

Rapport de testing

I. Recommandations

A. Tests à automatiser

Les 2 tests les plus critiques sont :

- **Connexion** : C'est l'entrée principale pour accéder aux fonctionnalités du site. Un problème ici empêcherait les utilisateurs de mettre des articles dans le panier, et donc d'acheter.
- **Panier** : Un bug dans le panier empêcherait les commandes et impacterait directement les ventes.

Les tests d'affichage, bien qu'importants, sont moins critiques car un bug ici peut être frustrant pour l'utilisateur, mais n'empêche pas directement l'achat. De plus, les erreurs d'affichage sont souvent visuelles et moins évidentes à tester automatiquement.

B. Préconisations pour la suite

A l'avenir, quand l'équipe QA aura le budget nécessaire, je préconiserais d'automatiser également les **tests d'affichage**.

En effet, s'assurer que les produits s'affichent correctement est essentiel pour l'expérience utilisateur (navigation et prise de décision).

Vérifier que les bonnes informations (description, prix, stock, images) s'affichent correctement peut éviter des erreurs qui pourraient impacter la confiance des utilisateurs envers le site, et donc le passage à l'achat.

II. Bilan de campagne de validation : tests automatisés

Document revu par :

Nom	Fonction	Version	Signature
-----	----------	---------	-----------

R. Molina	QA Engineer	1.0.0	

Contexte :

Eco Bliss Bath sort sa première version de site en ligne de vente de produits de beauté écoresponsables dont le produit principal est un savon solide.

Cette première version a été testée manuellement par la QA testeuse. En tant que QA Engineer, j'ai défini l'automatisation des tests à la demande du CTO, en me basant sur le bilan des tests manuels et les contraintes budgétaires du PO. Les tests ont duré une semaine.

L'environnement des tests présentés ci-après est le suivant :

- système d'exploitation : Windows 10 Pro
- navigateur : Chrome version 134.0.6998.89 (Build officiel) (64 bits)
- logiciels :
 - VS Code v1.98.2
 - Docker v4.39.0
 - Cypress v14.2.1

Objectifs :

- **Assurer la conformité et le bon fonctionnement du site** par rapport aux spécifications et aux besoins utilisateurs.
- **Évaluer la première version du produit** pour prévenir les défauts, détecter les anomalies et les corriger avant la mise en production, réduisant ainsi les coûts et garantissant la satisfaction client.

Scénarios de tests :

1. **Tests API automatisés** : Valider le travail de conception et détecter les bugs dans les endpoints, les données retournées et le traitement des requêtes.

2. **Tests automatisés des fonctionnalités critiques** : Vérifier la connexion et le panier pour garantir des achats réussis et des commandes valides.
3. **Smoke tests automatisés** : Valider rapidement les fonctionnalités de base du site et assurer leur stabilité lors de l'ajout de nouvelles fonctionnalités.
4. **Tests de faille XSS** : Etant donné que le site comporte un formulaire (pour la publication d'avis), il est fortement recommandé de faire des tests de faille XSS (Cross-Site Scripting), afin de garantir l'intégrité du site ainsi que la sécurité des utilisateurs.

Tests effectués :

Tests API

- i) Requête sur les **données confidentielles** d'un utilisateur **avant connexion** :
Vérifie qu'un utilisateur non authentifié ne peut pas accéder au panier
URL : http://localhost:8081/orders
method: 'GET'
Résultat attendu : code 401 (non authentifié) ou 403 (pas les bons droits)
- ii) Tentative de **connexion** avec un **utilisateur inconnu**
URL : http://localhost:8081/login
method: 'POST'
Résultat attendu : code 401 (non authentifié)
- iii) Tentative de **connexion** avec un **utilisateur connu**
URL : http://localhost:8081/login
method: 'POST'
 - email : test2@test.fr
 - mot de passe : testtest

Résultat attendu :

- code 200
- token présent dans la réponse

NB : les tests suivants nécessitent d'être connecté.

iv) **Ajout d'un produit** disponible **au panier**

URL : http://localhost:8081/orders/add

method: 'PUT'

Résultat attendu : code 200

v) Requête de la **liste** des **produits** du **panier**

URL : http://localhost:8081/orders

method: 'GET'

Résultat attendu : le panier contient l'article précédemment ajouté.

