Fiche Technique : Find-It

Explorateur de Fichiers en Rust

Projet WanderRust - EPITA

Date : 27 mai 2025

Auteurs : Jean Philippe, Clémence, Milan, Guillaume

Fiche Technique : Find-It

Informations Générales

— **Nom du projet** : Find-It

— **Type** : Explorateur de fichiers graphique

— **Objectif** : Fournir un outil léger, rapide et sécurisé pour naviguer, gérer et rechercher des fichiers sur Windows, macOS et Linux.

— **Équipe** : WanderRust (Jean Philippe Z., Clémence D., Milan M-L)

— **Contexte** : Projet académique réalisé à EPITA

— **Langage principal** : Rust

— **Version** : 1.0 (basée sur l'état du code au 27 mai 2025)

Technologies Utilisées

— **Langage** : Rust (choisi pour sa performance, sa sécurité mémoire et sa portabilité).

— **Bibliothèques principales** :

  — `iced` : Framework pour l'interface graphique, offrant des widgets réactifs et un rendu performant.

  — `walkdir` : Parcours récursif des répertoires.

  — `strsim` : Calcul de la distance de Levenshtein pour la recherche floue.

  — `std::fs` : Gestion des fichiers et dossiers (lecture, écriture, suppression).

— **Structures de données** :

  — `HashMap<String, Vec<PathBuf»` : Indexation des fichiers pour la recherche.

  — `HashMap<String, (Vec<SearchResult>, Instant)>` : Cache des résultats de recherche.

  — `HashSet<PathBuf>` : Élimination des doublons dans les résultats.

  — `Vec<FileEntry>` : Liste des fichiers affichés.

— **Plateformes supportées** : Windows, macOS, Linux (via des commandes spécifiques comme `cmd /C start`, `open`, `xdg-open`).

Fonctionnalités Principales

1. **Navigation dans les répertoires** :

   — Parcours des dossiers via une arborescence et une liste de fichiers.

   — Support des emplacements système (disques, dossiers utilisateur comme "Documents", "Bureau").

   — Navigation vers le dossier parent via un bouton dédié.

2. **Recherche de fichiers** :

   — Barre de recherche avec saisie dynamique et résultats instantanés.

   — Recherche floue basée sur la distance de Levenshtein, tolérant les fautes de frappe.

   — Priorisation des dossiers importants ("Documents", "Téléchargements") avec un bonus de score.

   — Cache des résultats (validité : 60 secondes) pour éviter les recalculs.

   — Indexation optimisée avec `HashMap` pour une recherche rapide.

3. **Gestion des fichiers** :

   — Création de fichiers et dossiers via un champ de saisie.

   — Suppression avec confirmation pour éviter les erreurs.

   — Copie, coupe et collage de fichiers/dossiers (support récursif pour les dossiers).

4. **Aperçu des fichiers** :

   - Prévisualisation des images (`jpg`, `png`, etc.), textes (`txt`, `md`, etc.), et informations sur les PDF/other fichiers.
   - Affichage des métadonnées (nom, taille, date de modification, type).
5. **Affichage des fichiers cachés** :
   - Bouton pour basculer l'affichage des fichiers/dossiers cachés (commençant par un point).
   - Mise à jour de l'index et des vues lors de l'activation.
6. **Ouverture de fichiers** :
   - Lancement des fichiers avec l'application par défaut du système (`start` sur Windows, `open` sur macOS, `xdg-open` sur Linux).

Architecture Logicielle

- **Modèle** : Application `iced` basée sur le patron ELM (modèle-état-message).
- **Composants principaux** :
  - `FileExplorer` : Structure principale contenant l'état (chemin courant, liste des fichiers, résultats de recherche, etc.).
  - `Message` : Énumération des événements utilisateur (changement de dossier, sélection de fichier, recherche, etc.).
  - `update` : Gère les transitions d'état en réponse aux messages.
  - `view` : Génère l'interface graphique (barre de navigation, liste de fichiers, détails, résultats de recherche).
- **Fonctions clés** :
- `load`$_{files}$: $Charge les fichiers d'un répertoire avec filtrage des fichiers cachés.$

Optimisations

- **Recherche** :
  - Indexation avec `HashMap` pour un accès rapide (O(1) en moyenne).
  - Cache des résultats pour réduire les recalculs.
- Filtrage des dossiers ignorés (`/.git/`, `node`$_{modules}$`/`, $etc.$).
- **Affichage** :
  - Limitation des résultats de recherche à 100 pour éviter la surcharge.
  - Chargement asynchrone des fichiers et prévisualisations via `Command`.
- **Ressources** :
  - Gestion efficace de la mémoire grâce à Rust.
  - Évitement des boucles infinies lors du parcours des répertoires avec `walkdir`.

Exemple de Code

Voici un extrait de la gestion de la recherche dans `search`$_{files}$:

```
for term in search_terms {
    for (indexed_term, paths) in &self.file_index {
        let distance = levenshtein(term, indexed_term) as f64;
        let max_len = term.len().max(indexed_term.len()) as f64;
        let mut score = 1.0 - (distance / max_len);
        if indexed_term == term {
            score *= 1.5; // Bonus pour correspondance exacte
        }
        // ... (calcul du score et ajout des r sultats)
    }
}
```

Listing 1 – Gestion de la recherche

Contraintes et Limites

- **Dépendance aux icônes** : Nécessite des fichiers d'icônes (icons/folder.png, etc.) dans le répertoire spécifié.
- **Recherche** : Limitée au répertoire courant et ses sous-dossiers (pas d'indexation globale du système).
- **Prévisualisation** : Support limité pour certains formats (par exemple, pas de rendu direct des PDF).
- **Permissions** : Peut échouer sur des fichiers/dossiers sans droits d'accès.

Comparaison avec les Alternatives

- **macOS Finder** :
  - Avantage : Recherche Spotlight rapide et intégrée.
  - Find-It : Recherche floue plus flexible, mais limitée au répertoire courant.
- **Linux Thunar** :
  - Avantage : Léger et personnalisable.
  - Find-It : Ajoute une recherche performante et une prévisualisation.
- **Windows File Explorer** :
  - Avantage : Compatibilité large.
  - Find-It : Plus rapide et moins gourmand en ressources.

Perspectives d'Évolution

- Indexation globale du système pour une recherche plus large.
- Support de la prévisualisation pour davantage de formats (PDF, vidéos).
- Ajout de filtres de recherche (par type, date, taille).
- Gestion avancée des permissions et erreurs.
- Internationalisation de l'interface (multilingue).

Conclusion

Find-It est un explorateur de fichiers moderne, conçu en Rust pour offrir performance, sécurité et portabilité. Ses fonctionnalités de navigation, recherche optimisée, gestion de fichiers et prévisualisation en font une alternative compétitive aux outils existants, avec un potentiel d'amélioration significatif pour les futures versions.