

TPI - Echange de cartes à collectionner

David Assayah – FIN2 - 2023

Table des matières

1	Analyse préliminaire	4
1.1	Introduction.....	4
1.2	Objectifs	4
	Matériel à disposition	5
1.3	Prérequis.....	5
1.4	Contenu livrable	5
2	Analyse / Conception	5
2.1	Méthodologie de travail.....	5
2.2	Environnement de travail	6
2.3	Maquettes.....	6
2.4	Base de données	16
2.4.1	MCD.....	16
2.4.2	Entités.....	16
2.4.3	Cardinalités	19
2.4.4	MLD	21
2.5	Stratégie de test	22
2.5.1	Logiciels et outils supplémentaires.....	24
2.6	Risques techniques.....	24
2.6.1	Mise en place de l'environnement Docker.....	24
3	Réalisation.....	25
	INTRO + VERSIONS OUTILS	25
3.1	25
3.2	Base de données	25
3.2.1	Importation du fichier .SQL.....	25
3.2.2	MPD	27
3.2.3	Tables et type des attributs	27
3.3	31
4	Tests.....	32
4.1.1	32
4.1.2	32
4.1.3	32
4.2	Erreurs restantes	32
4.3	Liste des documents fournis	32
5	Conclusions.....	33
5.1	Bilan des fonctionnalités	33
5.2	Comparaison de la planification.....	33
5.3	Critiques / Finalité du projet	33
5.4	Difficultés particulières	33

5.5	Conclusion personnelle	33
6	Lexique	33
7	Table d'illustrations	33
8	Annexes.....	34
8.1	Résumé du rapport du TPI / version succincte de la documentation..	34
8.1.1	Situation de départ	34
8.1.2	Mise en œuvre	34
8.1.3	Résultats.....	34
8.2	Sources – Bibliographie	34
9	Bibliographie	34
9.1	Manuel d'Utilisation	34
9.2	Archives du projet	34

1 Analyse préliminaire

1.1 Introduction

Ce TPI est réalisé sous la supervision de M.Charmier – Chef de projet – et de M.Venries ainsi que M.Bertino – Experts – dans le cadre de la formation FPA de l'ETML.

Le projet a pour but de produire une application WEB permettant à des collectionneurs de cartes d'échanger celles qu'ils ont à double. Un système de transaction de points de crédits – valeur accordée à chaque carte – doit être mis en place afin de fonctionner comme monnaie d'échange entre acheteurs et vendeurs. Lors de l'achat d'une carte, les crédits dépensés par l'acheteur doivent être mis en attente de la confirmation de la bonne réception des articles. Dès que cela se produit, les crédits sont versés sur le compte du vendeur.

Les utilisateurs doivent pouvoir ajouter/modifier/supprimer leurs propres cartes et consulter/acheter celles des autres. Le tout en ayant la possibilité de filtrer ou trier les cartes mises en vente sur le site.

1.2 Objectifs

1. Les formulaires de l'application respectent les bonnes pratiques, à savoir :
 1. Affichage des erreurs contextuelles.
 2. Re-remplissage des champs lors d'erreur.
 3. Validation des champs.
2. La recherche multicritère possède des fonctionnalités de tris et de filtres pertinentes
3. Le candidat explique dans le rapport au moins trois mesures de sécurité qu'il a implémentées dans son projet.
4. La gestion de l'authentification et des rôles des utilisateurs garantit un accès sécurisé aux données.
5. Le système de mise en attente des points de crédits lors d'une opération d'achat/de vente de carte assure une gestion cohérente des points de crédits des utilisateurs.
6. Le manuel d'installation est présent sous forme de README.md dans le dépôt git. Evalué selon la reproductibilité et la qualité des instructions pour l'environnement de travail du candidat.
7. La modélisation de la base de données respecte les conventions de nommage de l'ETML et le MCD / MLD / MPD sont présents et détaillés (justification des différents choix).

