

Nantes Université - IUT - BUT Informatique

**Rapport de SAE**

Développement Efficace

eq\_02\_01

*OULMAS Thomas, ROUX Célestin, MOREAU–THOMAS Nils, VAN LAERE Pierric*

2ème année de BUT Informatique - SAE du 1er semestre

# 

# 

# Introduction

## Le sujet

Une société souhaite une refonte de son système d’information (SI) et l’automatisation

d’une partie du SI, avec principalement la création d’un site (ou application) de présentation et de

ventes de produits en ligne. Le site devra être hébergé sur des serveurs. La société sera

responsable des serveurs et du déploiement de l’application sur ces derniers. L’ensemble devra

répondre aux contraintes de sécurité, de performance et aux aspects juridiques sur la protection

des données.

## Numéro de l’équipe

eq\_02\_01

## Membres de l’équipes

OULMAS Thomas : Réflexion sur les contraintes du site

ROUX Célestin : Réflexion sur les contraintes du site

VAN LAERE Pierric : Communication client

MOREAU–THOMAS Nils : Communication client

## Le client

Le client de notre site se nomme Antoine, il possède une graveuse laser avec laquelle il grave des pièces sur mesures en bois. Mais il vend aussi des pièces uniques qu’il produit en parallèle des demandes de ses clients.

Ces pièces sont des dessous de verres, des boules de noël, des portes-clés et autres.

Il possède :

* un instagram : @antoinefactory
* une adresse mail : antoinefactory@gmail.com

# Attendues de l’application

L’application que le client nous à demandé devait :

* **Permettre le passage d’une commande**

Le client voulait qu’un utilisateur puisse commander un article, sans contraintes

* **Permettre l’ajout d’un article dans le panier**

Le client voulait qu’un utilisateur puissent ajouter un article dans son panier, sans contraintes

* **Accéder à un catalogue avec des pièces personnalisées, qui possède des sous catégories porte-clés, boule de noël …, et autres**

Le client voulait que les articles soient séparés en deux catégories (personnalisé et unique). Lors de la commande d’article personnalisé le client pourrait transmettre le motif à graver et les instructions.

Les articles personnalisés sont disponibles dans différentes catégories : Porte-clés, boule de noël, dessous de verre, autre

* **Gérer les articles**

Le client n’a pas clairement spécifier cette fonctionnalité, mais nous l’avons déclarée comme nécessaire, car sinon le catalogue d’article serait vide

* **La création de compte et l’inscription**

Le client n’a pas clairement spécifié cette fonctionnalité, mais nous l’avons déclarée comme nécessaire également, car sinon les paniers devraient être stockés dans le cache/les cookies du site, et ne serait donc disponible que pour un navigateur sur une machine. Ce qui n’est pas optimal

## Résultat obtenus

L’application actuel nous permet de :

* **Passer une commande**

Un utilisateur, s’il est connecté, est capable de commander des articles qui sont contenus dans son panier, il est important de noter que si jamais la quantité dans le panier excède celle disponible alors l’article ne sera pas commandé et un message apparaîtra.

Chaque commandes sera attribuée à un numéro de téléphone (facultatif) et une adresse (celle du client)

* **Accéder à un catalogue avec des catégories (porte-clés, boule de noël)**

Nous n’avons pas fait la catégorie pièce Unique/Personnalisée car il y avait trop de complexité pour le temps que l’on avait, nous nous somme mis d’accord avec le client pour ce changement mais nous comptons réaliser ses attentes initiales (après le rendu)

* **Un filtre qui tri les éléments du catalogue**

Nous avons réalisé un filtre qui permet de rechercher des articles pour une catégorie (Porte-clés, Boule de noël, Dessous de verre et Autre) et une méthode de tri (Meilleures ventes, Prix décroissant, Prix croissant, En promotion).

* **Une barre de recherche pour les articles du catalogue**

L’algorithme de recherche effectue une recherche dans la description du nom et du matériau de l’article. Cette requête est réalisée en SQL.

