3291.2 JEE-Spring I – Rapport – ISC3il-b

|  |
| --- |
| **WalkingFish** |

|  |
| --- |
| Étudiants participant à ce travail : **AUBERT Nicolas, ISC2il-b**  Présenté à :  **CHÈVRE Sébastien**  Restitution du rapport : **30.01.2023**  Période : **2022 – 2023**  École : **HE-Arc, Neuchâtel** |

**Site vitrine pour la marque de vêtements WalkingFish**

Table des matières

[1 - Introduction I](#_Toc125558798)

[1.1 - Contexte I](#_Toc125558799)

[1.2 - But du projet I](#_Toc125558800)

[1.3 - Objectifs principaux I](#_Toc125558801)

[1.4 - Objectifs secondaires I](#_Toc125558802)

[1.5 - Livrables attendus I](#_Toc125558803)

[2 - Analyse III](#_Toc125558804)

[3 - Conception IV](#_Toc125558805)

[3.1 - Schéma d’interactions (Use Case) IV](#_Toc125558806)

[3.2 - Architecture IV](#_Toc125558807)

[3.2.1 - MCD IV](#_Toc125558808)

[3.2.2 - MLD V](#_Toc125558809)

[3.2.3 - Schéma des routes V](#_Toc125558810)

[3.3 - Planification VI](#_Toc125558811)

[4 - Réalisation 7](#_Toc125558812)

[4.1 - Les modèles 7](#_Toc125558813)

[4.1.1 - Article 7](#_Toc125558814)

[4.1.2 - Picture 7](#_Toc125558815)

[4.2 - Les contrôleurs 7](#_Toc125558816)

[4.2.1 - IndexController 7](#_Toc125558817)

[4.2.2 - ArticleController 7](#_Toc125558818)

[4.2.3 - AdminController 7](#_Toc125558819)

[4.3 - Les repository 7](#_Toc125558820)

[4.3.1 - ArticleRepository 7](#_Toc125558821)

[4.3.2 - PictureRepository 7](#_Toc125558822)

[4.4 - Les services 7](#_Toc125558823)

[4.4.1 - CatalogService 7](#_Toc125558824)

[4.4.2 - FileStorageService 7](#_Toc125558825)

[4.5 - L’accès administrateur 8](#_Toc125558826)

[4.5.1 - Connexion 8](#_Toc125558827)

[4.5.2 - Déconnexion 8](#_Toc125558828)

[4.6 - CI/CD 8](#_Toc125558829)

[4.6.1 - Tests unitaires 8](#_Toc125558830)

[5 - Résultats 10](#_Toc125558831)

[5.1 - Catalogue 10](#_Toc125558832)

[5.1.1 - Liste des articles 10](#_Toc125558833)

[5.1.2 - Pagination 10](#_Toc125558834)

[5.1.3 - Affichage des détails d’un article 10](#_Toc125558835)

[5.2 - Gestion des articles 10](#_Toc125558836)

[5.2.1 - Connexion / Déconnexion 10](#_Toc125558837)

[5.2.2 - Ajout 10](#_Toc125558838)

[5.2.3 - Modification 10](#_Toc125558839)

[5.2.4 - Suppression 10](#_Toc125558840)

[5.3 - Gestion des images d’un article 10](#_Toc125558841)

[5.3.1 - Ajout 10](#_Toc125558842)

[5.3.2 - Suppression 10](#_Toc125558843)

[6 - Limitations et perspectives 11](#_Toc125558844)

[6.1 - Ajout de nouvelles fonctionnalités 11](#_Toc125558845)

[6.2 - Amélioration de l’UI / UX 11](#_Toc125558846)

[7 - Conclusion 12](#_Toc125558847)

[8 - Annexes I](#_Toc125558848)

[8.1 - Guide utilisateur I](#_Toc125558849)

[8.2 - Guide développeur I](#_Toc125558850)