Matériel à disposition

- Un PC standard de l'ETML (Windows 10).
- Visual Studio code avec environnement PHP installé.
- Serveur web local (uWamp ou autre).
- Suite Microsoft Office pour la documentation.
- Un dépôt GIT (GitHub, BitBucket ou autre).

1.3 Prérequis

- Connaissances en programmation PHP et en POO (Modules ICT 403, 404, 226, 120, 326, 411, 133 et 151).
- Connaissances en modélisation et implémentation de base de données relationnelles (Modules ICT 104, 105, 153).

1.4 Contenu livrable

2 Analyse / Conception

2.1 Méthodologie de travail

Afin de mener à bien ce projet, la méthodologie Waterfall, également connue sous le nom de méthode en cascade, a été choisie.

Cette approche de gestion de projet se caractérise par sa nature linéaire, fixe et structurée, suivant un processus séquentiel composé d'étapes clairement définies.

Cette méthodologie convient particulièrement pour ce projet dont le cahier des charges est fixe, car elle permet de planifier chaque phase en conséquence sans avoir besoin d'anticiper de changements.

Cette approche permet d'optimiser la conception et la réalisation du projet en s'assurant que chaque étape est bien définie et que les tâches sont effectuées dans l'ordre prévu.

De plus, les 6 étapes illustrées dans la figure ci-dessous offrent une catégorisation claire des différentes tâches à accomplir et permettent une planification judicieuse du projet.

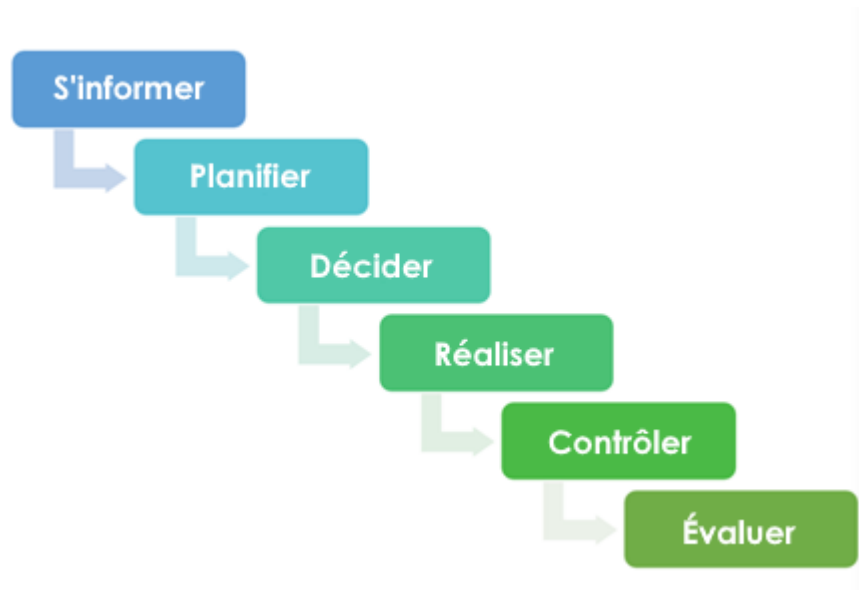


Figure 1 : Schéma de la méthode Waterfall des 6 pas

2.2 Environnement de travail

Afin de se rendre compte des difficultés et problématiques qui peuvent survenir lors de la réalisation de ce projet, il est important de redéfinir et de comprendre les fonctionnalités et les limites de chaque outils et concepts qui seront utilisés.

2.3 Maquettes

Différentes pages vont composer notre site. Nous verrons ici leur modèle de conception et leurs utilités pour notre projet.

