** and **interfaces**. The backend handles interaction with compatible Python repositories, processes package metadata, and manages their indexing in a cache. The collected metadata allows for advanced searches using classifiers or other criteria, offering users a smoother and more relevant experience.

An initial specific contribution focuses on developing a **Command Line Interface (CLI)** using the **Click** library. Its features include browsing the marketplace, searching for packages, viewing metadata, downloading packages, and managing local repositories with or without authentication. Particular attention is given to the CLI's usability, display options, and code robustness through unit testing. This work is supported by comprehensive **user documentation**.

This project aligns with a strong educational approach: it deepens skills in *Python* development, **collaborative project management**, *API* design, and version control using *GitLab*.

GALACTIC, as a transparent and interactive framework, promotes **responsible digital practices** and is suitable for various contexts, as demonstrated by its role in the **DA3T** project focused on enhancing tourism in La Rochelle. In this way, the **GALACTIC Market** stands as a practical technological extension, aimed at providing open, flexible, and powerful data management tools to its users.

1 Introduction

1.1 Le Laboratoire Informatique, Image et Interaction

Le **L**aboratoire **I**nformatique, **I**mage et **I**nteraction (**L3i**) est un centre de recherche de **La Rochelle Université**. Ses recherches se distinguent en mettant l'accent sur les interactions humaines. (Université, s. d.)

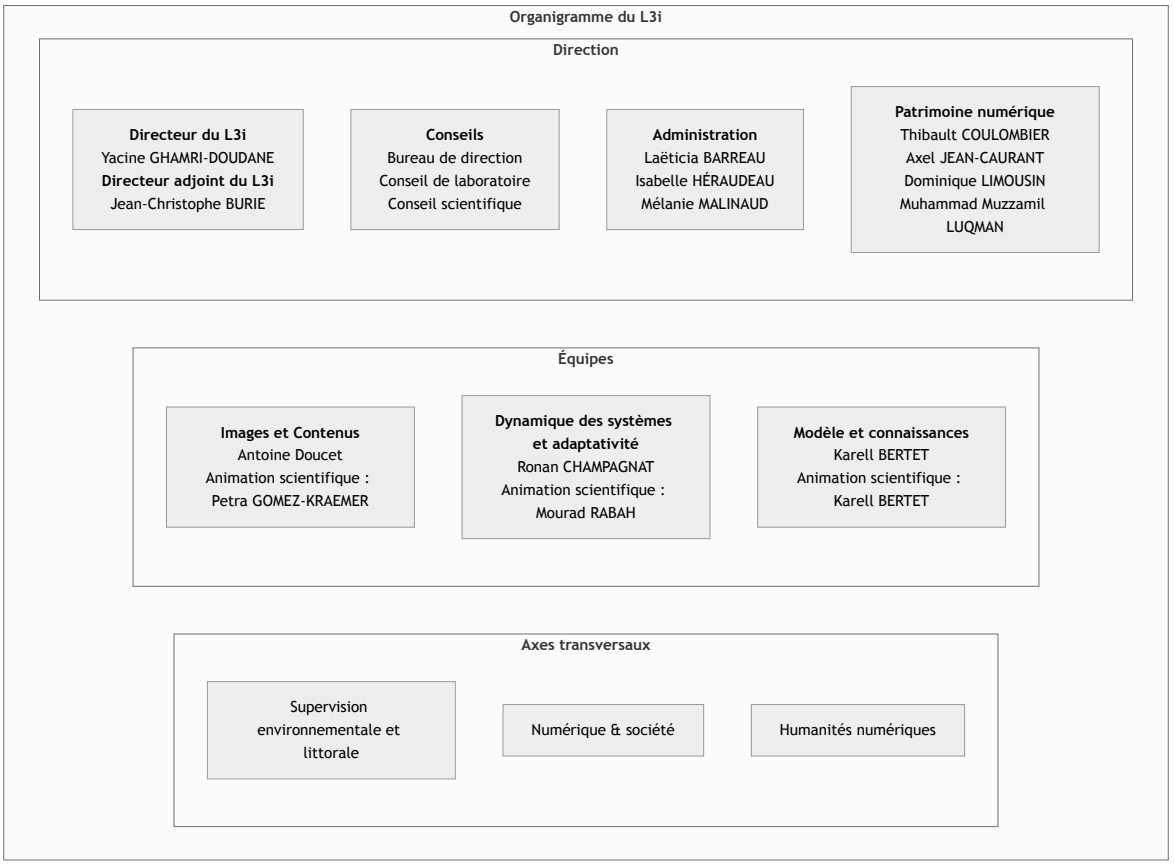
- **1993** : création du « Laboratoire Informatique et Imagerie Industrielle »;
- **1997** : labellisation en Équipe d'Accueil (Ministère de la Recherche);
- **2003** : changement de nom en « Laboratoire Informatique, Image et Interaction »;
- **2007** : labellisation en Équipe de Recherche en Technologie « Interactivité Numérique »;
- **2008** : intégration dans le programme régional **PRIDES**;
- **2011** : obtention de la note A suite à l'évaluation de l'**A**gence d'**E**valuation de la **R**echerche et de l'**E**nseignement **S**upérieur (**AERES**).

Figure 1: Logo de **La Rochelle Université**

Figure 2: Logo du **L3i**

1.1.1 Organigramme

Durant mon stage au **L3i**, je suis affecté à l'équipe **Modèle et connaissances**. Le présent organigramme est en date de février 2024. Le **L3i** est composé d'environ **120 membres**.



1.1.2 Équipes de recherche

Le **L3i** mène ses travaux autour de trois axes principaux organisés en équipes :

- **M**odèles et **C**onnaissances (**MC**) : Travaux sur les modèles formels, la représentation des connaissances, les données spatio-temporelles et les systèmes de raisonnement. L'équipe participe à des projets comme I-Trace et **GALACTIC**.
- **I**mages et **C**ontenus (**IC**) : Recherche sur le traitement, l'analyse et la structuration de contenus multimédias faiblement structurés (images, vidéos, textes). Elle s'intéresse à l'extraction, l'indexation et la recherche d'informations.
- Dynamique des Systèmes et Adaptativité (**eAdapt**) : Étude des systèmes logiciels adaptatifs, notamment ceux qui réagissent au comportement de l'utilisateur. Travaille sur des architectures logicielles dynamiques et le pilotage du traitement en fonction du contexte.

1.1.3 Partenariats

Depuis sa création, le **L3i** collabore avec de nombreux partenaires **académiques**, **industriels** et **institutionnels**.