[8.2.1 - Mise en place de la base données I](#_Toc125558851)

[8.2.2 - Lancement de l’application I](#_Toc125558852)

[8.2.3 - Lancement des tests I](#_Toc125558853)

[8.3 - Table des illustrations II](#_Toc125558854)

[8.4 - Bibliographies et références III](#_Toc125558855)

[8.4.1 - Sites Web III](#_Toc125558856)

[8.4.2 - Livres III](#_Toc125558857)

[8.4.3 - Autres III](#_Toc125558858)

1. Introduction
   1. Contexte

Walking Fish est une marque de vêtements créée par deux anciens étudiants de la HEP-Bejune, à la Chaux-de-Fonds.

Afin d'améliorer la visibilité de leur entreprise, je me suis proposé afin de réaliser une application web vitrine, représentant la marque et listant les différents articles disponibles.

* 1. But du projet

Créer une application web vitrine à l'aide du framework Spring, sur laquelle les utilisateurs pourront accéder aux articles proposés par la marque Walking Fish.

Les personnes possédant un accès administrateur pourront se connecter afin de modérer les articles (ajout, modification, suppression).

* 1. Objectifs principaux
* Lister les différents articles sur la page de vente (données simulées), avec une pagination permettant de limiter le nombre d’articles affichés sur une page
  + Un article est composé d'un nom, une description, un prix et une image,
  + Les images seront stockées et récupérées via une API (imgur par exemple)
* Ajouter le CRUD sur les articles,
* Ajouter la connexion / déconnexion pour les administrateurs, afin de sécuriser l’accès au CRUD des articles.
  1. Objectifs secondaires
* Nouvelle fonctionnalité grâce à laquelle un administrateur pourra ajouter de multiples images pour un seul article,
* Les administrateurs pourront ajouter les tailles disponibles pour les articles,
* Les administrateurs pourront ajouter les coloris disponibles pour les articles.
  1. Livrables attendus
* Cahier des charges
  + Planning initial
* Rapport de travail
  + Planning effectif
* Code source

1. Analyse

A supprimer je pense

1. Conception
   1. Schéma d’interactions (Use Case)

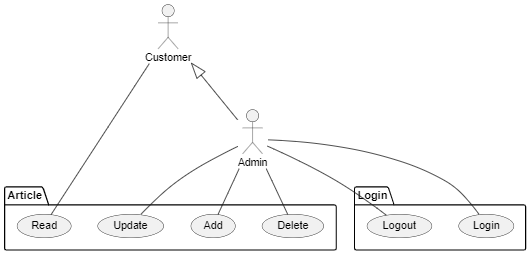


Figure - Schéma d'interactions (Use Case)

* 1. Architecture
     1. MCD

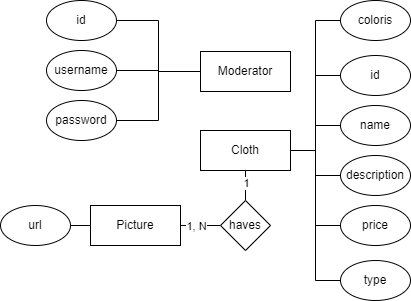


Figure - MCD (Modèle Conceptuel de Données)

* + 1. MLD

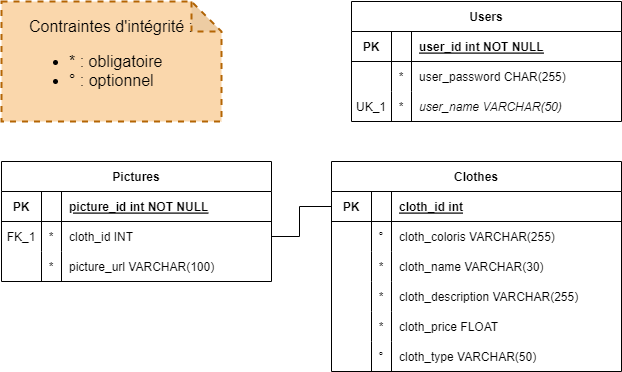


Figure - MLD (Modèle Logique de Données)

