

Plan

- 1 Introduction
- 2 Solutions existantes
- 3 Architecture
- 4 Technologies
- 5 Réalisation
- 6 Discussion
- 7 Conclusion

h e p i a

Haute école du paysage, d'ingénierie
et d'architecture de Genève

- Nombre de fichiers énorme.

Problématiques

- Nombre de fichiers énorme.
- Difficulté à retrouver des fichiers.

Problématiques

- Nombre de fichiers énorme.
- Difficulté à retrouver des fichiers.
- Plusieurs emplacements logiques pour un seul fichier.

Problématiques

- Nombre de fichiers énorme.
- Difficulté à retrouver des fichiers.
- Plusieurs emplacements logiques pour un seul fichier.

Système de "tagging" de fichiers et répertoires avec possibilité de recherche par tags.

Cahier des charges

- Répertorier les applications existantes permettant d'étiqueter les fichiers.

Cahier des charges

- Répertorier les applications existantes permettant d'étiqueter les fichiers.
- Étudier une façon de stocker les tags avec les fichiers : les attributs étendus des fichiers (XATTR).

Cahier des charges

- Répertorier les applications existantes permettant d'étiqueter les fichiers.
- Étudier une façon de stocker les tags avec les fichiers : les attributs étendus des fichiers (XATTR).
- Analyser les moyens d'indexer et de surveiller une arborescence de fichiers.

Cahier des charges

- Répertorier les applications existantes permettant d'étiqueter les fichiers.
- Étudier une façon de stocker les tags avec les fichiers : les attributs étendus des fichiers (XATTR).
- Analyser les moyens d'indexer et de surveiller une arborescence de fichiers.
- Concevoir et implémenter le système (open source et sur Linux) et mesurer ses performances.

Cahier des charges

- Répertorier les applications existantes permettant d'étiqueter les fichiers.
- Étudier une façon de stocker les tags avec les fichiers : les attributs étendus des fichiers (XATTR).
- Analyser les moyens d'indexer et de surveiller une arborescence de fichiers.
- Concevoir et implémenter le système (open source et sur Linux) et mesurer ses performances.
- Étudier et s'appropriier le langage Rust.

TMSU, Tagsistant et TagSpaces

- Gestion des tags.
- Liste de fichiers liés aux tags.
- CLI ou GUI.

TMSU, Tagsistant et TagSpaces

- Gestion des tags.
- Liste de fichiers liés aux tags.
- CLI ou GUI.

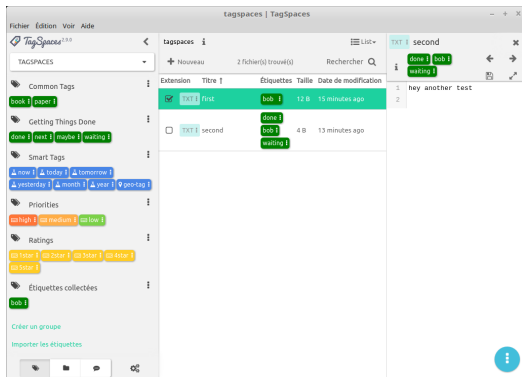


Figure – Utilisation de TagSpaces

TMSU, Tagsistant et TagSpaces

- Gestion des tags.
- Liste de fichiers liés aux tags.
- CLI ou GUI.

Points positifs	Points négatifs
Simple	Dépendance à une BDD externe
Rapides et efficaces	Modification et accès uniquement par l'app
Open source	

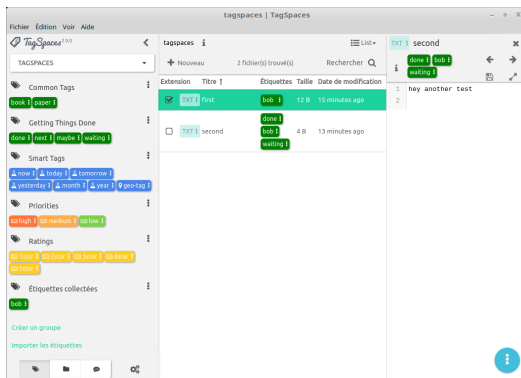


Figure – Utilisation de TagSpaces

macOS

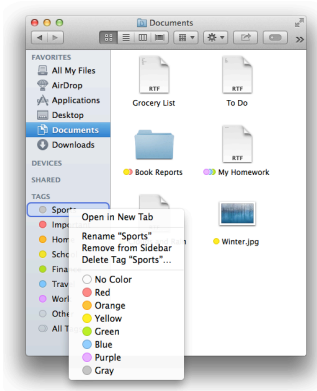
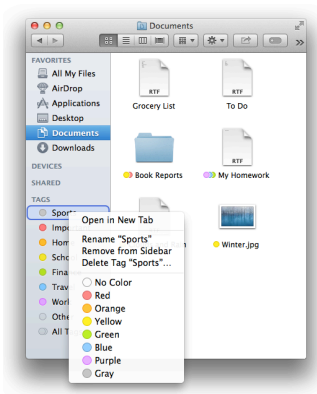


