

# RAPPORT DE DOCUMENTATION

## Dashboard Application — Projet Vue.js

---

## INFORMATIONS SUR LE PROJET

**Nom et Prénom :** Nicolas Bellina

**Titre du projet :** Dashboard Application – Plateforme d'administration e-commerce

**Lien vers le dépôt GitHub :** <https://github.com/NicolasBellina/Dashboard>

**Établissement :** ESGI – M1 Ingénierie du Web

---

## RÉSUMÉ DU PROJET

Ce projet consiste en la réalisation d'un dashboard d'administration e-commerce développé avec Vue.js 3.

L'application permet de gérer différents éléments d'une plateforme de vente en ligne, notamment :

- les produits
- les utilisateurs
- les commandes
- les statistiques de vente

Le dashboard propose une interface moderne, responsive et intuitive permettant de visualiser des indicateurs clés et d'effectuer des opérations CRUD sur les données.

Le projet utilise plusieurs technologies modernes du développement web :

- Vue.js 3 avec la Composition API
- Pinia pour la gestion d'état
- Vue Router pour la navigation-
- Axios pour les appels API
- Chart.js pour la visualisation des données

# APERÇU DES FONCTIONNALITÉS

## 1. Authentification

- Page de connexion avec formulaire de validation
- Routes protégées (accès au dashboard uniquement si authentifié)
- Fonctionnalité de déconnexion
- État d'authentification persistant avec localStorage
- Redirection automatique vers /login si l'utilisateur n'est pas authentifié
- Redirection vers / si l'utilisateur est déjà connecté

## 2. Dashboard Home

Affichage de statistiques principales :

- Total des ventes
- Nombre d'utilisateurs
- Nombre de produits
- Nombre total de commandes

Fonctionnalités :

- Cartes de statistiques
- Section d'actions rapides
- Vue d'ensemble des métriques principales
- Interface moderne et responsive

## 3. Gestion des Produits

Fonctionnalités principales :

- Liste complète des produits
- Affichage sous forme de grille responsive
- Pagination
- Recherche en temps réel par titre
- Filtrage par catégorie

Fonctions CRUD :

- Création de produit
- Modification de produit
- Suppression avec confirmation

Les produits affichent :

- image
- prix
- note
- description
- catégorie

---

## 4. Gestion des Utilisateurs

Fonctionnalités :

- Liste des utilisateurs
- Affichage des informations :
  - nom
  - email
  - téléphone
  - adresse

Autres fonctionnalités :

- Recherche utilisateur
- Affichage du rôle (admin / user)
- Modal de détails utilisateur
- Suppression d'utilisateur avec confirmation

## 5. Analytics / Rapports

La section analytics permet de visualiser les performances de la plateforme.

Graphiques disponibles :

- ventes par période (jour / semaine / mois)
- produits les plus vendus

Statistiques affichées :

- chiffre d'affaires total
- nombre de commandes
- valeur moyenne par commande

Les graphiques sont réalisés avec Chart.js.

---

## 6. Navigation et Layout

L'application propose une navigation claire et cohérente.

Éléments principaux :

- Sidebar (menu latéral)
- Navbar (barre supérieure)
- Breadcrumbs (fil d'Ariane)
- Interface responsive

Le layout reste identique sur toutes les pages pour assurer une expérience utilisateur cohérente.

# ARCHITECTURE GÉNÉRALE DE L'APPLICATION

L'application suit une architecture front-end classique basée sur Vue.js.

Utilisateur



Interface Vue.js (Views / Components)



State Management (Pinia Stores)



Service API centralisé



FakeStoreAPI (API externe)

# TECHNOLOGIES UTILISÉES

## Framework Frontend

### Vue 3

Framework JavaScript progressif permettant de créer des interfaces dynamiques.

Utilisation de :

- Composition API
- composants réutilisables
- reactive state

## Build Tool

### Vite

Outil de développement rapide permettant :

- démarrage instantané
- hot module replacement
- build optimisée

## Routing

### Vue Router

Permet :

- la navigation entre les pages
- la protection des routes
- la gestion des redirections

# **State Management**

## **Pinia**

Utilisé pour gérer l'état global de l'application :

- authentification
- produits
- utilisateurs
- analytics

# **HTTP Client**

## **Axios**

Permet d'effectuer des requêtes HTTP vers l'API.

Avantages :

- configuration centralisée
- gestion des erreurs
- gestion des timeouts
- code plus lisible

# **Visualisation des données**

## **Chart.js**

Utilisé pour créer :

- graphiques de ventes
- graphiques de produits

# **Tests**

## **Vitest**

Framework de tests pour vérifier le bon fonctionnement des composants et des stores.

# GESTION DE L'ÉTAT AVEC PINIA

L'application utilise 4 stores principaux.

## Auth Store

Responsable de l'authentification utilisateur.

Fonctions :

- login
- logout
- persistance de session
- vérification de connexion

## Products Store

Responsable de la gestion des produits.

Fonctions :

- récupération des produits
- filtrage par catégorie
- création
- modification
- suppression

## Users Store

Responsable de la gestion des utilisateurs.

Fonctions :

- récupération des utilisateurs
- suppression
- affichage des détails

## **Analytics Store**

Responsable des statistiques.

Fonctions :

- calcul du chiffre d'affaires
- nombre de commandes
- moyenne par commande

## **INTÉGRATION API**

Les appels API sont centralisés dans :

src/services/api.js

API utilisée :

<https://fakestoreapi.com>

Endpoints :

Produits

GET /products  
GET /products/:id  
GET /products/categories  
GET /products/category/:category

Utilisateurs

GET /users  
GET /users/:id

Commandes

GET /carts

# AMÉLIORATIONS POSSIBLES

Avec plus de temps, plusieurs améliorations pourraient être apportées :

## Backend réel

- authentification JWT
- base de données
- API sécurisée

## Fonctionnalités supplémentaires

- notifications temps réel
- export de rapports
- mode sombre
- recherche avancée

## Améliorations techniques

- migration vers TypeScript
- tests plus complets
- optimisation des performances
- accessibilité

# **INSTALLATION DU PROJET**

Regarder le readme dans le projet Github.

## **TESTS ET VALIDATION**

Les tests réalisés incluent :

- tests de connexion
- navigation entre les pages
- création de produit
- modification de produit
- suppression de produit
- affichage des utilisateurs
- affichage des graphiques

Tests de responsive :

- Desktop
- Tablette
- Mobile

Tests d'erreurs :

- erreur réseau
- API indisponible
- validation de formulaire
- accès non autorisé