* + 1. Schéma des routes

Une image contenant table

Description générée automatiquement

Figure - Schémas des routes

* 1. Planification

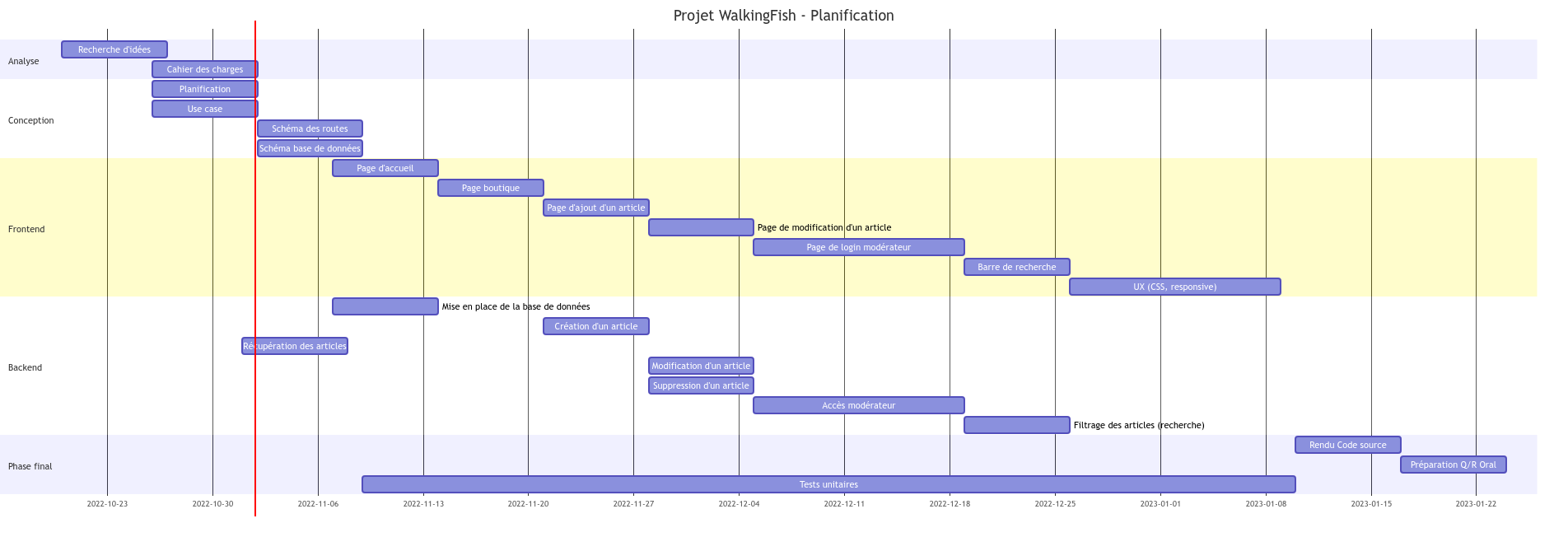


Figure – Planification

1. Réalisation
   1. Les modèles
      1. Article

Ce modèle représente une classe "Article" qui est utilisée pour stocker des informations sur un article d'un catalogue. Il contient des champs tels que le nom, la description, le prix et le type de l'article. Il possède également une liste d'images liées à cet article. Il contient des méthodes pour accéder et mettre à jour ces informations ainsi que des méthodes pour tester l'égalité entre les objets Article.

* + 1. Picture

Ce modèle représente une classe "Picture" qui est utilisée pour stocker des informations sur les images associées à un article d'un catalogue. Il contient des champs tels que l'identifiant, l'url et le nom de l'image. Il possède également une relation Many-To-One avec la classe Article pour stocker l'article lié à cette image. Il contient des méthodes pour accéder et mettre à jour ces informations ainsi que des méthodes pour tester l'égalité entre les objets Picture.

