**Documentation Projet : StockAV - Gestion de Stock Assistée par IA**

**Version :** 1.0 (Date : [08/04/2025])  
**Auteur :** [Zineddine Chergui]  
**Contact :** [chergui.zineddine@uqam.ca]

Table des Matières

1. [Introduction et Objectifs](https://www.google.com/url?sa=E&q=%231-introduction-et-objectifs)
2. [Architecture Technique](https://www.google.com/url?sa=E&q=%232-architecture-technique)
3. [Fonctionnalités Détaillées](https://www.google.com/url?sa=E&q=%233-fonctionnalit%C3%A9s-d%C3%A9taill%C3%A9es)
   * [3.1 Authentification](https://www.google.com/url?sa=E&q=%2331-authentification)
   * [3.2 Recherche de Composants (Chatbot IA)](https://www.google.com/url?sa=E&q=%2332-recherche-de-composants-chatbot-ia)
   * [3.3 Consultation du Stock](https://www.google.com/url?sa=E&q=%2333-consultation-du-stock)
   * [3.4 Modification Rapide des Quantités](https://www.google.com/url?sa=E&q=%2334-modification-rapide-des-quantit%C3%A9s)
   * [3.5 Historique des Mouvements](https://www.google.com/url?sa=E&q=%2335-historique-des-mouvements)
   * [3.6 Administration](https://www.google.com/url?sa=E&q=%2336-administration)
   * [3.7 Paramètres (Import/Export)](https://www.google.com/url?sa=E&q=%2337-param%C3%A8tres-importexport)
   * [3.8 Affichage Tiroir (7 Segments)](https://www.google.com/url?sa=E&q=%2338-affichage-tiroir-7-segments)
4. [Structure du Projet](https://www.google.com/url?sa=E&q=%234-structure-du-projet)
5. [Configuration et Déploiement](https://www.google.com/url?sa=E&q=%235-configuration-et-d%C3%A9ploiement)
   * [5.1 Prérequis](https://www.google.com/url?sa=E&q=%2351-pr%C3%A9requis)
   * [5.2 Configuration Supabase](https://www.google.com/url?sa=E&q=%2352-configuration-supabase)
   * [5.3 Configuration Locale](https://www.google.com/url?sa=E&q=%2353-configuration-locale)
   * [5.4 Déploiement Frontend](https://www.google.com/url?sa=E&q=%2354-d%C3%A9ploiement-frontend)
   * [5.5 Déploiement Fonction Edge](https://www.google.com/url?sa=E&q=%2355-d%C3%A9ploiement-fonction-edge)
6. [Améliorations Futures Possibles](https://www.google.com/url?sa=E&q=%236-am%C3%A9liorations-futures-possibles)
7. [Conclusion](https://www.google.com/url?sa=E&q=%237-conclusion)

**1. Introduction et Objectifs**

**StockAV** est une application web conçue pour simplifier et moderniser la gestion des stocks de composants électroniques, particulièrement adaptée aux environnements d'ateliers, laboratoires ou petites équipes.

L'objectif principal est de fournir une interface intuitive et rapide pour :

* Consulter l'état actuel du stock.
* Rechercher facilement des composants, y compris via un assistant conversationnel (IA) capable de suggérer des équivalents.
* Modifier rapidement les quantités lors de la prise ou du retour de pièces.
* Tracer les mouvements de stock pour une meilleure gestion.
* Permettre une administration centralisée des références et des catégories.
* Faciliter l'intégration avec des données existantes via import/export CSV.

Le projet vise à réduire le temps passé à chercher des composants, à minimiser les erreurs de stock et à offrir une visibilité claire sur les ressources disponibles.