Il est à l'origine du **consortium** *Valconum* pour la valorisation des recherches numériques.

Le laboratoire entretient des partenariats **internationaux** avec des institutions comme le *Computer Vision Center* (Barcelone), l'*Université de Hanoï* et l'*Université de Sfax*, ainsi qu'avec des laboratoires **nationaux** tels que le *GIPSA-lab* et l'*INRAE*.

Il travaille également avec plusieurs **entreprises**, comme *AM Créations*, *Arkhênum* et *Shift Technology*, ainsi qu'avec des **institutions publiques** telles que la *Région Poitou-Charentes* et la *Communauté d'Agglomération de La Rochelle*.

1.2 Le cadriciel GALACTIC

GAlois **LA**ttices, **C**oncept **T**heory, **I**mplicational systems and **C**losures (**GALACTIC**) Demko et al. (2024) est un projet de recherche qui a débuté en janvier 2004. Les référents du projet sont **Karell BERTET** (directrice de recherche) et **Christophe DEMKO** (architecte logiciel). (Organization 2025)

GALACTIC se distingue par son approche « boîte blanche », c'est-à-dire qu'elle reste **transparente** sur les procédés appliqués aux données. Elle permet ainsi aux analystes de données d'explorer de manière interactive et progressive les données.

L'**interactivité** permet à l'analyste de cibler les stratégies adaptées à son contexte et ainsi de limiter les traitements sans intérêt et coûteux en ressources, favorisant ainsi le **numérique responsable**.

Étymologie
Le projet **GALACTIC** s'appuie sur les travaux du mathématicien français Évariste GALOIS, décédé en 1832 lors d'un duel. Son nom a inspiré celui du projet.

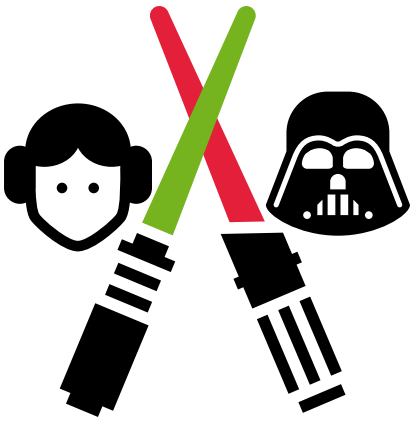


Figure 3: Logo du projet **GALACTIC**

1.2.1 Architecture

Le noyau de **GALACTIC** est structuré en plusieurs couches sur lesquelles peuvent venir se greffer des extensions :

- **les caractéristiques**, qui permettent d'extraire tout type de valeur à partir des données;
- **les descriptions**, qui définissent un ensemble de valeurs (et donc de données) à travers les frontières d'une enveloppe convexe généralisée;
- **les stratégies**, qui explorent un ensemble de données en se concentrant sur des groupes répondant à certains prédicats monadiques;
- **les mesures**, qui alimentent des méta-stratégies en informations quantitatives;
- **les lecteurs de données**, qui permettent d'importer des fichiers de tout format pour constituer des ensembles de données.

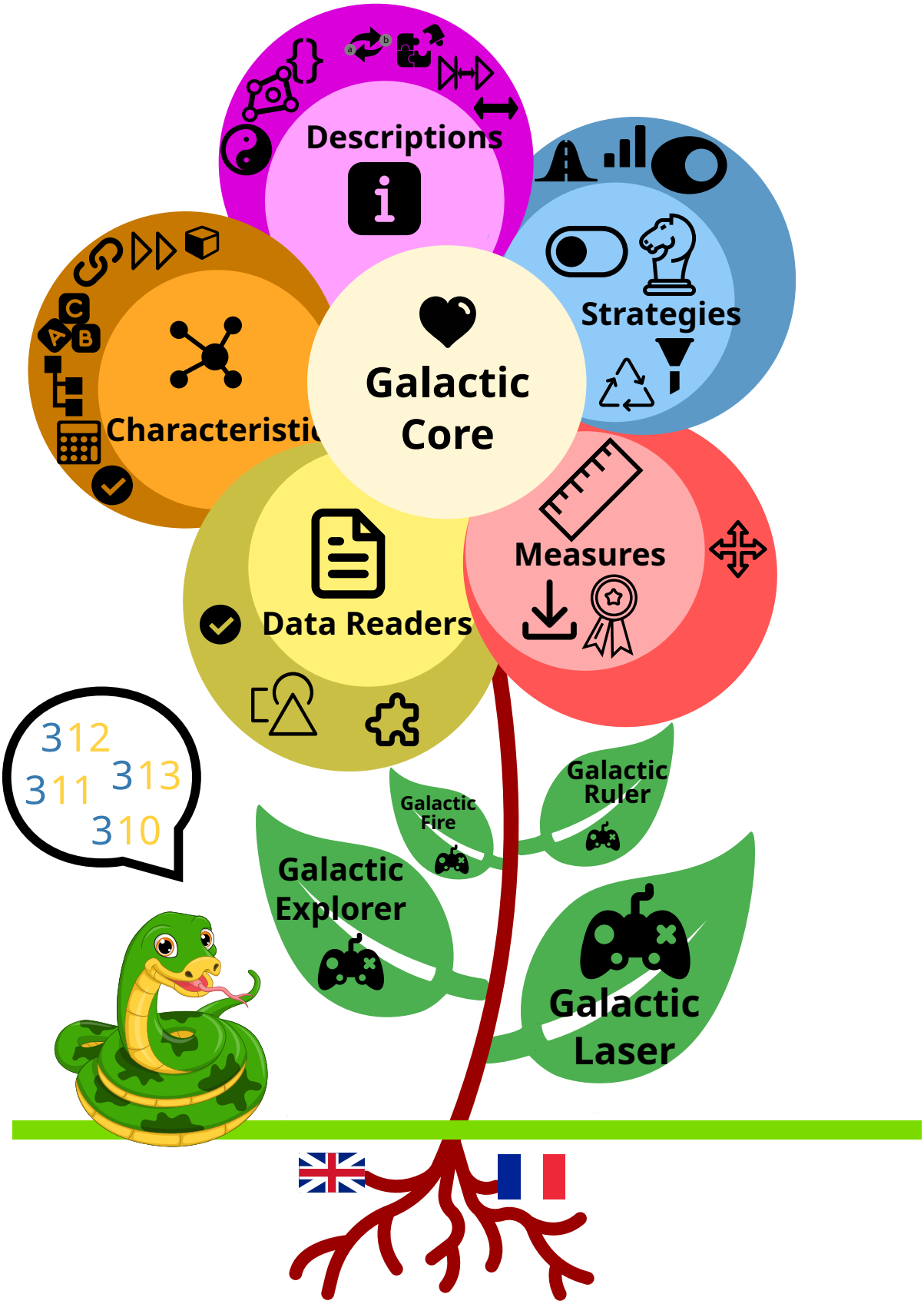


Figure 4: Architecture de **GALACTIC**

1.2.2 Fondements scientifiques

GALACTIC est basé sur l'**A**nalyse **F**ormelle des **C**oncepts (**AFC**) parue en 1982. Elle organise les données en sous-groupes à partir de descriptions communes et exploite les propriétés algébriques des **treillis** et des **opérateurs de fermeture**.