* **Gérer les articles**

L’administrateur est capable d’ajouter, d’éditer et de supprimer un article via une page du site.

Nous avons aussi implémenter un url qui exécute du SQL afin d’insérer des produits dans le catalogue (car nous faisons sans cesse des réinitialisations des tables).

Il y accède en tapant [URL de la page d’accueil]/productinsert et doit être connecté en tant qu’administrateur pour pouvoir l'exécuter.

Un article devra avoir :

* un nom
* un prix
* une promotion
* un stock disponible
* une description
* un matériau
* une épaisseur
* une largeur
* une longueur
* une image (Voir la section base de données pour plus d’informations)
* une catégorie
* un statut
* **Consultations des tables**

L’administrateur est capable de consulter les tables product (qui contient les articles) et la table commandes (qui contient les articles commandés)

* **La création de compte et l’inscription**

Un utilisateur est capable de consulter le catalogue sans compte, cependant pour faire un ajout au panier il devra se connecter/s’inscrire.

Un compte contient : un nom, un prénom, une adresse mail (Unique à chaque compte) et mot de passe.

Il peut modifier les informations suivantes : nom, prénom, mot de passe

* **L’ajout au panier**

Un utilisateur, qui est connecté, peut ajouter un article au panier, avec pour limite la quantité disponible au moment de l’achat. Ce qui veut dire que si l’article est commandé au stock maximum et qu'il reste dans le panier puis que le stock disponible descend entre-temps. Quand il va commander, la page de commande vérifiera si la quantité est toujours disponible

* **Une page de paiement**

La page de paiement est la finalisation du processus d’achat d’un article, elle s’affiche après que l’utilisateur ai passé sa commande.

Cette page demande des informations sur la carte bancaire (numéro de carte, date d’expiration, CVC), le numéro de téléphone et l’adresse de livraison.

L'adresse de livraison est capable de suggérer une adresse en utilisant une API : geoapify, cependant aux vu des circonstances d’hébergement (la VM n’as pas accès aux API externes) elle ne marchera pas.