**2. Architecture Technique**

StockAV adopte une architecture moderne basée sur des services cloud et des technologies web standard :

* **Frontend:**
  + **Langages:** HTML5, CSS3, JavaScript (ES6+ avec mode strict).
  + **Librairies/Frameworks:** Aucune librairie JS majeure (Vanilla JS), mais utilisation de :
    - **PapaParse:** Pour l'analyse et la génération de fichiers CSV.
    - **Supabase JS Client:** Pour l'interaction avec le backend Supabase.
  + **Hébergement:** Déployable sur des plateformes d'hébergement statique (GitHub Pages, Netlify, Vercel, etc.).
* **Backend & Base de Données (Supabase):**
  + **Base de Données:** PostgreSQL managée par Supabase. Structure principale :
    - inventory: Table principale du stock (réf, description, qté, catégorie, fabricant, datasheet, tiroir, seuil critique, attributs JSON).
    - categories: Table des catégories de composants (nom, liste d'attributs spécifiques).
    - logs: Table pour l'historique des mouvements (qui, quoi, quand, combien).
  + **Authentification:** Supabase Auth pour la gestion des utilisateurs (basée sur email/mot de passe, avec un système de code utilisateur simplifié).
  + **API:** Supabase fournit automatiquement une API RESTful pour interagir avec la base de données.
  + **Sécurité:** Row Level Security (RLS) activée sur les tables pour contrôler les permissions (lecture, écriture, suppression) en fonction de l'état d'authentification de l'utilisateur.
  + **Fonctions Edge (Deno):** Une fonction (ai-component-info) hébergée sur Supabase, écrite en TypeScript/Deno, pour interagir avec une API d'IA externe (OpenRouter) afin de trouver des équivalents de composants.
* **API Externe (IA):**
  + **OpenRouter.ai:** Utilisé comme passerelle pour accéder à des modèles de langage (LLM) comme Llama 3.1, afin de traiter les demandes de recherche d'équivalents.

graph LR

A[Navigateur Utilisateur<br>(HTML, CSS, script.js)] -- HTTPS --> B(Supabase);

B -- Auth --> BA(Supabase Auth);

B -- DB API --> BB(PostgreSQL<br>Tables: inventory, categories, logs);

B -- Function Call --> BC(Edge Function: ai-component-info<br>Deno/TypeScript);

BC -- API Call --> D(OpenRouter.ai<br>API LLM);

style A fill:#f9f,stroke:#333,stroke-width:2px

style B fill:#ccf,stroke:#333,stroke-width:2px

style D fill:#cfc,stroke:#333,stroke-width:2px

content\_copydownload

Use code [with caution](https://support.google.com/legal/answer/13505487).Mermaid

**3. Fonctionnalités Détaillées**

**3.1 Authentification**

* Système de connexion basé sur un "Code Technicien" (simplification pour l'utilisateur) et un mot de passe. Le code est combiné avec un domaine fictif (@stockav.local) pour former une adresse email utilisable par Supabase Auth.
* Affichage dynamique de la zone de connexion ou des informations utilisateur (code, bouton déconnexion).
* Protection des vues/fonctionnalités sensibles (Historique, Administration, Paramètres) : elles ne sont accessibles qu'aux utilisateurs connectés.
* **Restriction d'accès :** La vue Paramètres est spécifiquement réservée à l'utilisateur avec le code zine.

**3.2 Recherche de Composants (Chatbot IA)**

* Interface conversationnelle (type chatbot).
* L'utilisateur saisit une référence ou une question.
* Le script tente d'extraire une référence de composant (extractReference()).
* Vérifie le stock local de la référence (getStockInfoFromSupabase()).
* Appelle la fonction Edge ai-component-info pour obtenir des suggestions d'équivalents techniques.
* Affiche le statut de la référence originale et des équivalents suggérés (disponibilité, quantité, tiroir si connecté).
* Propose des boutons "Prendre" pour les composants disponibles (si connecté).
* Permet de confirmer la quantité à retirer, met à jour le stock (updateStockInSupabase()) et enregistre l'opération dans les logs (addLogEntry()).
* Fournit des liens de recherche externes (Mouser, Digi-Key, AliExpress) pour les composants non disponibles.

**3.3 Consultation du Stock**

* Vue "Stock Actuel" affichant le contenu de la table inventory sous forme de tableau.
* **Indicateur Visuel:** Un indicateur coloré (stock-indicator) signale l'état du stock (OK, Avertissement, Critique) en fonction de la quantité et du seuil critique défini.
* **Filtrage:** Par catégorie (liste déroulante dynamique) et/ou recherche textuelle (référence, description, fabricant, tiroir si connecté).
* **Pagination:** Affichage du stock par pages (ITEMS\_PER\_PAGE) avec boutons de navigation.
* **Affichage Conditionnel:** La colonne "Tiroir" n'est visible que pour les utilisateurs connectés.