En 2020, deux verrous scientifiques ont été levés : l'**analyse de données complexes et hétérogènes** et le **déluge des motifs descriptifs de sous-groupes**.

Exemple
Dans le cadre du **Dispositif d'Analyse des Traces numériques pour la valorisation des Territoires Touristiques** (**DA3T**), des touristes à La Rochelle ont été tracés (avec leur accord, via une application smartphone) pour étudier leurs mouvements. Les localisations spatiales et temporelles sont des données hétérogènes et complexes. **GALACTIC** permet de traiter ces données pour en fournir une **lecture plus explicite**.

1.2.3 Partenariats

GALACTIC est financé par le *réseau SATT* ainsi que le **L3i**. Il compte de nombreux partenaires, par exemple l'*OIP Atlantique*.

1.3 Le GALACTIC *Market*

Le projet consiste à développer des applications de gestion et de consultation d'un **GALACTIC *Market***, un marché virtuel de paquets `Python` pour le cadriciel **GALACTIC** respectant la spécification **Simple Repository API**. Ces paquets ne contiennent pas de code, mais agissent comme des proxys pour d'autres paquets hébergés sur divers dépôts compatibles, éventuellement accessibles via authentification.

Chaque paquet du **GALACTIC *Market*** est enrichi par des **attributs `Python`** spécifiques, facilitant la recherche et la catégorisation. Des métadonnées supplémentaires comme des icônes, des liens vers la documentation ou des pages web permettent d'enrichir l'expérience utilisateur, notamment pour les interfaces graphiques.

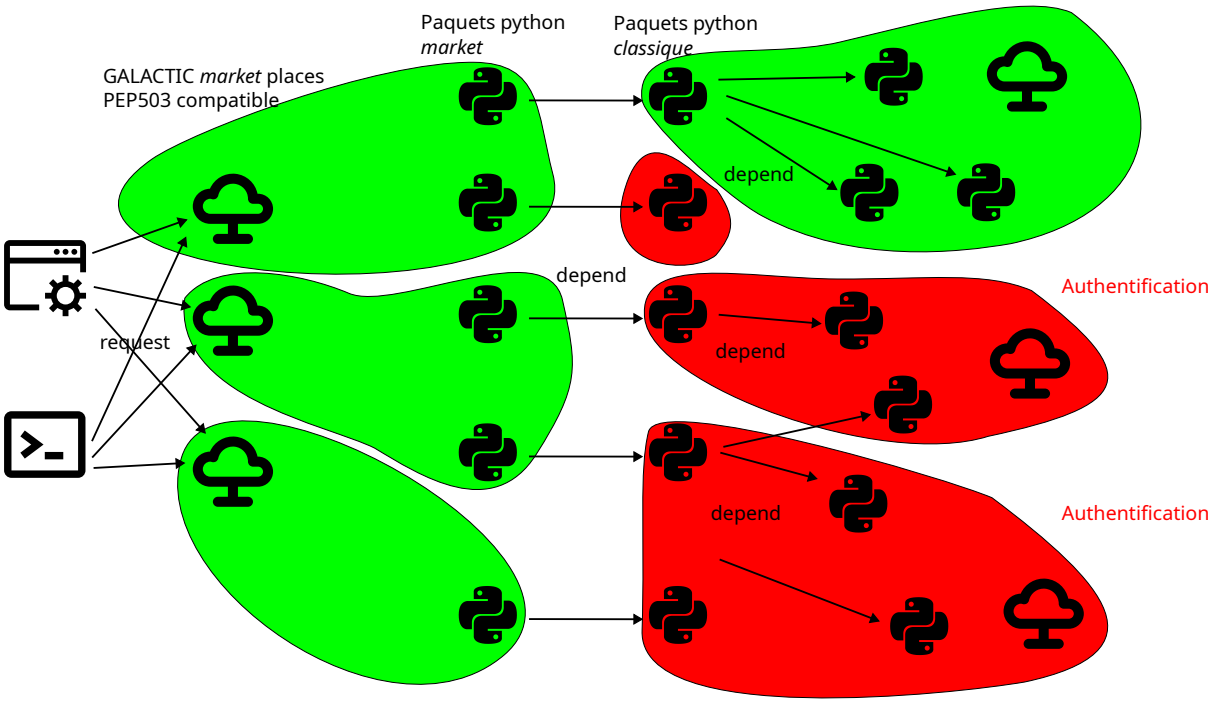


Figure 5: Architecture du **GALACTIC *Market***

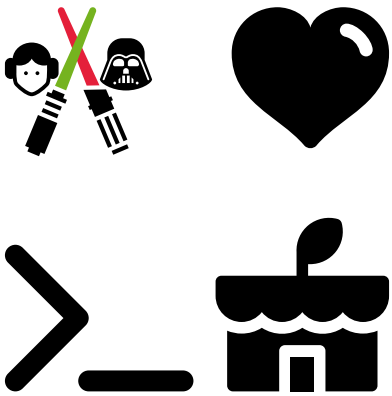


Figure 6: Logo du **GALACTIC *Market CLI***

1.3.1 *Backend* : Bibliothèque `Python`

Le *backend* est une bibliothèque `Python` destinée à interagir avec des dépôts conformes au **PEP 503**. La bibliothèque est ensuite utilisée par une **I**nterface en **L**igne de **C**ommande (**CLI**, en anglais) et une **I**nterface **U**tilisateur **G**raphique (**GUI**, en anglais).

Elle implémente les fonctionnalités suivantes :

- gérer la récupération, l'analyse et l'indexation des métadonnées des paquets (attributs, liens spécifiques, etc.);
- implémenter une gestion des *proxys* pour rediriger les requêtes vers d'autres dépôts;
- ajouter une authentification pour accéder aux dépôts protégés;
- stocker les métadonnées et informations des paquets récupérés;
- supporter des recherches avancées par attributs ou autres critères.