* 1. Les contrôleurs

Ce projet possède trois contrôleurs distincts :

* *IndexController*, utilisé pour afficher la page d’accueil
* *ArticleController*, utilisé principalement pour l’affichage des articles
  + */catalogue,*
  + */catalogue/show/{id}*, où {id} représente l’identifiant unique de l’article à afficher
* *AdminController*, utilisé pour la gestion des articles (CRUD)
  + */admin/create*
  + */admin/edit/{id}*
  + */admin/show/{id}*
  + */admin/delete/{id}*
  1. Les repository
     1. ArticleRepository
     2. PictureRepository
  2. Les services
     1. CatalogService
     2. FileStorageService
  3. L’accès administrateur
     1. Connexion
     2. Déconnexion
  4. CI/CD

CI/CD (ou "Intégration continue et déploiement continu") est une pratique de développement logiciel qui consiste à automatiser le processus de construction, de test et de déploiement d’une application. Cela permet de détecter les erreurs plus tôt dans le processus de développement et de les corriger plus rapidement.

Pour mettre en place le CI/CD pour ce projet, un pipeline utilisant GitHub Actions et Maven a été mis en place sur le repository GitHub. Il décrit les étapes à suivre lorsque du code est poussé sur la branche principale.

Les étapes incluent :

* Cloner le dépôt de code
* Configurer une version de JDK
* Construire le projet avec Maven
* Tester le projet avec Maven (lancer les tests unitaires)
  + 1. Tests unitaires

Afin d’assurer le bon fonctionnement de l’application, une batterie de tests unitaires a été mise en place. Ils s’occupent de tester, par fragment, le service *CatalogService*.

Les tests utilisent la bibliothèque JUnit Jupiter pour les assertions et Spring Boot Test pour l'injection de dépendances.

Les tests vérifient les différentes fonctionnalités du service, telles que :

* L'injection de dépendances pour le dépôt d'articles et le service de catalogue,
* La récupération de tous les articles du catalogue,
* L’ajout d’un article dans le catalogue,
* La récupération d'un article par son identifiant,
* La mise à jour d'un article,
* La suppression d'un article,
* L’enregistrement d'une image pour un article.

Ces tests ne couvrent pas tous les cas d'utilisation possibles. D'autres scénarios pourraient être créés afin de s'assurer que le service de catalogue fonctionne correctement. Plus d’informations sur les tests unitaires se trouvent dans les annexes.

* 1. Problèmes rencontrés
     1. Suppression en cascade
     2. Stockage des images sur le serveur
     3. Accès aux images sur le serveur

1. Résultats
   1. Catalogue

Pour les exemples suivants, le nombre d’articles affichés par pages a été limité à trois, afin d’augmenter la lisibilité des captures. Cependant, ce nombre s’élève à six.

* + 1. Liste des articles

Comme illustré sur la « Figure 6 - Résultats : Catalogue », les articles disponibles sont listés sous forme de cartes. Chaque carte possède un carrousel, permettant de feuilleter les différentes images représentant l’article. Sous ces images se trouve une zone, dans laquelle différentes informations sont affichées, telles que la description de l’article, son prix et également son type.



Figure - Résultats : Catalogue

* + 1. Pagination

Toujours en se référant à la « Figure 6 - Résultats : Catalogue », en bas de la page se trouvent des numéros, correspondant au numéro de page du catalogue Le numéro courant est affiché en beige. L’utilisateur peut alors naviguer entre les différentes pages afin de consulter tous les articles.

* 1. Gestion des articles

Afin de modérer les articles disponibles, une zone administrateur a été mise en place.

* + 1. Connexion

Afin d’accéder à cette zone restreinte, un modérateur doit d’abord se connecter. Pour cela, il peut se rendre sur la route suivante : « /admin ». S’il n’est pas connecté, il sera redirigé sur une page de connexion, comme illustré sur la « Figure 7 - Résultats : Connexion ».