Figure – Gestion d'un tag dans le Finder - Apple

h e p i a

Haute école du paysage, d'ingénierie
et d'architecture de Genève

macOS



Points positifs	Points négatifs
Système de tags intégré à l'explorateur de fichiers	Code propriétaire
Stocke les tags dans les XATTR	Seulement pour macOS
Performant	

Figure – Gestion d'un tag dans le Finder - Apple

h e p i a

Haute école du paysage, d'ingénierie
et d'architecture de Genève

Gestion des tags

Gestion des tags

- Stockage des tags avec les fichiers (indépendance à une base de données).

Gestion des tags

- Stockage des tags avec les fichiers (indépendance à une base de données).
- Outil dédié de gestion de tags (confort d'utilisation).

Indexation des fichiers et des tags

Indexation des fichiers et des tags

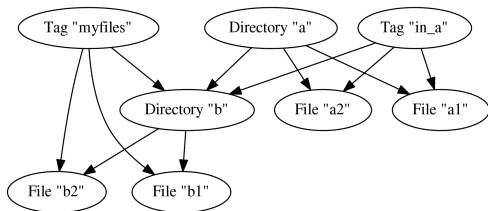


Figure – Graphe de tags, fichiers et répertoires

Indexation des fichiers et des tags

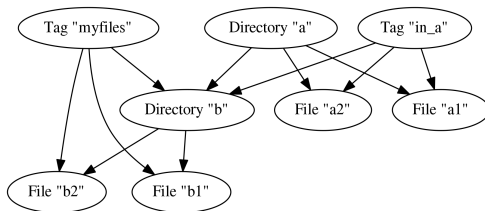


Figure – Graphe de tags, fichiers et répertoires

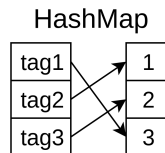


Figure – HashMap associant le nom du tag à son noeud

Surveillance du système de fichiers

Mise à jour du graphe lors des événements suivants :

Surveillance du système de fichiers

Mise à jour du graphe lors des événements suivants :

- Changement sur les tags.

Surveillance du système de fichiers

Mise à jour du graphe lors des événements suivants :

- Changement sur les tags.
- Création de fichiers/répertoires.

Surveillance du système de fichiers

Mise à jour du graphe lors des événements suivants :

- Changement sur les tags.
- Création de fichiers/répertoires.
- Suppression de fichiers/répertoires.

Surveillance du système de fichiers

Mise à jour du graphe lors des événements suivants :

- Changement sur les tags.
- Création de fichiers/répertoires.
- Suppression de fichiers/répertoires.
- Déplacement/renommage de fichiers/répertoires.

Requêtes de tags et fichiers

Requêtes de tags et fichiers

- Lister les fichiers et répertoires associés à des tags => requêtes sous forme d'expressions logiques (opérateurs logiques).

Requêtes de tags et fichiers

- Lister les fichiers et répertoires associés à des tags => requêtes sous forme d'expressions logiques (opérateurs logiques).
- Lister les tags existants.

Requêtes de tags et fichiers

- Lister les fichiers et répertoires associés à des tags => requêtes sous forme d'expressions logiques (opérateurs logiques).
- Lister les tags existants.
- Renommer un tag.

Rust

- Langage système moderne, performant, fiable (plus sécurisé qu'Ada), compilé, et fortement typé.

Rust

- Langage système moderne, performant, fiable (plus sécurisé qu'Ada), compilé, et fortement typé.
- Disponible sur Linux, Windows et macOS.

Rust

- Langage système moderne, performant, fiable (plus sécurisé qu'Ada), compilé, et fortement typé.
- Disponible sur Linux, Windows et macOS.
- Cargo : outil de compilation et d'exécution et gestionnaire de paquets intégré à Rust.

Rust

- Langage système moderne, performant, fiable (plus sécurisé qu'Ada), compilé, et fortement typé.
- Disponible sur Linux, Windows et macOS.
- Cargo : outil de compilation et d'exécution et gestionnaire de paquets intégré à Rust.
- Structures, collections, généricité, immutabilité, énumérations et *pattern matching*.