Le header

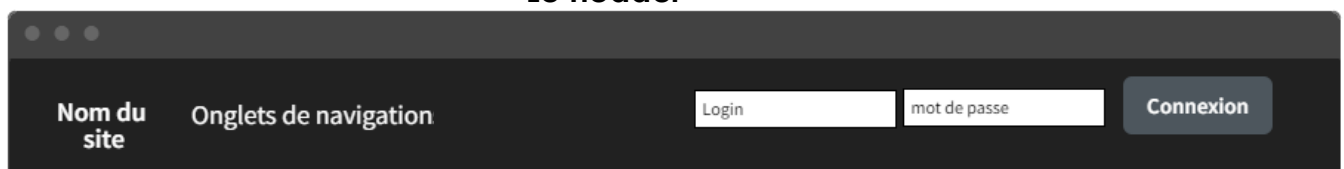


Figure 2 : modèle de conception du header du site

L'affichage du header est prévu pour être évolutif selon le profil de l'utilisateur connecté. Pour chaque cas, différentes informations seront affichées et accessibles en conséquence.

Cas no1 : L'utilisateur n'a pas de compte et ne peut pas se connecter

Dans ce cas de figure, l'utilisateur n'aura que la possibilité de créer un compte via les onglets de navigation ou de contacter un admin via les onglets de contact présents dans le footer. Tant qu'il n'aura pas créé de compte et ne sera pas connecté, il ne pourra effectuer aucune action en lien avec le contenu de l'application.

Cas no2 : L'utilisateur dispose d'un compte utilisateur et peut se connecter

Dans ce second cas de figure, dès l'instant où l'utilisateur se connecte à son compte en renseignant son login et son mot de passe, le formulaire de connexion disparaît pour laisser place à un message de bienvenue ainsi qu'un compteur de crédits pour rappeler à l'utilisateur l'état de son compte de crédits.

Les onglets de navigation lui permettent d'accéder à différentes pages :

- Mon profil
- Ajouter une carte
- Panier
- Accueil

Cas no3 : L'utilisateur dispose d'un compte admin

Dans le cas où l'utilisateur est un administrateur, il a les mêmes possibilités qu'un utilisateur à la différence qu'il dispose de tous les droits.

Le footer



Figure 3 : modèle de conception du footer du site

L'affichage du footer n'est pas évolutif. Il contient simplement différentes possibilités d'entrer en contact avec l'administrateur de l'application via mail ou les réseaux sociaux