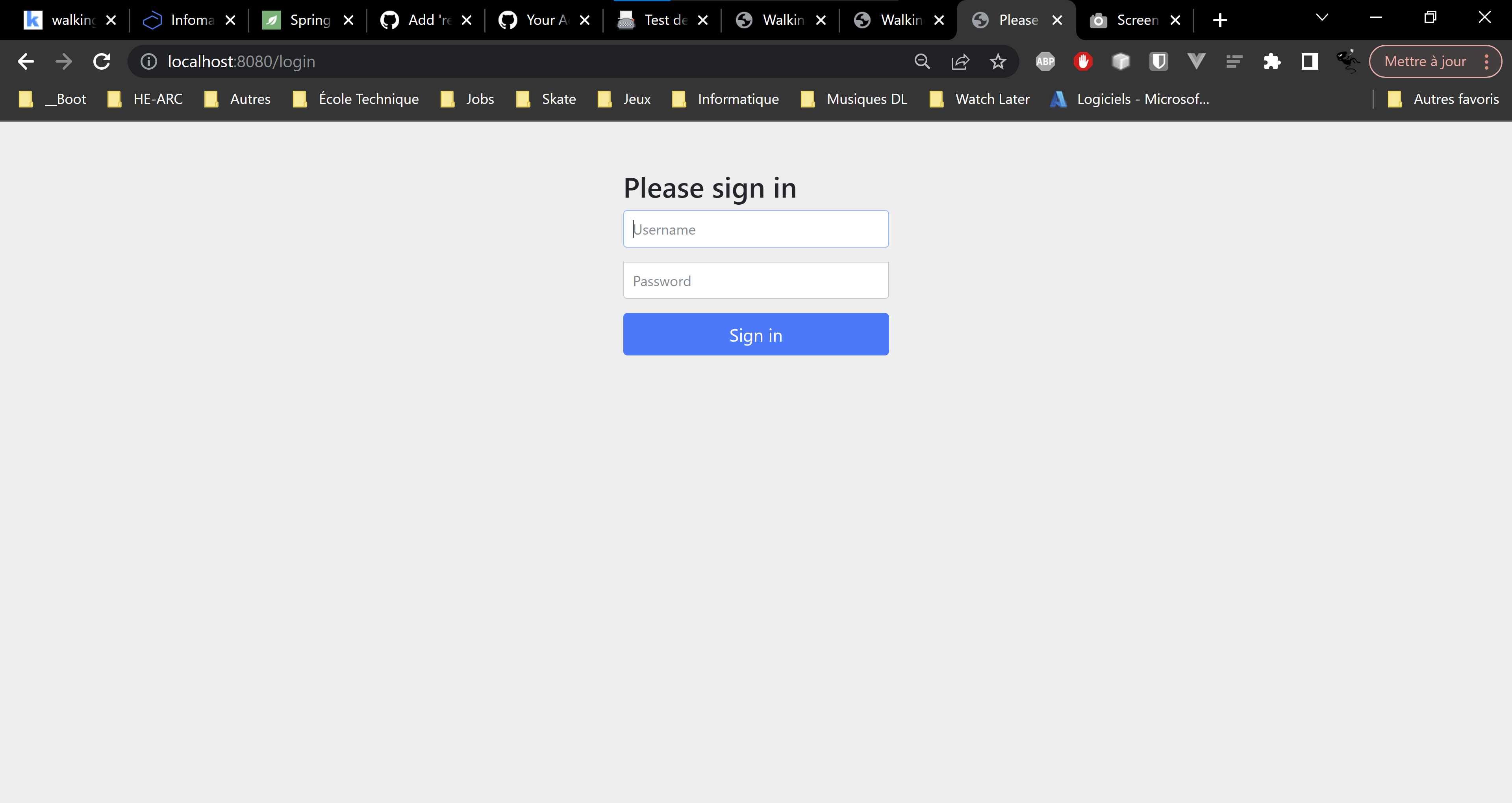


Figure - Résultats : Connexion

Une fois connecté, le modérateur aura accès à la gestion des articles et leurs images. Cela commence par une modification de barre de navigation. Celle-ci possèdera, en plus des anciens champs, deux nouveaux boutons : « Se déconnecter » en rouge, et « Ajouter un article » en vert.