Rust

- Langage système moderne, performant, fiable (plus sécurisé qu'Ada), compilé, et fortement typé.
- Disponible sur Linux, Windows et macOS.
- Cargo : outil de compilation et d'exécution et gestionnaire de paquets intégré à Rust.
- Structures, collections, généricité, immutabilité, énumérations et *pattern matching*.
- Gestion des erreurs et tests unitaires.

Rust

- Langage système moderne, performant, fiable (plus sécurisé qu'Ada), compilé, et fortement typé.
- Disponible sur Linux, Windows et macOS.
- Cargo : outil de compilation et d'exécution et gestionnaire de paquets intégré à Rust.
- Structures, collections, généricité, immutabilité, énumérations et *pattern matching*.
- Gestion des erreurs et tests unitaires.
- ***Ownership, Borrowing*** (références).

Attributs étendus (XATTR)

- Métadonnées sous forme de paires nom: valeur.

Attributs étendus (XATTR)

- Métadonnées sous forme de paires `nom: valeur`.
- Nom = chaîne de caractères, valeur = chaîne de caractères ou données binaires.

Attributs étendus (XATTR)

- Métadonnées sous forme de paires `nom: valeur`.
- Nom = chaîne de caractères, valeur = chaîne de caractères ou données binaires.
- Existent sous ext2-3-4, XFS, Btrfs, UFS1-2, NTFS, HFS+, ZFS.

Attributs étendus (XATTR)

- Métadonnées sous forme de paires `nom: valeur`.
- Nom = chaîne de caractères, valeur = chaîne de caractères ou données binaires.
- Existent sous ext2-3-4, XFS, Btrfs, UFS1-2, NTFS, HFS+, ZFS.
- Outils CLI pour facilement les manipuler.

Inotify

- API de notifications d'événements sur le système de fichiers.

Inotify

- API de notifications d'événements sur le système de fichiers.
- Trois appels système : initialisation, ajout de surveillance sur un chemin de fichiers donné et suppression de cette surveillance.

Inotify

- API de notifications d'événements sur le système de fichiers.
- Trois appels système : initialisation, ajout de surveillance sur un chemin de fichiers donné et suppression de cette surveillance.
- Lecture d'un événement avec `read()`.