vi) Requête d'une **fiche produit** spécifique

URL : http://localhost:8081/products/{id}

method: 'GET'

Résultat attendu :

- code 200
- l'id du produit correspond à celui spécifié dans l'URL.

vii) **Ajout d'un produit** indisponible **au panier**

URL : http://localhost:8081/orders/add

method: 'PUT'

Résultat attendu : code 400

viii) **Ajout d'un avis**

URL : http://localhost:8081/reviews

method: 'POST'

Résultat attendu :

- code 200
- l'avis a un ID
- la note correspond à celle renseignée
- le texte du commentaire correspond à celui renseigné

URL : http://localhost:8080

- Cliquer sur le bouton de connexion ;
- La page de connexion avec le formulaire s'affiche ;
- Entrer "test2@test.fr" dans le champ de l'email ;
- Entrer "testtest" comme mot de passe ;
- Cliquer sur le bouton « Se connecter » ;

Résultat attendu :

- bouton panier visible
- bouton « déconnexion » visible

Tests panier

NB 1 : les tests suivants nécessitent d'être connecté.

NB 2 : Le panier doit être vidé avant chaque test.

[le hook beforeEach a été utilisé pour automatiser cela]

i) **Ajout d'un produit disponible au panier et vérification de la mise à jour du stock :**

- Cliquer sur le lien des produits ;
- Cliquer sur le quatrième produit de la liste (stock > 1) ;
- Cliquer sur le bouton d'ajout au panier ;
- Vérifier que le bon produit est dans le panier ;
- Retourner sur la page du produit et vérifier la bonne mise à jour du stock (le stock doit avoir diminué de la quantité mise dans le panier) ;

Résultats attendus :

- Le produit a été ajouté au panier
- le stock produit a diminué de la quantité mise dans le panier

ii) Vérification des **limites**

- Cliquer sur le lien des produits ;
- Cliquer sur le quatrième produit de la liste (stock > 1) ;
 - Limite **inférieure** :
- Entrer une quantité négative ;

- Cliquer sur le bouton d'ajout au panier ;
- Cliquer sur le lien « Mon panier » ;
- Vérifier qu'aucun produit n'est présent ;

Résultat attendu : panier vide

– Limite **supérieure** :

- Entrer une quantité supérieure à 20 ;
- Cliquer sur le bouton d'ajout au panier ;
- Cliquer sur le lien « Mon panier » ;
- Vérifier qu'aucun produit n'est présent ;

Résultat attendu : panier vide

iii) **Ajout** d'un élément au **panier via frontend** et **vérification** du contenu du panier **via l'API** :

– Ajout d'un article au panier via frontend :

- Cliquer sur le lien des produits ;
- Cliquer sur le quatrième produit de la liste (stock > 1) ;
- Cliquer sur le bouton d'ajout au panier ;

– Vérification du contenu du panier via l'API :

Récupérer le token d'authentification

URL : http://localhost:8081/orders

method: 'GET'

Résultats attendus :

- code 200
- le nom du produit ajouté est présent dans le panier

Smoke tests

i) Présence des **champs et boutons de connexion** :

URL : http://localhost:8080

- Cliquer sur le bouton de connexion ;
- La page de connexion avec le formulaire s'affiche ;
- Vérifier la présence des champs et boutons de connexion ;

Résultats attendus :

- Le champ « Email » doit être présent
- Le champ « Mot de passe » doit être présent
- Le bouton « Se connecter » doit être présent

- ii) Présence des **boutons d'ajout au panier** pour un utilisateur connecté :

URL : http://localhost:8080

- Cliquer sur le bouton de connexion ;
- La page de connexion avec le formulaire s'affiche ;
- Entrer "test2@test.fr" dans le champ de l'email ;
- Entrer "testtest" comme mot de passe ;
- Cliquer sur le bouton « Se connecter » ;
- Cliquer sur le premier produit de la page d'accueil ;
- Vérifier la présence du bouton d'ajout au panier ;

Résultat attendu :

- Le bouton « Ajouter au panier » doit être présent

- iii) Présence du **champ de disponibilité du produit** :

URL : http://localhost:8080

- Cliquer sur le premier produit de la page d'accueil ;
- Vérifier la présence du champ de disponibilité du produit ;

Résultat attendu :

- Le champ indiquant la quantité en stock doit être présent

Tests de faille XSS

- i) Frontend :