1.3.2 *Frontend* : Interface en ligne de commande

Le livrable est une **CLI**, intuitive et ergonomique.

Elle implémente les fonctionnalités suivantes :

- lister, rechercher, afficher, télécharger les paquets disponibles dans le **GALACTIC *Market***;
- ajouter et gérer des dépôts conformes au **PEP 503**, y compris avec authentification;
- ajouter des options pour personnaliser l'affichage (tri par attributs et affichage détaillé).

1.3.3 Objectifs du stage

- approfondir les compétences en développement `Python`, incluant des bibliothèques spécialisées;
- comprendre et implémenter des spécifications techniques;
- apprendre à structurer un projet collaboratif avec une partie commune et des contributions spécifiques;
- développer des compétences en gestion des métadonnées et des **API** `Python`;
- apprendre à utiliser les outils de création de paquets `Python`;
- approfondir les mécanismes de gestion de versions logicielles.

2 Technologies et outils utilisés

Le projet **GALACTIC** s'appuie sur un ensemble d'outils qui couvrent tout le **cycle de vie du logiciel** : construction, dépendances, qualité, tests, documentation et versionnage. Ces outils ont été sélectionnés pour leur efficacité, leur intégration avec les environnements de développement, et leur utilisation au sein de la communauté `Python`.

Le **diagramme de séquence** suivant présente les actions réalisées après une mise à jour (*push*) sur `GitLab`. Par la suite, `GitLab` utilise `Docker` afin de publier le paquet standard sur le serveur `PyPI` standard* ainsi que le paquet *docs* (si existant). Le paquet *market* (si existant) est quant à lui envoyé sur le serveur `PyPI` *market*.

Un serveur Python Package Index (PyPI)
C'est un **dépôt en ligne** où sont hébergés des **paquets Python**. Ces paquets contiennent du code réutilisable (bibliothèques, outils, cadriciel, etc.).

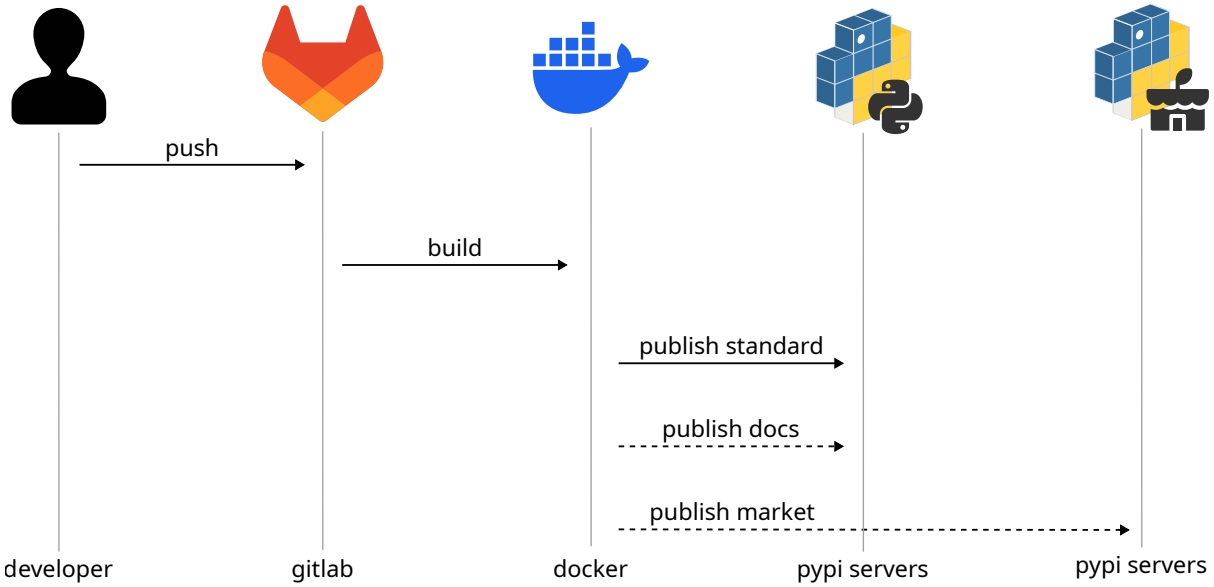


Figure 7: Publication des paquets **GALACTIC**

2.1 Gestion, automatisation et empaquetage

`Docker` **Plateforme de virtualisation** utilisant des conteneurs pour empaqueter une application avec toutes ses dépendances, afin d'assurer son exécution sur différents environnements. (« Docker » 2025)

`GitLab` **Plateforme de gestion de code** source basée sur `Git`, intégrant le suivi des problèmes, les requêtes de fusions, la documentation, et les intégrations/déploiement continue (**CI/CD**). Elle centralise les processus de **développement collaboratif**. (« GitLab » 2025)

`Hatch` Outil central de gestion des **paquets**, des **environnements virtuels** et des **scripts de développement**, basé sur un fichier `pyproject.toml`. Il unifie plusieurs fonctionnalités dans une interface moderne, rapide et conforme aux standards de la communauté Python. (« Hatch » 2025)

`uv` Gestionnaire de dépendances rapide, basé sur `Rust`, utilisé avec `Hatch` pour améliorer la **résolution des paquets** dans les environnements virtuels. (« UV » 2025)

`Pre-commit` Cadriciel pour exécuter automatiquement de nombreux outils de validation (`Black`, `Ruff`, `Mypy`, etc.) **avant chaque commit**, garantissant la qualité du code dès sa soumission. (« Pre-commit » 2025)

`Towncrier` Génère automatiquement les **notes de version** (`changelog`) à partir de fichiers de modifications, facilitant un suivi clair de l'évolution du projet. (« Towncrier » 2025)

2.2 Qualité du code et vérifications statiques

`Black` Formateur automatique de code `Python` selon les standards **PEP 8**. Il assure un style uniforme, avec une intégration facile dans *Hatch* ou *VS Code*, et une **politique de stabilité** rigoureuse. (Langa et Black 2025)

`Ruff` Linter extrêmement rapide, combinant les **règles de plusieurs outils** classiques (`Flake8`, `isort`, etc.) pour une analyse efficace du code `Python`. (« Ruff » 2025)

`Mypy` Analyseur statique de types. Utilisé en mode strict dans **GALACTIC** pour détecter les **erreurs de typage** à l'avance et améliorer la robustesse du code. (« Mypy » 2025)