Le numéro de téléphone respecte le format local : 01 23 45 67 89

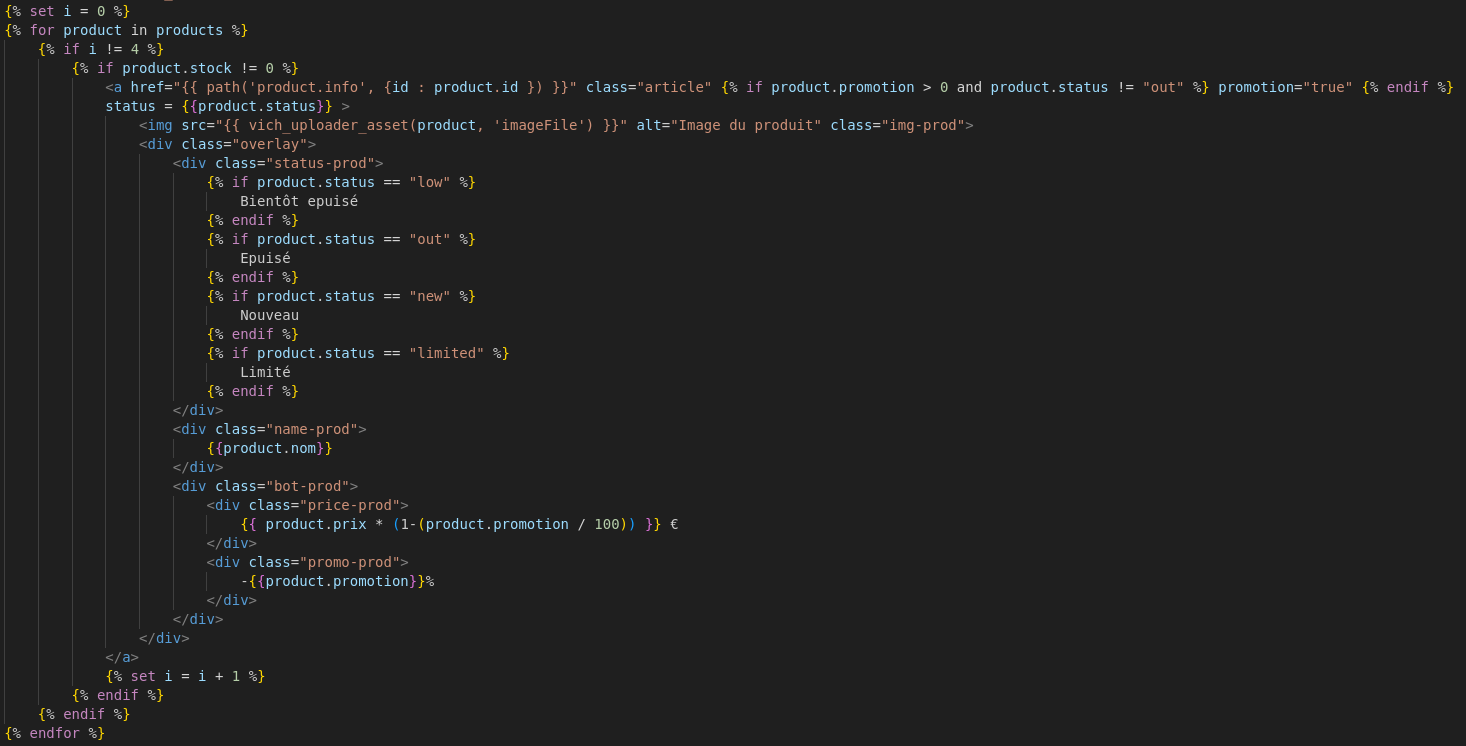
* **Le stockage d’images des articles sur le site et le lien avec la base de données**

Les images ont des formats restreints (Voir la section base de données pour plus d’informations) et sont publiables sur le site via le formulaire d’ajout d’article.

Les images ne sont donc pas dépendantes d’un hébergement externe (comme Google Drive ou UNcloud) et sont donc disponibles partout sur le site.

## Tests

Précédemment nous avions un problème car notre page d’accueil, qui affichait des articles, crashait. La raison était qu’une boucle, permettant l’ajout d’articles en vedettes sur la page d’accueil, était mal constituée. Nous avons donc modifié la boucle de manière à ce qu’elle s’effectue tant que la limite de produit que nous voulons afficher n’est pas atteinte. Ensuite nous regardons si le produit n’est pas épuisé pour l’afficher ou non.

Nous n’avons pas réussi à mettre en place les post et pré-conditions dans notre code . Mais si nous avions pu, nous aurions regardé en post-condition si la table product n’est pas vide tandis qu’en pré-condition, nous aurions regardé si la limite de produit à afficher était atteinte.

Nous aurions aussi pu simuler des interactions utilisateurs, pour faire des tests, en utilisant Cypress (<https://www.cypress.io/>) pour simuler l’interaction d’un utilisateur sur le site. Cependant pour les mêmes raisons que la monté en charge nous n’avons pas pu le faire

# L’hébergement du site

Le client nous à fait part de sa volonté d’héberger le site sur un opérateur externe pour une phase de test, et au long terme l’héberger chez lui avec ses propres moyens (à l’aide d’une raspberry). La base de données suivra le même processus d’hébergement.

# La base de données

## Le SGBD

Le SGBD est lié par le framework Symfony, à l’aide du package Doctrine et ORM, qui gère ces entités. Nous n'avons pas explicitements programmé le lien avec les tables, nous ne faisons qu'utiliser des méthodes déjà disponibles pour une instance d’un objet (étant lié à une ligne de la base de données)

## Structure des tables

dire type, limite de char, Mocodo

Voici les colonnes de chaques tables :

[Img tables (mariadb show)]

[bla bla VARCHAR = string jsp]  
  
Voici le schéma relationnel entre chaque table :

[MOCODO]

## Contraintes

La colonne email dans la table user est UNIQUE, ce qui signifie qu’une adresse mail n’existe que dans une seule ou aucune ligne.