URL : http://localhost:8080

- Cliquer sur le bouton de connexion ;
- La page de connexion avec le formulaire s'affiche ;
- Entrer "test2@test.fr" dans le champ de l'email ;

- Entrer "testtest" comme mot de passe ;
- Cliquer sur le bouton « Se connecter » ;
- Cliquer sur la page d'avis ;
- Renseigner le titre de l'avis
- Saisir le script XSS « `<script>alert("XSS")</script>` » dans le champ de commentaire
- Sélectionner une note
- Cliquer sur « Publier »
- Vérifier que le HTML n'est pas injecté
- Vérifier que le texte brut est affiché

Résultats attendus :

- Le HTML n'est pas injecté
- Le texte brut est affiché

ii) Backend :

Se connecter

URL : `http://localhost:8081/reviews`

method: ' POST'

Résultats attendus :

- code 200
- le script est échappé
- le script est affiché comme texte brut

Résultats de tests

Tests API

i) Requête sur les **données confidentielles** d'un utilisateur **avant connexion**



ii) Tentative de **connexion** avec un **utilisateur inconnu**



iii) Tentative de **connexion** avec un **utilisateur connu**



iv) **Ajout d'un produit** disponible **au panier**



v) Requête de la **liste** des **produits** du **panier**



vi) Requête d'une **fiche produit** spécifique



vii) **Ajout d'un produit** indisponible **au panier**



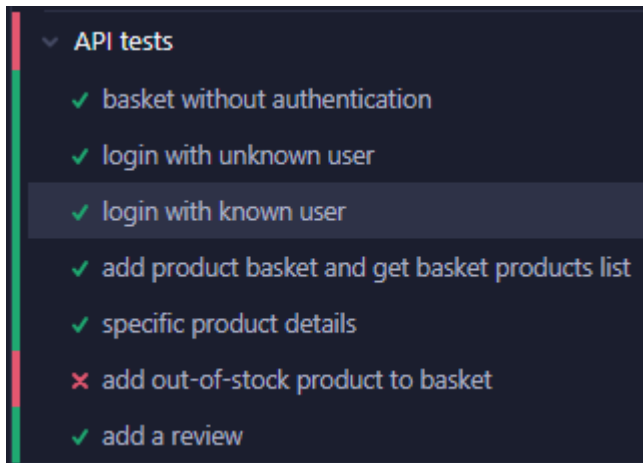
Résultat attendu : code 400

Résultat obtenu : code 200

viii) **Ajout d'un avis**



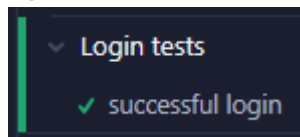
Synthèse :



Test de connexion Front-end



Synthèse :



Tests panier

- i) **Ajout** d'un **produit** disponible **au panier** et vérification de la **mise à jour du stock**



- ii) Vérification des **limites**

- Limite **inférieure**



- Limite **supérieure**



Résultat attendu :

- panier vide

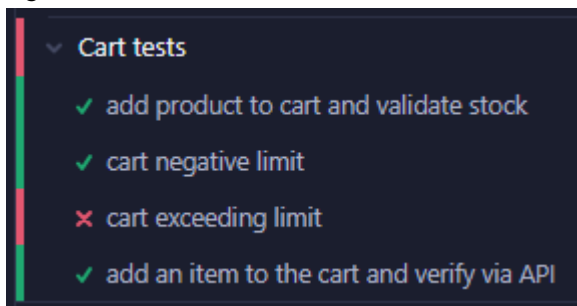
Résultat obtenu :

- article présent dans le panier

- iii) **Ajout** d'un élément au **panier via frontend** et **vérification** du contenu du panier **via l'API**



Synthèse :



Smoke tests

- i) Présence des **champs et boutons de connexion**



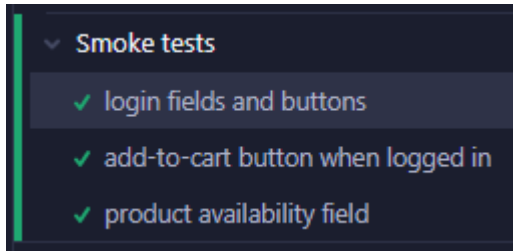
- ii) Présence des **boutons d'ajout au panier** pour un utilisateur connecté



- iii) Présence du **champ de disponibilité du produit**



Synthèse :