`Pylint` Linter complet qui vérifie le style de code, les bonnes pratiques et, associé à `slotscheck`, l'**optimisation mémoire**. (« Pylint » 2025)

`Refurb` détecte les **mauvaises pratiques** et les constructions obsolètes. (« Refurb » 2025)

`Teyit` **stabilité des tests** et signale les usages obsolètes. (« Teyit » 2025)

2.3 Tests, couverture et documentation

`Pytest` Cadriciel principal pour les **tests unitaires**, avec une syntaxe simple, des fixtures et de nombreux plugins. (« Pytest » 2025)

`Coverage` Outil mesurant la **couverture des tests**, avec génération de rapports **HTML** ou **XML**, pour évaluer la qualité de test du code source. (« Coverage.py » 2025)

`Sphinx` Générateur de documentation technique à partir des **docstrings** `Python`, produisant des formats comme **HTML** ou **PDF**. Référence dans l'écosystème `Python`. (« Sphinx » 2025)

3 Conception et développement

3.1 *Backend* : Bibliothèque Python

3.1.1 Gestion du cache

J'ai eu besoin de stocker localement des fichiers. Pour respecter les bonnes pratiques de développement multiplateforme et garantir un stockage dans un emplacement standardisé selon le système d'exploitation, j'ai choisi de placer ces fichiers dans le **cache de l'utilisateur**.

Plusieurs choix étaient possibles parmi eux :

`platformdirs` est une bibliothèque `Python` qui fournit des fonctions pour récupérer les chemins standards utilisés par les applications pour stocker les fichiers de configuration, de cache et de données utilisateur, en tenant compte des spécificités des systèmes d'exploitation.

`appdirs` est une bibliothèque `Python` permettant d'obtenir les répertoires standards pour les données d'application, la configuration et le cache, selon la plateforme utilisée. Elle est aujourd'hui obsolète et remplacée par `platformdirs`.

`os` est un module standard de `Python` qui fournit des fonctions pour interagir avec le système de fichiers, notamment la manipulation des chemins.

Outil	Facilité	Installé	Portabilité	Version	Date
platformdirs	✓		✓	4.3.8	**2025**
appdirs	✓		✓	1.4.4	2020
os		✓		N/A	N/A

Critères de comparaison

Facilité Le code est-il est inférieur ou égal à 5 lignes. **Installé** L'outil est-il inclus dans la bibliothèque standard `Python`.
Version Numéro de version publié sur **Py**thon **P**ackage **I**ndex PyPI
Portabilité L'outil s'adapte t'il automatiquement selon le système d'exploitation.
Date Dernière mise à jour d'après PyPI et la Documentation Python

Le choix s'est orienté vers `platformdirs`. En effet, `appdirs` n'a pas été mis à jour depuis le 11 mai 2020, contrairement à `platformdirs` qui lui a été mis à jour le 8 mai 2025. En considérant l'évolution des systèmes d'exploitation, il existe un sérieux risque de **maintenabilité**. Ensuite `os`, bien que ce soit une bibliothèque standard et qu'elle soit fiable, elle ne propose pas d'**abstraction** pour chaque chemin selon le système d'exploitation. En ce sens, cela rend la bibliothèque bien plus complexe à utiliser, et moins adaptée à notre cas d'utilisation.

En résumé, le choix de `platformdirs` s'est imposé naturellement pour ses atouts en termes de **portabilité**, **simplicité**, et **respect des conventions** de chaque système d'exploitation, assurant une gestion cohérente du cache utilisateur sans compromettre la maintenabilité du code.

3.1.2 Interrogation des métadonnées

Suite à l'interrogation du dépôt (partie réalisée par ma collègue Djénaba BARRY), on récupère des fichiers *wheels* (extension .whl), la **PEP 427** définit un *wheel* comme un format binaire standardisé pour la distribution de paquets Python.

La Python Enhancement Proposal (PEP)
Il s'agit d'un document qui propose des améliorations pour le langage Python. Elle décrit de nouvelles fonctionnalités, des processus ou des normes. Chaque **PEP** est examinée par la communauté, et certaines sont adoptées pour guider l'évolution de Python.

C'est une archive au format *ZIP*, celle-ci contient un fichier *METADATA* conforme aux spécifications définies par le **Py**thon **P**ackaging **A**uthority (**PyPA**). Le fichier *METADATA* est donc appelé à évoluer avec le temps.

Il existe une multitude d'outils permettant d'extraire les métadonnées d'un *wheel*. J'ai étudié cinq outils pour déterminer lequel est le plus adapté pour notre utilisation :

- `distlib.wheel`
- `importlib.metadata` (abrégé *metadata*)
- `pkginfo`
- `wheel.wheelfile` (abrégé *wheelfile*)
- `zipfile`

Outil	Facilité	Installé	Version	Date	Téléch.	Mémoire	Vitesse
distlib	✓		0.3.9	09/09/24	135k	88kB (±0,4kB)	1,8ms (±0,4ms)
metadata	✓	✓	N/A	**27/04/25**	N/A	**27kB (±16,3kB)**	1,1ms (±0,3ms)
pkginfo	✓		1.12.1	19/02/25	68k	87kB (±0.0kB)	1,2ms (±0,3ms)
wheelfile			0.45.1	23/10/24	**261k**	89kB (±0.2kB)	0,6ms (±0,2ms)
zipfile		✓	N/A	08/04/25	N/A	86kB (±0.0kB)	**0,4ms (±0,1ms)**

Données manquantes
Les bibliothèques standard de Python n'ont pas de version indiqué, ni de téléchargement, car leur numéro de version et le nombre de téléchargements dépend directement de l'interpréteur Python.

Critères de comparaison

Facilité Le code est-il est inférieur ou égal à 3 lignes. **Installé** L'outil est-il inclus dans la bibliothèque standard Python.
Version Numéro de version publié sur **Py**thon **P**ackage **I**ndex PyPI

Date Dernière mise à jour d'après PyPI et la Documentation Python
Téléch. Nombre de téléchargement lors d'avril 2025, d'après PyPI Stats
Mémoire Mémoire consommée et écart-type, sur 10 000 lectures d'attributs.
Vitesse Vitesse exécution et écart-type, sur 10 0000 lectures d'attributs.

Parmi les outils disponibles, `pkginfo` se distingue par son bon équilibre entre **simplicité**, **performances** et **maintenabilité**. Il a été mis à jour récemment (19/02/2025), ce qui garantit un certain niveau de fiabilité. Ses performances sont bonnes avec une vitesse de 1,2 ms et une consommation mémoire de 87 kB.