Les clé primaires de chaque table sont des id en AUTO-INCREMENT, c’est de l’auto-incrémentation, ce qui signifie que l’id est UNIQUE et augmente à chaque ajout de ligne.

Ces auto-incrémentations sont toutes indépendantes entre elle

Exemple :

On ajoute un article, son id devient alors 1

On ajout un 2ème article, son id est alors 2 grâce à l’auto-incrémentation

Et ainsi de suite.

La table product référence en clé étrangère l’id d’un article (table product) et l’id d’un utilisateur (table user). la table commande fait là même chose

## Procédures

### Note

Notre site utilise mariaDB.

Il est important de noter que ces procédures ont été écrites à l’aide d’une documentation en ligne car la syntaxe diffère d’Oracle et certaines choses (comme le “LOOP”) n’ont pas été vu en cours.

### Suppression d’un utilisateur

| /\*procédure n1\*/  CREATE OR REPLACE PROCEDURE supprimer\_user(IN user\_id INT)  BEGIN  DECLARE cnt INT;  SELECT COUNT(\*) INTO cnt FROM commande WHERE id\_user\_id = user\_id;  IF cnt > 0 THEN  SIGNAL SQLSTATE '45001'  SET MESSAGE\_TEXT = 'L utilisateur possede des commandes en cours';  ELSE  DELETE FROM panier WHERE id\_user\_id = user\_id;  DELETE FROM commande WHERE id\_user\_id = user\_id;  DELETE FROM user WHERE id = user\_id;  SELECT 'L utilisateur est supprimé';  END IF;  END;  /\*insert test1\*/  INSERT INTO user VALUES(999,'user@test.fr ','["ROLE\_USER"]','$2y$13$Km3pXs/NW6NFMr9JrdqZHOKij1SQJyhRctUECm2RAOEgZJa1AUPYW','Letest','Utilisateur');  /\*test1\*/  PREPARE stmt1 FROM 'CALL supprimer\_user(?)';  SET @a = 999;  EXECUTE stmt1 USING @a; |
| --- |

Cette procédure fonctionne comme ceci :

Quand un utilisateur est supprimer

* si jamais il est dans la table commande, alors la suppression ne se fait pas
* sinon toutes les lignes contenant son id sont supprimer

### Suppression d’un articles

| /\*procédure n2\*/  CREATE OR REPLACE PROCEDURE supprimer\_article(IN product\_id INT)  BEGIN  DECLARE cnt INT;  SELECT COUNT(\*) INTO cnt FROM commande WHERE id\_product\_id = product\_id;  IF cnt > 0 THEN  SIGNAL SQLSTATE '45002'  SET MESSAGE\_TEXT = 'L article est contenu dans des commandes en cours';  ELSE  DELETE FROM panier WHERE id\_product\_id = product\_id;  DELETE FROM commande WHERE id\_product\_id = product\_id;  DELETE FROM product WHERE id = product\_id;  SELECT 'L article est supprimé';  END IF;  END;  /\*insert test2\*/  INSERT INTO product VALUES(999,'Test',54.0,2,72,1,'Un test','Bois',84.0,36.0,77.0,'image.jpg','Porte-clé','low');  /\*test2\*/  PREPARE stmt2 FROM 'CALL supprimer\_article(?)';  SET @a = 999;  EXECUTE stmt2 USING @a; |
| --- |

Cette procédure fonctionne comme ceci :

Quand un article est supprimer

* si jamais il est dans la table commande, alors la suppression ne se fait pas
* sinon toutes les lignes contenant son id sont supprimer