Tests de faille XSS

i) Frontend :

Résultats attendus :

- Le HTML n'est pas injecté
- Le texte brut est affiché

Résultats obtenus :

- Le HTML n'est pas injecté ✓
- Le texte brut n'est pas affiché ✗

ii) Backend :

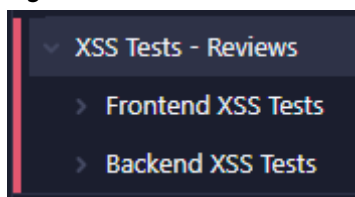
Résultats attendus :

- code 200
- le script est échappé
- le script est affiché comme texte brut

Résultats obtenus :

- code 200 ✓
- le script n'est pas échappé ✗
- le script n'est pas affiché comme texte brut ✗

Synthèse :



Rapport d'incidents

Anomalie 1:

10/04/2025

Ajout au panier possible d'un produit indisponible

1. Environnement :

- Système d'exploitation : Windows 10 Pro
- Navigateur : Chrome version 134.0.6998.89 (Build officiel) (64 bits)

2. Étape pour reproduire le bug :

URL : `http://localhost:8081/orders/add`
method: 'PUT'

Indiquer dans le corps de la requête :
l'id du produit (stock < 0) à ajouter
la quantité à ajouter : 1

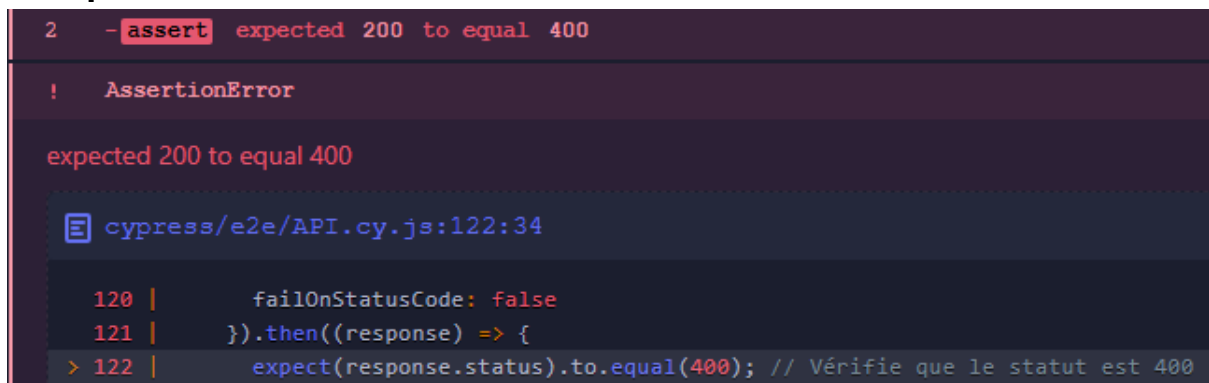
3. Comportement attendu :

Code 400

4. Comportement observé :

Code 200

5. Capture d'écran :



```
2 - assert expected 200 to equal 400

! AssertionError

expected 200 to equal 400

cypress/e2e/API.cy.js:122:34

120 |         failOnStatusCode: false
121 |     }).then((response) => {
> 122 |         expect(response.status).to.equal(400); // Vérifie que le statut est 400
```

6. Autres informations :

Impact : Le bug peut générer des commandes qui ne pourront pas être honorées.

Répétabilité : Toujours

Actions recommandées : Verrouiller la mise au panier des articles en rupture de stock.

Anomalie 2 :

10/04/2025

Ajout d'un produit au panier possible pour une quantité supérieure à 20

1. Environnement :

- Système d'exploitation : Windows 10 Pro
- Navigateur : Chrome version 134.0.6998.89 (Build officiel) (64 bits)

2. Étape pour reproduire le bug :

- Se connecter ;
- Vider le panier ;
- Cliquer sur le lien des produits ;
- Cliquer sur le quatrième produit de la liste (stock > 1) ;
- Entrer une quantité supérieure à 20 ;
- Cliquer sur le bouton d'ajout au panier ;
- Cliquer sur le lien « Mon panier » ;
- Vérifier qu'aucun produit n'est présent ;