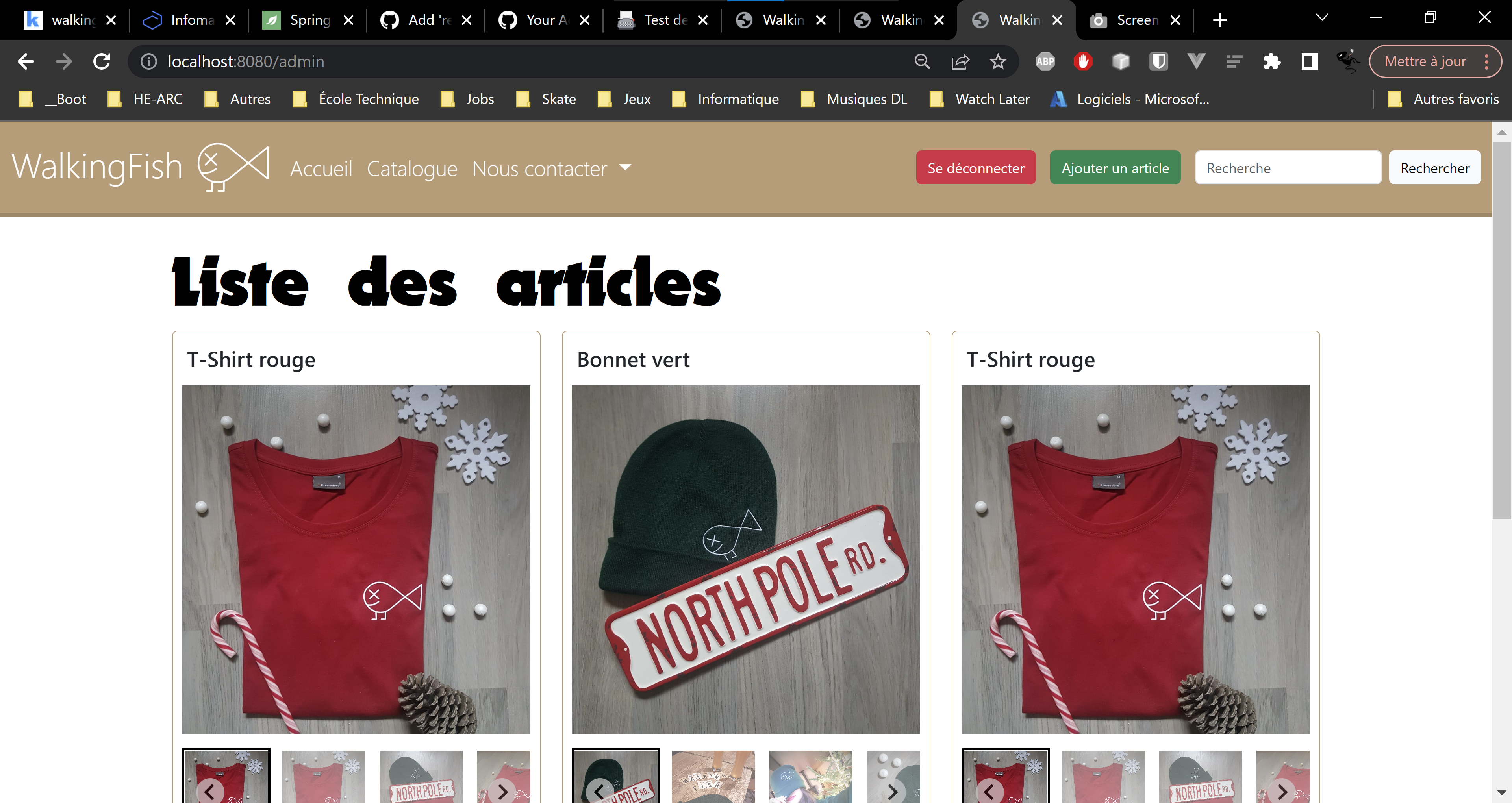


Figure - Résultats - Barre de navigation admin

* + 1. Déconnexion

Pour se déconnecter, un utilisateur administrateur peut simplement cliquer le bouton rouge « Se déconnecter » se trouvant dans la barre de navigation. Il sera alors redirigé sur la page catalogue.

* + 1. Ajout
    2. Modification
    3. Suppression
  1. Gestion des images d’un article
     1. Ajout
     2. Suppression

1. Limitations et perspectives
   1. Ajout de nouvelles fonctionnalités

Comme précisé dans le cahier des charges, certaines fonctionnalités n’ont pas été implémentées. Par exemple, il pourrait être judicieux d’ajouter la gestion des couleurs et tailles disponibles pour un article.

* 1. Amélioration de l’UI / UX

Les différentes interfaces utilisateurs, bien que responsives, manquent d’originalité et de personnalité. En effet, celles-ci ont été en grande partie réalisées à l’aide du Framework Bootstrap, afin de beaucoup du temps.

1. Conclusion

En fonction de la longueur des premières pages du rapport (avant la table des matières), il se peut que le numéro de la dernière page courante (normalement celle-ci) ne corresponde pas au nombre de pages total.

Il faut alors modifier ce dernier champ (numéro de page total, à droite du « / »).

* Placer la souris sur ce champ, qui sera alors grisé
* Appuyer les touches SHIFT + F9 pour afficher le code du champ
* Modifier le nombre à soustraire au nombre de pages total (à droite du signe « - »)
* Appuyer sur les touches SHIFT + F9 pour rebasculer en mode normal

1. Annexes
   1. Guide utilisateur
   2. Guide développeur