### Ajout d’un article dans panier

| CREATE OR REPLACE PROCEDURE ajout\_panier(IN p\_product\_id INT, IN p\_user\_id INT, IN p\_quantite INT)  BEGIN  DECLARE pan\_item\_count INT;  /\*regarde si l article est déjà dans le panier de user\*/  SELECT COUNT(\*) INTO pan\_item\_count FROM panier WHERE id\_product\_id = p\_product\_id AND id\_user\_id = p\_user\_id;  IF pan\_item\_count > 0 THEN  /\*l article est déjà dans le panier de user\*/  UPDATE panier  SET quantite = quantite + p\_quantite  WHERE id\_product\_id = p\_product\_id AND id\_user\_id = p\_user\_id;  ELSE  /\*l article n’est pas déjà dans le panier de user\*/  INSERT INTO panier (id\_product\_id, id\_user\_id, quantite)  VALUES (p\_product\_id, p\_user\_id, p\_quantite );  END IF;  END;  /\*insert test3\*/  INSERT INTO product VALUES(999,'Test',54.0,2,72,1,'Un test','Bois',84.0,36.0,77.0,'image.jpg','Porte-clé','low');  INSERT INTO user VALUES(999,'user@test.fr ','["ROLE\_USER"]','$2y$13$Km3pXs/NW6NFMr9JrdqZHOKij1SQJyhRctUECm2RAOEgZJa1AUPYW','Letest','Utilisateur');  /\*test3\*/  PREPARE stmt3 FROM 'CALL ajout\_panier(?,?,?)';  SET @a = 999;  SET @b = 999;  SET @c = 5;  EXECUTE stmt3 USING @a,@b,@c; |
| --- |

Cette procédure fonctionne comme ceci :

Quand un utilisateur ajoute au panier un article

* si jamais l’article est déjà dans son panier, alors la quantité dans le panier sera celle déjà dans le panier + celle que l'on ajoute
* sinon une ligne se crée avec l’article et l’utilisateur qui l’a commandé

### Passage de la commande d’un utilisateur

| CREATE PROCEDURE passage\_commande(IN p\_user\_id INT, IN adr VARCHAR, IN tel VARCHAR)  BEGIN  DECLARE product\_id, pan\_quantite, available\_stock INT;  DECLARE done BOOLEAN DEFAULT FALSE;  DECLARE pan\_cursor CURSOR FOR  SELECT p.id\_product\_id, p.quantite, pr.stock  FROM panier p  JOIN product pr ON p.id\_product\_id = pr.id  WHERE p.id\_user\_id = p\_user\_id;  DECLARE CONTINUE HANDLER FOR NOT FOUND  SET done = TRUE;  OPEN pan\_cursor;  pan\_loop: LOOP  FETCH pan\_cursor INTO product\_id, pan\_quantite, available\_stock;  IF done THEN  LEAVE pan\_loop;  END IF;  IF pan\_quantite > available\_stock THEN  SIGNAL SQLSTATE '45003' SET MESSAGE\_TEXT = 'La quantité commandée n est pas en stock';  ELSE  -- Ajout dans la table commande  INSERT INTO commande (id\_user\_id, id\_product\_id, adresse, telephone, quantite)  VALUES (p\_user\_id, product\_id, adr, tel, pan\_quantite);    -- Retire de la table panier  DELETE FROM panier WHERE p\_user\_id = p\_user\_id AND p\_product\_id=product\_id and;  -- Update product pour montrer la vente  UPDATE product  SET stock = stock - pan\_quantite,  nb\_ventes = nb\_ventes + pan\_quantite  WHERE id = product\_id;  END IF;  END LOOP;  CLOSE pan\_cursor;  END;  /\*insert test4\*/  INSERT INTO product VALUES(999,'Test',54.0,2,72,1,'Un test','Bois',84.0,36.0,77.0,'image.jpg','Porte-clé','low');  INSERT INTO user VALUES(999,'user@test.fr ','["ROLE\_USER"]','$2y$13$Km3pXs/NW6NFMr9JrdqZHOKij1SQJyhRctUECm2RAOEgZJa1AUPYW','Letest','Utilisateur');  PREPARE stmt4 FROM 'CALL ajout\_panier(?,?,?)';  SET @a = 999;  SET @b = 999;  SET @c = 5;  EXECUTE stmt4 USING @a,@b,@c;  /\*test4\*/  PREPARE stmt5 FROM 'CALL passage\_commande(?)';  SET @d = 999;  EXECUTE stmt5 USING @d; |
| --- |