**3.4 Modification Rapide des Quantités**

* En cliquant sur une ligne dans la vue "Stock Actuel", une fenêtre modale (quantity-change-modal) s'ouvre.
* Affiche la référence, la quantité actuelle et les **attributs spécifiques** du composant sous forme de badges colorés.
* Permet d'incrémenter ou décrémenter la quantité via des boutons +/-.
* Le bouton "Confirmer" met à jour la base de données (updateStockInSupabase()) et les logs (addLogEntry()), puis rafraîchit la vue Inventaire.

**3.5 Historique des Mouvements**

* Vue "Historique (LOG)" accessible aux utilisateurs connectés.
* Affiche le contenu de la table logs (Date/Heure, Utilisateur, Action, Réf, Changement, Stock final).
* Trié par date décroissante.
* Pagination incluse.

**3.6 Administration**

* Vue "Administration" accessible aux utilisateurs connectés.
* **Gestion des Catégories:**
  + Liste des catégories existantes.
  + Formulaire pour ajouter ou modifier une catégorie (Nom, Attributs séparés par virgules).
  + Boutons pour supprimer une catégorie (avec confirmation et vérification si elle est utilisée).
* **Gestion du Stock:**
  + Formulaire pour vérifier, ajouter ou modifier une référence de composant.
  + Bouton "Vérifier Stock" pré-remplit les informations si la référence existe.
  + Champs pour tous les détails: Catégorie, Description, Fabricant, Tiroir, Datasheet, Quantité, Seuil Critique.
  + **Attributs Spécifiques:** Des champs de saisie sont générés dynamiquement en fonction des attributs définis pour la catégorie sélectionnée.
  + Bouton "Mettre à jour" pour un ajustement rapide de la quantité.
  + Bouton "Enregistrer Composant" pour un ajout/mise à jour complet (upsert).

**3.7 Paramètres (Import/Export)**

* Vue "Paramètres" accessible **uniquement** à l'utilisateur zine.
* **Export:**
  + Bouton pour exporter l'inventaire complet au format CSV.
  + Bouton pour exporter l'historique complet au format TXT.
* **Import:**
  + Permet de sélectionner un fichier CSV pour importer/mettre à jour l'inventaire.
  + Format CSV attendu spécifié dans l'interface.
  + **Choix du Mode:** L'utilisateur peut choisir entre :
    - "Enrichir / Mettre à jour" (comportement upsert par défaut).
    - "Écraser l'inventaire existant" (effectue un delete complet avant l'upsert, avec confirmation).
  + Validation des données du CSV avant l'importation.
  + Affichage de messages de feedback (succès, erreurs de validation, erreurs DB).

**3.8 Affichage Tiroir (7 Segments)**

* Un afficheur 7 segments simulé en CSS est présent dans l'en-tête.
* Il s'allume et affiche la valeur du champ "Tiroir" du dernier composant consulté (via clic inventaire, vérification admin, ou sélection chat) **uniquement si l'utilisateur est connecté**.
* Gère l'affichage des chiffres et de certains caractères (A à H, L, P, S, U, -, \_).

**4. Structure du Projet**

La structure attendue des fichiers clés est la suivante :

/ (racine du projet)

├── index.html # Fichier HTML principal

├── style.css # Feuille de style CSS

├── script.js # Script JavaScript principal (frontend)

│

└── supabase/ # Dossier pour la configuration Supabase CLI

├── functions/ # Dossier pour les Edge Functions

│ ├── \_shared/ # Dossier pour le code partagé entre fonctions

│ │ └── cors.ts # Gestion des en-têtes CORS (exemple)

│ └── ai-component-info/ # Fonction Edge spécifique

│ └── index.ts # Code de la fonction Edge (TypeScript/Deno)

└── config.toml # Configuration projet Supabase (généré par `supabase init`)

content\_copydownload

Use code [with caution](https://support.google.com/legal/answer/13505487).

**5. Configuration et Déploiement**

**5.1 Prérequis**

* Compte Supabase ([supabase.com](https://www.google.com/url?sa=E&q=https%3A%2F%2Fsupabase.com%2F))
* Compte OpenRouter.ai ([openrouter.ai](https://www.google.com/url?sa=E&q=https%3A%2F%2Fopenrouter.ai%2F)) pour obtenir une clé API (si utilisation de l'IA pour équivalents).
* Node.js et npm (pour outils de développement éventuels, non strictement requis pour ce projet Vanilla JS).
* Supabase CLI installée et configurée (supabase login, supabase link).
* Docker Desktop (requis par Supabase CLI pour le build des fonctions Edge).