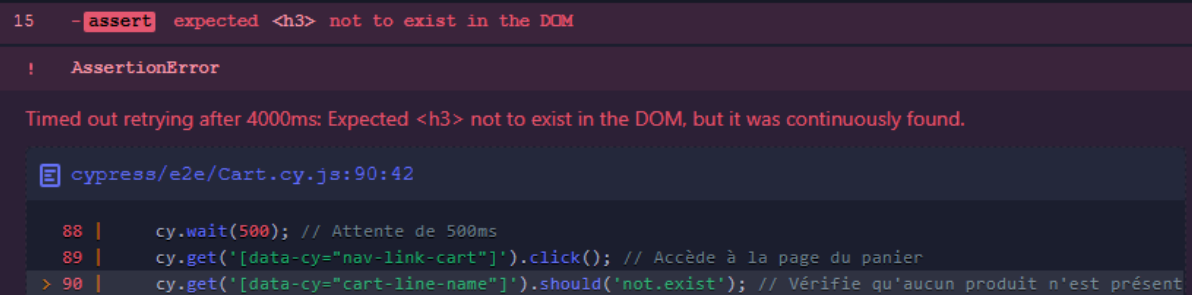
3. Comportement attendu :

Panier vide

4. Comportement observé :

Article présent au panier

5. Captures d'écran :



```
15 - assert expected <h3> not to exist in the DOM
! AssertionError
Timed out retrying after 4000ms: Expected <h3> not to exist in the DOM, but it was continuously found.
cypress/e2e/Cart.cy.js:90:42
88 |   cy.wait(500); // Attente de 500ms
89 |   cy.get('[data-cy="nav-link-cart"]').click(); // Accède à la page du panier
> 90 |   cy.get('[data-cy="cart-line-name"]').should('not.exist'); // Vérifie qu'aucun produit n'est présent
```

Accueil


Produits

Avis

eco.bliss.bath

Mon panier

Déconnexion



profondeur en laissant votre peau douce et hydratée.

PEAU

Peau mixte

AROMES

Bois de santal

INGRÉDIENTS CLÉS

Soude caustique

24,00
€

1

Ajouter au panier

12 en stock

6. Autres informations :

Impact : Le bug peut générer des commandes qui ne pourront pas être honorées.

Répétabilité : Toujours

Actions recommandées : Verrouiller la mise au panier pour des quantités supérieures à 20 unités d'un même article.

Anomalie 3:

10/04/2025

Présence d'une potentielle faille XSS dans le formulaire d'avis

1. Environnement :

- Système d'exploitation : Windows 10 Pro
- Navigateur : Chrome version 134.0.6998.89 (Build officiel) (64 bits)

A] Frontend

2. Étape pour reproduire le bug :

URL : http://localhost:8080

- Cliquer sur le bouton de connexion ;
- La page de connexion avec le formulaire s'affiche ;
- Entrer "test2@test.fr" dans le champ de l'email ;
- Entrer "testtest" comme mot de passe ;
- Cliquer sur le bouton « Se connecter » ;
- Cliquer sur la page d'avis ;
- Renseigner le titre de l'avis
- Saisir le script XSS « `<script>alert("XSS")</script>` » dans le champ de commentaire
- Sélectionner une note
- Cliquer sur « Publier »
- Vérifier que le HTML n'est pas injecté
- Vérifier que le texte brut est affiché

3. Comportement attendu :

- Le HTML n'est pas injecté
- Le texte brut est affiché

4. Comportement observé :

- Le HTML n'est pas injecté
- Le texte brut n'est pas affiché

5. Captures d'écran :

```
! AssertionError

Timed out retrying after 4000ms: expected '[ <p>, 50 more... ]' to contain text '<script>', but the text was
'ExcellentExcellentExcellentExcellentExcellentExcellenttestttttttExcellentExcellentExcellentExcellentessaisstringstring'adore l'odeur des produits !J'aime
beaucoup les savons proposés par EcoBlissBath !Bien, mais un peu cher'

cypress/e2e/XSS_Reviews.cy.js:29:44

27 | // Vérifie que le script n'est pas exécuté et est correctement échappé
28 | cy.get('[data-cy="review-comment"]').should('not.contain.html', xssPayload); // Vérifie que le HTML n'est pas injecté
> 29 | cy.get('[data-cy="review-comment"]').should('contain.text', '&lt;script&gt;'); // Vérifie que le texte brut est affiché
```



test - XSS Test



A] Backend

2. Étape pour reproduire le bug :