Cette procédure fonctionne comme ceci :

Quand un utilisateur passe un commande, on regarde chaque article de son panier

* si jamais la quantité dans le panier excède celle disponible alors l’article n’est pas commandé
* sinon il est commandé et le nombre de vente augmente + le stock disponible diminue

## Les triggers

### La mise à jour du statut d’un article

Nous avons réalisé deux triggers (pour l’update et l’insert) pour le changement dynamique du statut d’un article :

| CREATE OR REPLACE TRIGGER update\_product\_status\_before\_insert  BEFORE INSERT ON product  FOR EACH ROW  BEGIN    IF NEW.stock > 0 AND NEW.stock < 11 THEN  SET NEW.status = 'low';  ELSEIF NEW.stock = 0 THEN  SET NEW.status = 'out';  ELSEIF (NEW.status = 'out' OR NEW.status='low') AND NEW.stock > 10 THEN  SET NEW.status = '';  END IF;  END;  CREATE OR REPLACE TRIGGER update\_product\_status\_before\_update  BEFORE UPDATE ON product  FOR EACH ROW  BEGIN  IF NEW.stock > 0 AND NEW.stock < 11 THEN  SET NEW.status = 'low';  ELSEIF NEW.stock = 0 THEN  SET NEW.status = 'out';  ELSEIF (OLD.status = 'out' OR OLD.status='low') AND NEW.stock > 10 THEN  SET NEW.status = '';  END IF;  END;  /\*insert test5\*/  INSERT INTO product VALUES(999,'Test',54.0,2,72,0,'Un test','Bois',84.0,36.0,77.0,'image.jpg','Porte-clé','low');  /\*le statut doit être out\*/  /\*update test5\*/  UPDATE INTO product SET stock=5 WHERE id = 999  /\*le statut doit être low\*/ |
| --- |

Le trigger en cas d’update d’un article fonctionne comme ceci :

On regarde chaque ligne qui sera modifié

* si le stock est entre 0 et 10, le statut devient “low” (bientôt épuisé)
* sinon si le stock est à 0, le statut devient “out” (épuisé)
* sinon si l’article était épuisé ou bientôt épuisé et que le nouveau stock est supérieur à 10, alors son statut est normal (soit “”)

Le trigger en cas d’insert d’un article fonctionne comme ceci :

On regarde chaque ligne qui sera modifié

* si le stock est entre 0 et 10, le statut devient “low” (bientôt épuisé)
* sinon si le stock est à 0, le statut devient “out” (épuisé)
* sinon si l’article est épuisé ou bientôt épuisé et que le stock est supérieur à 10, alors son statut est normal (soit “”)

Le différences entre les deux triggers c’est que l’insert ne prend pas en compte “l’ancien statut” mais celui actuelle, au cas où le client crée un produit déclaré comme épuisé mais avec du stock, alors le trigger corrige son erreur

## Sécurité

Nous avions prévu de faire une structure de base de données ou il y avait une table distincte pour les utilisateurs et les administrateurs. Sauf que le framework Symfony de permet de gérer qu’une table, nous avons donc fait une table regroupant utilisateurs et administrateurs en les distinguant par un rôle.

C’est problématique car si la table est hacké, le hacker disposera de tous les comptes, admin et user mais c’était malheureusement là seul manière de faire l’utilisation de compte

Les mots de passes sont haché afin de garantir la sécurité si jamais la table est hacké

## L’image d’un article

Une image par article pourra être publiée dans la base de données. Cette image sera limitée à 4 formats : WEBP, PNG, JPG et JPEG. JPG et JPEG étant les principaux formats WEBP et PNG si le client souhaite uploader des images transparentes (cependant le WEBP et PNG prennent plus de place que les deux autres formats).