Tag Manager

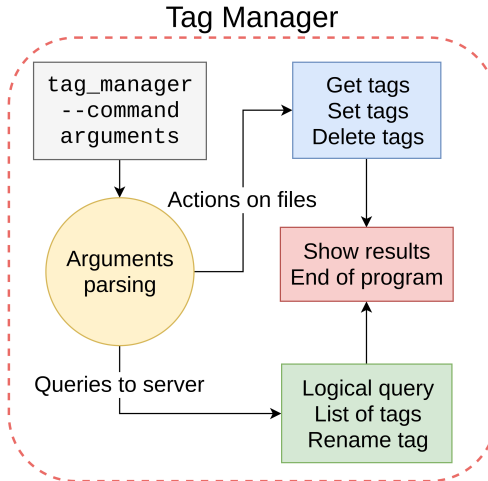


Figure – Fonctionnement de Tag Manager

Tag Engine

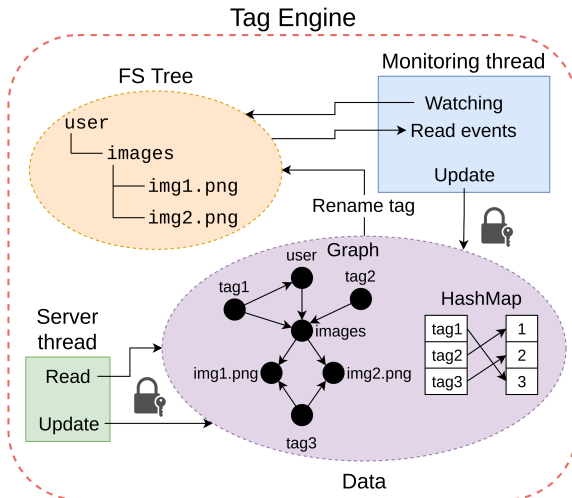


Figure – Fonctionnement de Tag Engine

TagFS

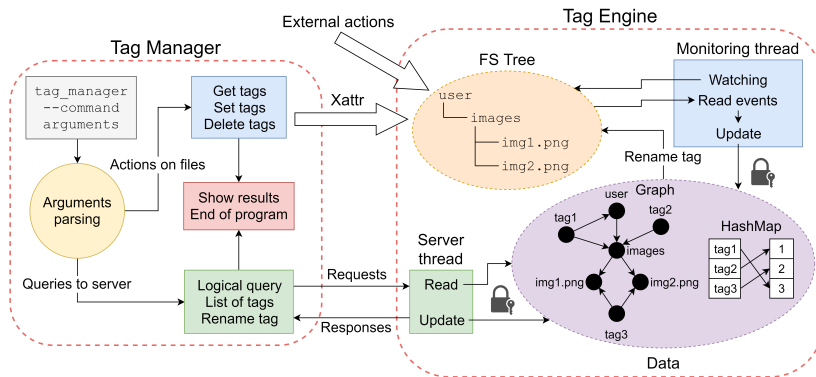
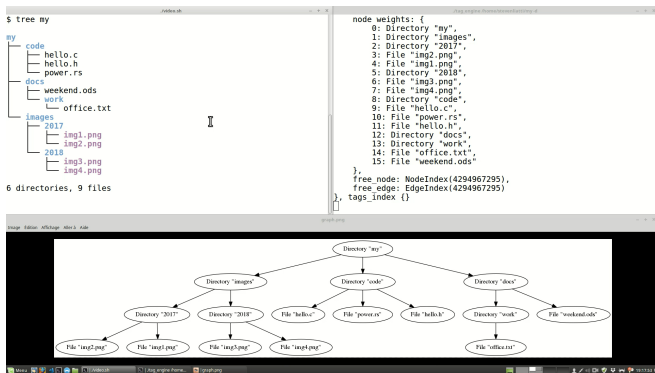


Figure – Système global TagFS

Démo



Vidéo

h e p i a

Haute école du paysage, d'ingénierie
et d'architecture de Genève

Mesures de performances

Répertoire	Sous-répertoires	Fichiers
Android	15'172	112'046
Documents	15'442	64'486
android-studio	3'331	13'287
bin	553	9'306
Dropbox	2'377	8'659
Musique	135	1'352
Images	5	863

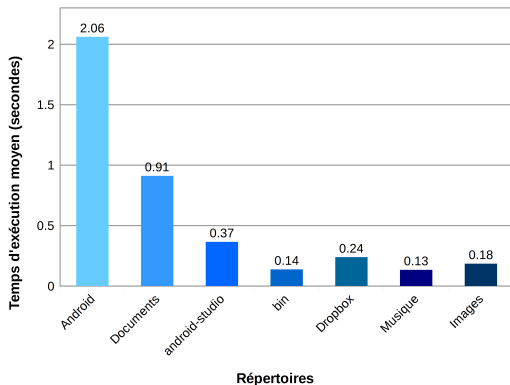


Figure – Temps d'exécution de Tag Engine en fonction des répertoires

Améliorations

Améliorations

- GUI : environnement de bureau ou application web.

Améliorations

- GUI : environnement de bureau ou application web.
- Daemon pour Tag Engine.

- GUI : environnement de bureau ou application web.
- Daemon pour Tag Engine.
- Ajout de nouveaux répertoires de surveillance (partiel).

Améliorations

- GUI : environnement de bureau ou application web.
- Daemon pour Tag Engine.
- Ajout de nouveaux répertoires de surveillance (partiel).
- Gestion des périphériques amovibles (limitation inotify).

Améliorations

- GUI : environnement de bureau ou application web.
- Daemon pour Tag Engine.
- Ajout de nouveaux répertoires de surveillance (partiel).
- Gestion des périphériques amovibles (limitation inotify).
- Cache des dernières requêtes logiques adressées au serveur.

Améliorations

- GUI : environnement de bureau ou application web.
- Daemon pour Tag Engine.
- Ajout de nouveaux répertoires de surveillance (partiel).
- Gestion des périphériques amovibles (limitation inotify).
- Cache des dernières requêtes logiques adressées au serveur.
- Ajouter des opérateurs logiques (NOT).

Bilan personnel

Bilan personnel

- Conception et réalisation d'un moteur de gestion de tags.

Bilan personnel

- Conception et réalisation d'un moteur de gestion de tags.
- Étude du langage Rust.

Bilan personnel

- Conception et réalisation d'un moteur de gestion de tags.
- Étude du langage Rust.
- **Progression personnelle : nouvelles technologies et bonnes pratiques.**

Remerciements

- Florent Glück
- Orestis Malaspinas
- Joël Cavat

Remerciements

- Florent Glück
- Orestis Malaspinas
- Joël Cavat

Merci pour votre attention ! Questions ?