**5.2 Configuration Supabase**

1. **Créer un Projet Supabase.**
2. **Activer l'Authentification:** Configurer les fournisseurs (Email est utilisé ici).
3. **Créer les Tables (via l'éditeur SQL ou l'interface) :**
   * categories (id uuid pk, name text unique, attributes text[])
   * inventory (ref text pk, description text, manufacturer text, quantity integer, datasheet text, drawer text, category\_id uuid fk->categories.id, critical\_threshold integer, attributes jsonb)
   * logs (id uuid pk, created\_at timestampz default now(), user\_id uuid fk->auth.users.id, user\_code text, component\_ref text, quantity\_change integer, quantity\_after integer)
4. **Configurer la Row Level Security (RLS):**
   * Activer RLS pour les tables inventory, categories, logs.
   * Ajouter les politiques SQL nécessaires (voir section 6 du script ou documentation dédiée) pour autoriser les opérations SELECT, INSERT, UPDATE, DELETE aux utilisateurs connectés (authenticated), en restreignant éventuellement certaines opérations à des rôles spécifiques si implémenté.
5. **Configurer les Secrets (via CLI ou Dashboard):**
   * supabase secrets set OPENROUTER\_API\_KEY=sk-or-v1-...
   * supabase secrets set APP\_URL\_REFERER=https://votre-domaine.com (URL de votre application déployée)

**5.3 Configuration Locale**

* Cloner le dépôt du projet.
* Aucune étape de build spécifique n'est requise pour le frontend (Vanilla JS).
* Assurer que les constantes SUPABASE\_URL et SUPABASE\_ANON\_KEY dans script.js correspondent à votre projet Supabase.

**5.4 Déploiement Frontend**

* Déployer les fichiers index.html, style.css, script.js (et autres assets éventuels comme favicon.ico) sur une plateforme d'hébergement statique (GitHub Pages, Netlify, Vercel, Cloudflare Pages, etc.).
* S'assurer que l'URL de déploiement correspond au secret APP\_URL\_REFERER configuré dans Supabase.

**5.5 Déploiement Fonction Edge**

1. Naviguer vers la racine du projet dans le terminal.
2. S'assurer que la structure supabase/functions/ai-component-info/index.ts est correcte.
3. Exécuter la commande :

supabase functions deploy ai-component-info --no-verify-jwt

content\_copydownload

Use code [with caution](https://support.google.com/legal/answer/13505487).Bash

1. Vérifier le déploiement dans le Dashboard Supabase > Edge Functions.

**6. Améliorations Futures Possibles**

* **Système de Rôles:** Implémenter des rôles ('admin', 'user') pour une gestion plus fine des permissions (RLS + contrôle UI).
* **Interface de Gestion Utilisateurs:** Si des rôles sont ajoutés, permettre aux administrateurs de gérer les utilisateurs.
* **Feedback Visuel Amélioré:** Notifications temporaires pour succès/erreurs, indicateurs de progression pour l'import/export.
* **Validation Temps Réel:** Ajouter des validations plus interactives dans les formulaires.
* **Recherche par Attributs:** Fonctionnalité de filtrage avancée basée sur les paires clé/valeur des attributs JSONB.
* **Gestion des Emplacements Multiples:** Si le stock est réparti sur plusieurs sites/boîtes .
* **Photos des Composants:** Permettre d'associer une image à chaque référence.
* **Mode Hors Ligne:** Utilisation de Service Workers pour une consultation basique hors connexion.
* **Tests Automatisés:** Ajout de tests unitaires et d'intégration.

**7. Conclusion**

StockAV offre une solution fonctionnelle et moderne pour la gestion de stock de composants électroniques. En combinant une interface utilisateur simple, la puissance de Supabase et une touche d'intelligence artificielle, l'application vise à améliorer l'efficacité et la fiabilité du suivi des stocks. La base technique est solide et permet des évolutions futures pour répondre à des besoins plus complexes.