La page d'accueil

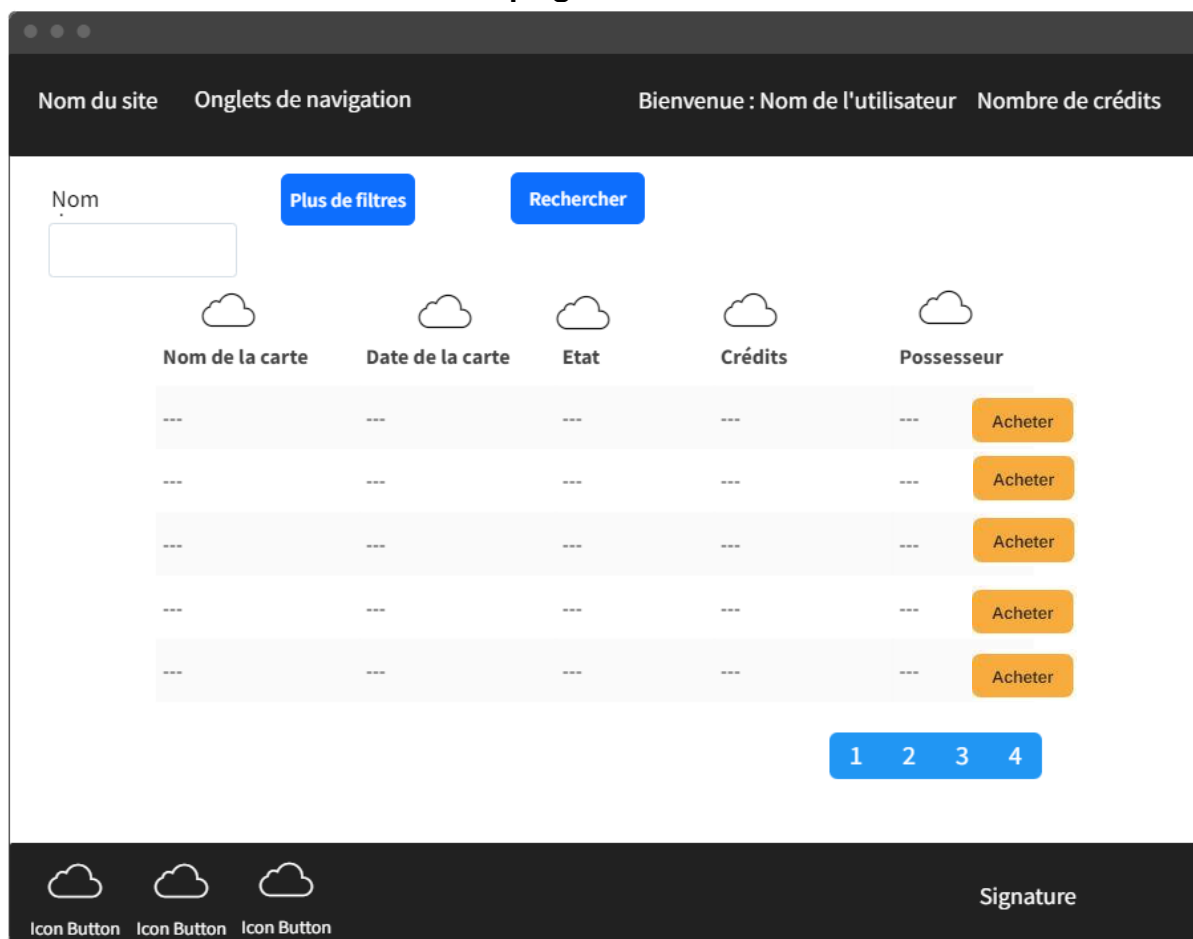


Figure 4: page d'accueil du site

La page d'accueil évolue en fonction du rôle de l'utilisateur connecté. Il est à tout moment possible de revenir à cette page en cliquant sur l'onglet accueil présent dans le header.

Cas no1 : L'utilisateur n'a pas de compte et ne peut pas se connecter

La page d'accueil affiche uniquement un message invitant l'utilisateur à créer un compte. Aucune autre interaction n'est possible et aucune information relative aux cartes de collection n'est accessible.

Cas no2 : L'utilisateur dispose d'un compte utilisateur ou administrateur et est connecté