Ces images seront stockées sous formes de vignettes afin de minimiser l’espace de stockage occupé.

Ces images permettent d’assurer la durée dans le temps, sauf si le site est hors-ligne, auquel cas les articles ne sont affichés et donc que nos images ne soit pas disponibles dans cette situation n’est pas un soucis

# Le coût de l’application (temps + mémoire)

## Les mesures

Pour les mesures, nous avons créé un tableau afin de faire 10 tests individuels et obtenir une moyenne. Nous avons effectués ces test sur le site hébergé localement et sans test en simultanées, la raison étant que la VM ne disposais pas de l’outils de mesure de symfony (il n’est disponible que dans le serveur de développement) et que les limitations techniques (et de temps) nous empêchait de faire des tests en simultané sur un serveur local.

Voici les situations qui ont été mesurés :

* L’affichage du catalogue (56 articles au total)
* Après le connexion d’un compte
* Après l’inscription d’un compte
* L’Ajout d’un article dans le panier
* Le tri des articles avec le statut “en promotion” et la catégorie “porte-clé” (28 articles résultant du tri)

Nous avons mesuré le temps (en ms) et la mémoire (en Mo) que le système utilisé pour l’affichage de la situation et les voici :

| **L’affichage du catalogue** | | | | | |
| --- | --- | --- | --- | --- | --- |
|
| Temps Moyen : | 74,4 | Temps Minimum : | 20 | Temps Maximum : | 446 |
| Mémoire Moyenne : | 5,8 | Mémoire Minimum : | 4 | Mémoire Maximum : | 6 |
| **Après le connexion d’un compte** | | | | | |
|
| Temps Moyen : | 141,7 | Temps Minimum : | 55 | Temps Maximum : | 472 |
| Mémoire Moyenne : | 4 | Mémoire Minimum : | 4 | Mémoire Maximum : | 4 |
| **Après l’inscription d’un compte** | | | | | |
|
| Temps Moyen : | 17,4 | Temps Minimum : | 11 | Temps Maximum : | 35 |
| Mémoire Moyenne : | 6 | Mémoire Minimum : | 6 | Mémoire Maximum : | 6 |
| **L’Ajout d’un article dans le panier** | | | | | |
|
| Temps Moyen : | 18,5 | Temps Minimum : | 15 | Temps Maximum : | 25 |
| Mémoire Moyenne : | 4 | Mémoire Minimum : | 4 | Mémoire Maximum : | 4 |
| **Le tri des articles** | | | | | |
|
| Temps Moyen : | 34,8 | Temps Minimum : | 13 | Temps Maximum : | 54 |
| Mémoire Moyenne : | 4 | Mémoire Minimum : | 4 | Mémoire Maximum : | 4 |

# Conclusion

L'application a atteint un certain niveau de fonctionnalité, mais il reste des points à améliorer et des fonctionnalités à implémenter.

Les tests et l'hébergement nécessitent une attention particulière pour garantir la qualité et la stabilité de l'application.

La sécurité des données doit être renforcée, en envisageant des solutions pour différencier les utilisateurs et administrateurs dans le système.

Pour le projet, nous aurions pu améliorer plusieurs choses :

* Mettre une colonne enregistrant la date de création d’un compte, pour le supprimer de la base de données 4 ans précédant cette date de création afin de respecter le RGPD.
* Les tests, nous aurions pu effectuer un test de montée en charge (comme un DDoS) mais cela n’a pas été concrétisé par manque de temps et de connaissances.
* Les articles personnalisés et uniques, même si nous avons trouvé un solution temporaire en accord avec le client, l'engagement initial doit quand même être compléter
* L’algorithme de recherche, qui pourrait être plus efficace si n’avions pas utilisé de requêtes SQL, mais sans assertions en php cela n’était pas possible.