En comparaison, `importlib.metadata` , bien que léger et rapide, ce dernier nécessite **l'installation du paquet** avant l'analyse de ces métadonnées, cela s'avère une opération coûteuse en mémoire, et en temps. `wheel.wheelfile` et `zipfile` sont plus rapides, mais **moins adaptés** : le premier n'est pas simple à utiliser; le second, bien qu'installé par défaut, n'est pas spécialisé pour les métadonnées. Enfin, `distlib` est globalement correct, mais **moins performant** et la dernière mise à jour commence à dater.

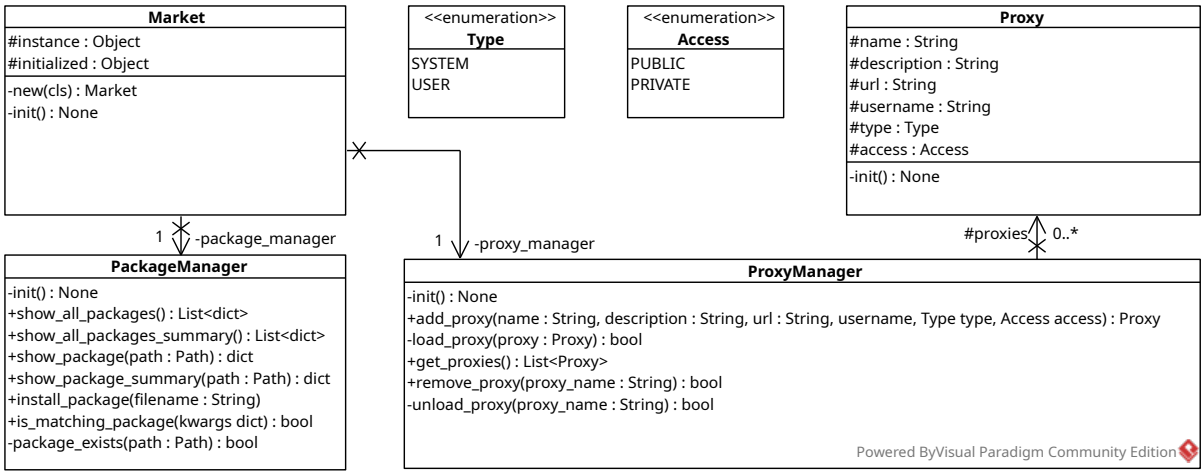
En conclusion, `pkginfo` s'impose comme l'option la plus cohérente pour manipuler les métadonnées de paquets `Python` dans un cadre **fiable** et **durable**.

3.1.3 Développement d'une bibliothèque

Le diagramme de classe (conçu sur *Visual Paradigm*) présenté permet de visualiser et conceptualiser le fonctionnement de la bibliothèque. Il permet de représenter clairement les relations entre les différentes classes et leurs responsabilités.

Par exemple, la classe *Market* centralise la gestion en détenant une instance de *PackageManager*, pour les paquets, et de *ProxyManager*, pour les proxys. Ce dernier peut contenir de zéro à plusieurs objets Proxy, chacun possédant divers attributs.

Certains proxys peuvent inclure un token ou un userPassword si un accès authentifié est requis (comme pour des serveurs privés ou premium).

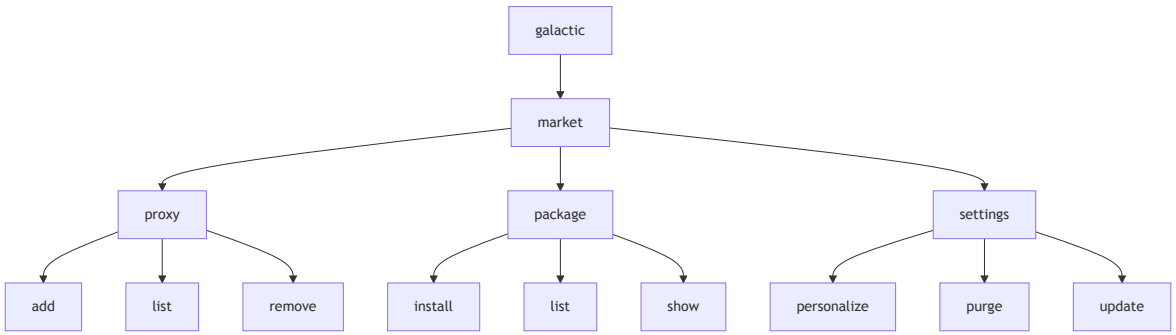


3.2 *Frontend* : Interface en ligne de commande

La **CLI** a pour objectif de fournir une expérience utilisateur fluide et intuitive, pour permettre l'exploration et la gestion des paquets disponibles au sein du **GALACTIC *Market***, ou plus généralement envers les dépôts conformes **PEP 503**.

3.2.1 Arborescence des commandes

Une arborescence cohérente des commandes a été effectuée. Le graphe ci-dessus présente les commandes disponibles au sein du **GALACTIC *Market***.



3.2.2 Interaction utilisateur : `Click`

La **CLI** est développée à l'aide de la bibliothèque `Click`, qui permet de : * Structurer les commandes de manière claire et hiérarchique. * Gérer les paramètres (options, arguments) de commande de manière intuitive. * Générer automatiquement l'aide en ligne (–help).

Chaque commande propose un retour immédiat sur la réussite ou l'échec de l'opération.

3.2.3 Mise en forme enrichie : `Rich`

L'affichage de la **CLI** est enrichi grâce à la bibliothèque `Rich` qui permet :

