Rapport de réalisation — CSV Viewer (React + TypeScript)

Démo déployée : https://csv-viewer-flax.vercel.app/

Sommaire

1. Contexte & objectifs

L'objectif est de concevoir une application web légère permettant d'**ouvrir**, **explorer** et **exporter** des fichiers CSV, y compris de grande taille, avec une expérience fluide (import par glisser-déposer, tri, filtre, pagination, édition inline, export). Le tout en **React + TypeScript**, avec un **parser CSV** robuste et une **prévisualisation en streaming** des gros fichiers, ainsi qu'un **historique persistant** pour naviguer entre plusieurs fichiers tout en conservant les modifications utilisateur.

2. Méthodologie & démarche

La démarche adoptée est itérative et orientée retours rapides :

- Découpage en étapes minimales (MVP) : import de base → affichage → pagination → tri → filtre → édition → export → historique.
- Validation fréquente à partir d'exemples CSV couvrant de nombreux cas : séparateurs différents, guillemets, retours à la ligne, en-têtes manquants/dupliqués, encodages.

3. Architecture & choix techniques

Le projet s'appuie sur **React + TypeScript** (Vite) et une feuille de style simple. La logique est découpée pour isoler la persistance, la vue (tri/filtre/pagination) et les utilitaires. Les décisions de conception visent la lisibilité, la testabilité et la maintenabilité (hooks dédiés, utilitaires purs, composants focalisés).

4. Fonctionnalités livrées

- Import par clic ou glisser-déposer de fichiers .csv.

- Compatibilité CSV complète : séparateurs `;`/`,` (tabulation à l'export), guillemets `"` et `""`, retours à la ligne internes, CRLF/CR normalisés.
- Gestion d'encodage : UTF-8 (BOM ou non), fallback ISO-8859-1 (pour CSV Windows).
- Prévisualisation **en streaming des 5 000 premières lignes** pour les gros volumes (pas de chargement complet).
- Suppression des lignes dupliquées (égalité stricte après `trim`).
- Pagination (100 lignes/page par défaut).
- Tri par colonne (stable), nombres/dates auto, cellules vides en bas.
- Filtre global (accent-insensible).
- Édition inline des cellules (double-clic ; Entrée/Shift+Entrée/Échap/blur).
- Export CSV (vue courante ou toutes les données) ; séparateur configurable et BOM UTF-8.
- Historique des fichiers (sélection & suppression), persistance locale, restauration à l'ouverture.

5. Points clés d'implémentation

5.1 Parsing robuste

Le parser implémente un automate simple : état « inQuotes », prise en charge des `""` (échappement), détection dynamique du séparateur sur la première ligne, et normalisation universelle des fins de ligne. Les entêtes sont complétées/uniquées si manquantes/dupliquées.

5.2 Gros fichiers & streaming

La lecture s'effectue en **streaming** avec constitution de lots pour limiter l'empreinte mémoire et permettre un affichage progressif. Par conception, l'application ne charge qu'une **prévisualisation des 5 000 premières lignes** des fichiers volumineux. Il n'y a **pas** de déclenchement de lecture complète.

5.3 Tri, filtre, pagination

Le pipeline est maîtrisé : *filtre* → *tri* → *pagination*. Le tri est stable (index source en tiebreaker) et prend en charge nombres/dates via un comparateur dédié.

5.4 Historique persistant & robustesse

Chaque upload est associé à un **id de session**. Toutes les mises à jour (headers, batches, éditions) sont remontées avec cet id, ce qui évite les collisions lorsque l'utilisateur change de fichier pendant le chargement. La persistance localStorage est protégée : hydratation initiale (snapshot) puis écriture conditionnelle (pas d'écrasement sous StrictMode). La suppression d'une entrée persiste immédiatement l'état.

5.5 Édition & export

Les modifications sont répercutées sur les données « globales », donc l'export (vue courante ou complet) reflète toujours l'état modifié. L'export peut inclure BOM UTF-8 pour Excel et choisir le séparateur.

6. Déploiement & utilisation

Le projet est déployé sur Vercel : https://csv-viewer-flax.vercel.app/.
En local : `pnpm install` puis `pnpm dev` (Vite). Build de prod : `pnpm build` puis `pnpm preview`.

7. Limites & pistes d'amélioration

- Capacité de stockage locale (localStorage ≈5–10 Mo). Pour des historiques volumineux : basculer sur IndexedDB.
- Améliorer l'inférence de formats (dates locales multiples, monnaies).
- Export partiel (colonnes sélectionnées, filtres avancés, sauvegarde de vues).
- Undo/Redo des éditions et journal d'audit.
- Optimisations supplémentaires pour les CSV >1 Go (indexation, workers).

8. Recours aux LLMs

Dans une démarche pragmatique, j'ai tiré parti de différents **LLMs** pour accélérer et fiabiliser certaines parties, notamment :

- **streamParseCsv** : affinage de la logique de streaming (gestion des batches, déduplication à la volée, callbacks progressifs).
- **Historique persistant** : conception d'un mécanisme résilient à React StrictMode (hydratation avant persistance, snapshot, id de session pour désambiguïser les flux).
- Relectures de code et suggestions UX (accessibilité, clavier, messages d'erreur explicites, sécurité à l'export avec BOM).

La responsabilité de conception, d'intégration et des arbitrages techniques m'incombe. Les LLMs ont servi d'*assistant* pour explorer plus vite l'espace des solutions et documenter les choix.

9. Bilan

Le livrable respecte les exigences minimales et implémente de nombreux bonus : parsing robuste, gestion de gros volumes, pagination/tri/filtre, édition inline, export fiable et historique persistant. Le code est factorisé (hooks + utils) pour rester lisible et maintenable. Le déploiement sur Vercel garantit un accès simple à la démo. Des pistes sont identifiées pour aller plus loin (IndexedDB, workers, vues sauvegardées) selon les besoins.