Se connecter à l'API

URL : `http://localhost:8081/reviews`

method: 'POST'

Indiquer dans le corps de la requête :

- le titre de l'avis

- le script XSS « `<script>alert("XSS")</script>` » dans le champ de commentaire
- une note

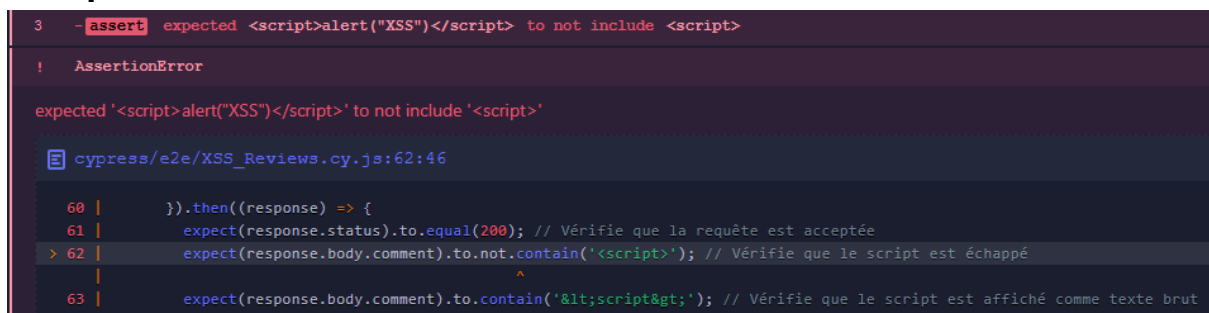
3. Comportement attendu :

- code 200
- le script est échappé
- le script est affiché comme texte brut

4. Comportement observé :

- code 200
- le script n'est pas échappé
- le script n'est pas affiché comme texte brut

5. Capture d'écran :



```

3 -assert expected <script>alert("XSS")</script> to not include <script>

! AssertionError

expected '<script>alert("XSS")</script>' to not include '<script>'

cypress/e2e/XSS_Reviews.cy.js:62:46

60 |     }).then((response) => {
61 |       expect(response.status).to.equal(200); // Vérifie que la requête est acceptée
> 62 |       expect(response.body.comment).to.not.contain('<script>'); // Vérifie que le script est échappé
    |                                             ^
63 |       expect(response.body.comment).to.contain('&lt;script&gt;'); // Vérifie que le script est affiché comme texte brut

```

6. Autres informations :

Impact : Une faille XSS dans le formulaire peut remettre en question l'intégrité du site ainsi que la sécurité des utilisateurs.

Répétabilité : Toujours

Actions recommandées : Utiliser des moteurs de template avec échappement automatique (ex : Twig, Blade ou Pug).

Confiance

Au vu des anomalies identifiées lors de la campagne de tests, je ne peux pas avoir confiance en la version actuelle de l'application.

Les défauts relevés impactent des aspects fondamentaux du site e-commerce, tels que la **sécurité**, la **gestion du stock** et **l'intégrité fonctionnelle du parcours d'achat**.

Les trois anomalies détectées sont **critiques** :

- **La faille XSS sur le formulaire d'avis** représente un **risque majeur de sécurité**, pouvant exposer les utilisateurs à des attaques malveillantes et nuire à la réputation de la plateforme.

- **L'ajout au panier de produits indisponibles et la possibilité de dépasser les quantités maximales autorisées** remettent en question la **fiabilité du système de stock**, ce qui peut générer des commandes non honorables et provoquer une insatisfaction client.

Par conséquent, **la faille XSS doit être corrigée en priorité**, en raison de sa nature sensible et de ses potentielles conséquences légales.

Les problèmes de stock doivent également être traités immédiatement pour garantir la cohérence des commandes et préserver la confiance des utilisateurs.

Il est donc **fortement recommandé de ne pas déployer cette version en production tant que ces anomalies critiques n'ont pas été résolues et vérifiées.**

Dans un second temps, je préconiserais de revoir l'affichage et la gestion des stocks, et d'approfondir les tests à ce sujet. En effet, lors de la réalisation de certains tests, j'ai pu m'apercevoir qu'il y avait des incohérences, telles que l'affichage de stocks négatifs, ou la possibilité de renseigner une quantité négative, ou supérieure au stock, avant ajout au panier.