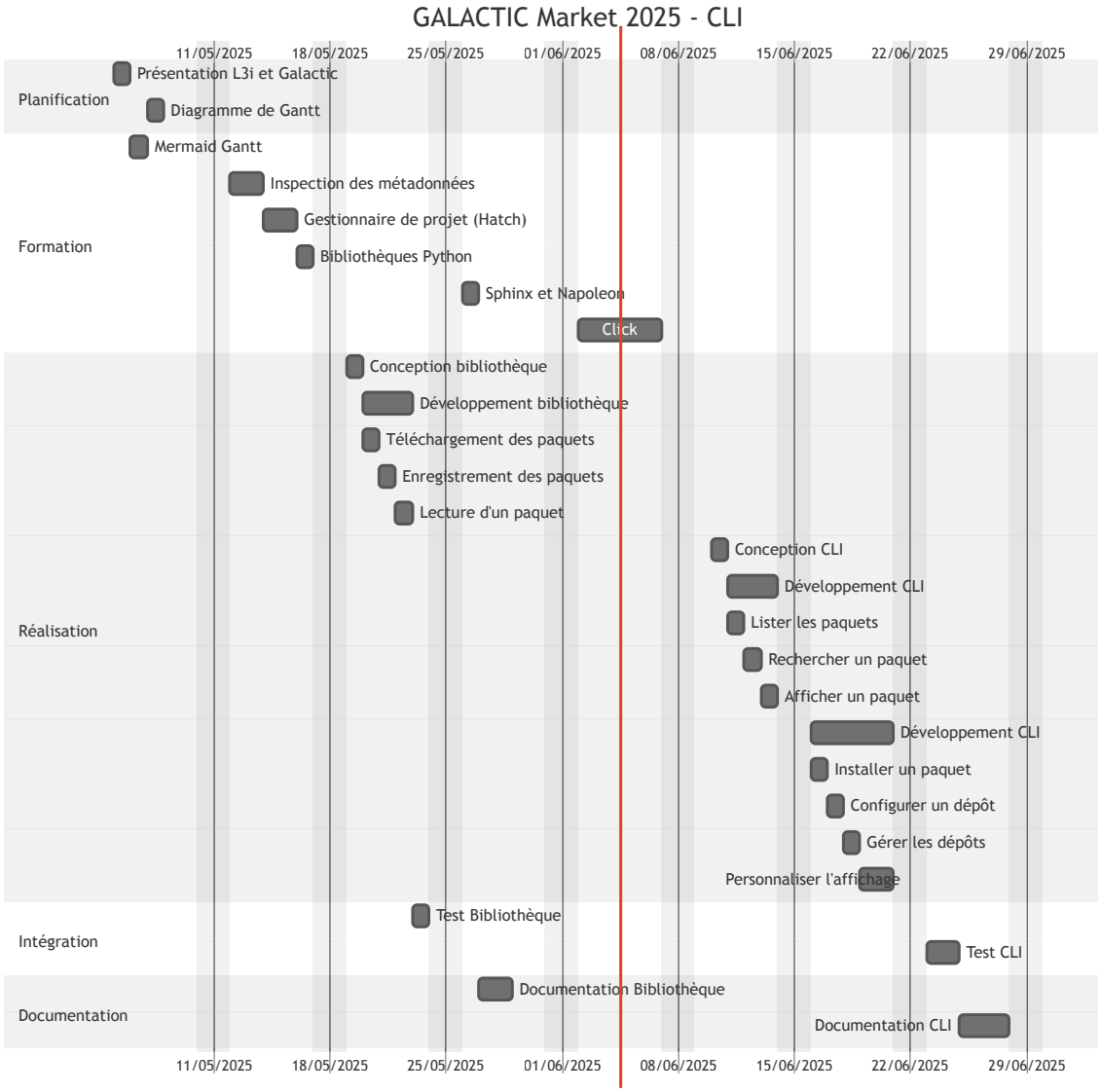
- L'utilisation de tables colorées, pour lister les paquets ou les proxys.
- L'affichage de panneaux encadrés pour présenter les métadonnées d'un paquet.
- Une hiérarchisation visuelle claire des titres, des avertissements, et des erreurs.
- Le support du markdown dans l'aide intégrée et les messages d'état.
- Des barres de progression dynamiques lors du téléchargements des paquets.

Cela améliore considérablement l'expérience utilisateur, en rendant la **CLI** à la fois plus lisible et agréable à utiliser. De plus une attention particulière a été portée aux couleurs qui sont utilisées, pour se rapprocher le plus de la charte graphique du logo **GALACTIC**.

4 Analyse des résultats

L'analyse des résultats repose sur un planning prévisionnel / diagramme de *Gantt*, construit dès le lancement du projet à l'aide de l'outil `Mermaid Gantt`, qui a servi de **fil conducteur** pour répartir les tâches, coordonner les efforts de l'équipe, et garantir le bon déroulement du développement.

4.1 Planning prévisionnel



4.2 Résultats obtenus

Durant le début du projet un planning prévisionnel avait été fait, mais dès le troisième jour il a été **revu complètement** car il ne correspondait pas à l'avancée du projet. En effet, nous avions prévu une période de **formation trop longue**, alors qu'en réalité la formation s'est réalisée **en parallèle de la conception**, ce qui nous a permis de mieux saisir les enjeux.

À ce stade du projet, on a une **bibliothèque fonctionnelle**. Néanmoins quelques fonctionnalités restent à implémenter notamment l'authentification, mais nous avons déjà une idée de comment faire donc cela ne devrait poser aucun un problème. Les **tests sont actuellement rédigés** pour la bibliothèque.

Ensuite la **CLI** a été commencée, ainsi que les **travaux de recherche et la conception** sur cette dernière. Donc le **planning prévisionnel est plutôt suivi**. Même si c'est à nuancer avec un retard sur la bibliothèque et une avance sur la **CLI**.

4.3 Perspectives et travaux à venir

Les prochaines étapes consisteront à **finaliser le développement de la bibliothèque** `Python`, dont l'architecture a déjà été entièrement définie à l'aide d'un diagramme de classes réalisé avec *Visual Paradigm*. Une fois cette phase terminée, l'accent sera mis sur la **poursuite du développement de la CLI**, qui devra offrir une interaction fluide et intuitive avec la bibliothèque.

L'objectif est de garantir une solution à la fois **fonctionnelle, maintenable et ergonomique pour l'utilisateur final**. À ce jour, le diagramme de *Gantt* et le planning prévisionnel n'ont pas nécessité de nouvelle révision, et il est prévu qu'ils soient suivis jusqu'à la fin du projet.

5 Bilan Personnel

Au cours de ce stage, j'ai pu vivre une **expérience enrichissante** sur plusieurs plans, que ce soit au niveau du travail en équipe, de l'utilisation d'outils professionnels ou encore de la conception logicielle. Voici un retour structuré sur ces différents aspects.

5.1 Point de vue collaboratif

Dans le domaine du développement logiciel, la collaboration est omniprésente. La grande majorité des projets sont réalisés en groupe, ce qui implique une **organisation rigoureuse** et l'usage d'outils collaboratifs, tels que `GitLab` pour la gestion de versions, ou *Microsoft Teams* pour la communication, aussi bien en présentiel qu'à distance.