      1. Mise en place de la base données
      2. Lancement de l’application
      3. Lancement des tests
   3. Table des illustrations

[Figure 1 - Schéma d'interactions (Use Case) IV](#_Toc125721613)

[Figure 2 - MCD (Modèle Conceptuel de Données) IV](#_Toc125721614)

[Figure 3 - MLD (Modèle Logique de Données) V](#_Toc125721615)

[Figure 4 - Schémas des routes V](#_Toc125721616)

[Figure 5 – Planification VI](#_Toc125721617)

[Figure 6 - Résultats : Catalogue 10](#_Toc125721618)

[Figure 7 - Résultats : Connexion 11](#_Toc125721619)

[Figure 8 - Résultats - Barre de navigation admin 11](#_Toc125721620)

* 1. Test unitaires

|  |  |
| --- | --- |
| Nom | Description |
| testInjectedComponentsAreNotNull | Vérifie que les composants injectés (ArticleRepository et CatalogService) ne sont pas null |
| testGetAllArticles | Vérifie que la méthode getAllArticlesFromCatalog() renvoie une liste d'articles non vide |
| testAddArticleToCatalogue | Vérifie que la méthode addArticleToCatalog() ajoute correctement un article à la base de données |
| testGetArticleById | Vérifie que la méthode getArticleById() renvoie le bon article en fonction de l'ID donné |
| testUpdateArticleInDB | Vérifie que la méthode updateArticleInDB() met à jour correctement un article dans la base de données |
| testDeleteArticleInDB | Vérifie que la méthode deleteArticleInDB() supprime correctement un article de la base de données |
| testSavePicture | Vérifie que la méthode savePicture() sauvegarde correctement une image liée à un article dans la base de données |

* 1. Bibliographies et références
     1. Sites Web