Dans le cadre du projet **GALACTIC** Market, des réunions d'équipe étaient organisées tous les deux jours, durant environ deux heures. Elles avaient pour but de faire le **point sur l'avancement**, de trancher sur les choix technologiques, mais aussi de partager des connaissances pour faire progresser tout le groupe.

Ce mode de fonctionnement impose également certaines **bonnes pratiques**. Par exemple, il était recommandé de ne pas effectuer de push dans `GitLab` durant la dernière heure de travail, afin d'éviter de compromettre la stabilité du projet en cas de bug de dernière minute, ce qui risquerait de bloquer l'équipe le lendemain.

5.2 Appropriation des outils professionnels

Ce stage a été pour moi l'occasion de découvrir et d'apprendre à maîtriser de nombreux outils. Au départ, certains me semblaient secondaires, voire superflus. Mais au fil du projet, j'ai compris leur rôle et leur complémentarité dans un **processus de développement** bien structuré.

Un exemple marquant est l'utilisation de Hatch, qui a transformé ma manière de travailler. Avant chaque commit, je devais m'assurer localement que tous les tests passaient. Cela impliquait plusieurs étapes : la vérification du style de code, l'analyse statique, puis l'exécution des tests unitaires. Cette **rigueur** m'a permis de mieux comprendre l'importance de la qualité du code et de la **validation continue**.

Autre évolution importante : la rédaction de **documentation technique**. Pour chaque fonctionnalité développée, une documentation complète et normée devait être produite. Cette exigence m'a appris à **formaliser mon travail** de manière professionnelle et à anticiper les besoins futurs des développeurs qui reprendront le projet.

Enfin, j'ai dû adapter mes habitudes en matière de rédaction. Habitué à utiliser *Google Docs*, j'ai dû apprendre à rédiger ce rapport en *Markdown*, comme demandé par mon maître de stage. Si ce changement a été un peu déstabilisant au début, il m'a permis de gagner en efficacité et en rigueur. Grâce à `Pandoc`, le rapport final est bien structuré et facilement **exportable** dans différents formats.

5.3 Conception et développement

Sur le plan technique, j'ai pris conscience que le cœur du métier de développeur ne réside pas uniquement dans l'écriture de code, mais surtout dans la **conception réfléchie** de solutions adaptées aux besoins du client.

Trouver la bonne solution, c'est d'abord comprendre le **problème, analyser les contraintes, comparer les technologies disponibles, et faire des choix éclairés**. Ce travail de réflexion prend souvent davantage de temps que le développement lui-même, mais il est indispensable pour garantir un logiciel robuste, maintenable et évolutif.

Concevoir une **architecture logicielle efficace, anticiper les futurs besoins, garantir la lisibilité et la modularité du code** : ce sont autant de défis que j'ai dû relever, et qui m'ont permis de progresser considérablement dans ma compréhension du métier.

6 Conclusion

Ce stage au sein du **L3i** m'a offert une immersion riche et formatrice dans le monde de la recherche appliquée au développement logiciel. En participant au projet **GALACTIC *Market***, j'ai eu l'opportunité d'explorer en profondeur les aspects de conception, d'architecture logicielle et de développement outillé, tout en travaillant au sein d'une équipe projet structurée et collaborative.

Les résultats obtenus à ce jour sont significatifs. Malgré une **révision anticipée du planning** dès les premiers jours, l'avancée du projet est globalement conforme aux objectifs initiaux. La bibliothèque `Python` est aujourd'hui fonctionnelle, avec une **architecture solide** et des tests unitaires rédigés. Les fondations de la **CLI** ont également été posées, avec une organisation claire des commandes et une attention particulière portée à l'ergonomie et à l'expérience utilisateur.

Les perspectives sont tout aussi prometteuses. Les prochaines étapes permettront d'aboutir à une solution complète, intégrant une **CLI** fluide et une bibliothèque robuste, **conforme aux standards modernes de l'empaquettage** `Python`. Ce projet contribuera ainsi à l'enrichissement de l'écosystème du cadriciel **GALACTIC**, en facilitant la distribution et l'exploration de modules spécialisés.

Sur le plan personnel, ce stage m'a permis de consolider des **compétences techniques** essentielles en gestion de versions, documentation, tests, industrialisation, tout en développant une approche plus rigoureuse de la **conception logicielle**. J'ai également renforcé ma capacité à **travailler en équipe**, à collaborer efficacement et à prendre des **décisions techniques argumentées**.

En somme, cette expérience constitue une étape significative dans mon parcours. Elle a renforcé ma motivation à poursuivre dans le domaine du développement logiciel, avec une **conscience plus fine des exigences et des responsabilités** qu'implique la création de solutions techniques durables et utiles.

Références

« Coverage.py ». 2025. https://coverage.readthedocs.io/.

Demko, Christophe, Karell Bertet, Cyril Faucher, Jean-François Viaud, et Sergei O. Kuznetsov. 2020. « NextPriorityConcept: A new and generic algorithm computing concepts from complex and heterogeneous data ». *Theoretical Computer Science* 845 (décembre): 1-20.

Demko, Christophe, Karell Bertet, Jean-François Viaud, Cyril Faucher, et Damien Mondou. 2024. « Description lattices of generalised convex hulls ». *International Journal of Approximate Reasoning*, août, 109269.

« Docker ». 2025. https://www.docker.com/.

« GitLab ». 2025. https://docs.gitlab.com/.

« Hatch ». 2025. https://hatch.pypa.io/.

Langa, Łukasz, et contributors to Black. 2025. « Black ». https://black.readthedocs.io/.

« Mypy ». 2025. https://mypy-lang.org/.

Organization, The Galactic. 2025. « GAlois LAttices, Concept Theory, Implicational systems and Closures ». https://l3i.univ-larochelle.fr/.

« Pre-commit ». 2025. https://pre-commit.com/.

« Pylint ». 2025. https://pylint.readthedocs.io/.

« Pytest ». 2025. https://docs.pytest.org/.

« Refurb ». 2025. https://github.com/dosisod/refurb.

« Ruff ». 2025. https://docs.astral.sh/ruff/.

« Sphinx ». 2025. https://www.sphinx-doc.org/.

« Teyit ». 2025. https://github.com/isidentical/teyit.

« Towncrier ». 2025. https://towncrier.readthedocs.io/.

Université, La Rochelle. s. d. « L3i - Laboratoire d'Informatique, Image et Interaction ». https://l3i.univ-larochelle.fr/.

« UV ». 2025. https://docs.astral.sh/uv/.