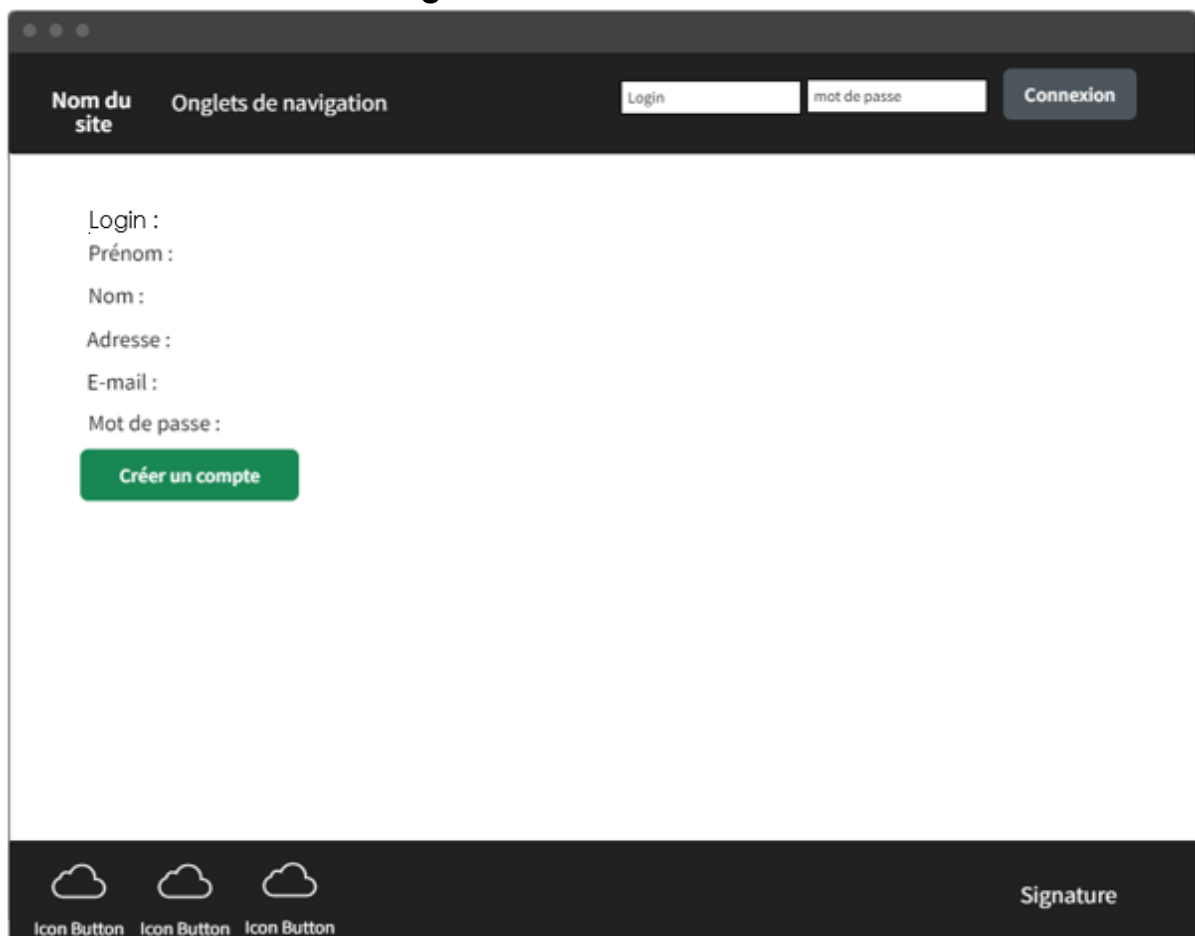
La page d'accueil affiche un tableau contenant toutes les cartes mises en vente sur le site ainsi que certaines de leurs informations. Il est possible de les trier grâce à un bouton au-dessus des colonnes ainsi que de les filtrer. Un filtre sur le nom est présent de base et il est possible de cliquer sur un bouton *plus de filtres* pour en afficher davantage et affiner la recherche. Ce bouton fait à

nouveau disparaître les filtres lorsqu'un nouveau clic se produit. La recherche par filtre s'effectue lorsque le bouton *rechercher* est cliqué.

Login :

L'utilisateur a également la possibilité d'afficher la page contenant toutes les informations d'une carte en cliquant sur le bouton *détails* dans les options du tableau. Il peut aussi ajouter une carte à son panier en cliquant sur le bouton *acheter* dans les options du tableau.

Page de création d'utilisateur



The screenshot shows a web application interface for user creation. At the top, there is a dark navigation bar with the text 'Nom du site' and 'Onglets de navigation'. To the right of this bar are two input fields labeled 'Login' and 'mot de passe', followed by a 'Connexion' button. Below the navigation bar, the main content area contains a form with the following labels: 'Login :', 'Prénom :', 'Nom :', 'Adresse :', 'E-mail :', and 'Mot de passe :'. A green button labeled 'Créer un compte' is positioned below the 'Mot de passe :' label. At the bottom of the page, there is a dark footer bar containing three cloud icons, each labeled 'Icon Button', and a 'Signature' label on the right.

Figure 5 : page de création d'utilisateur

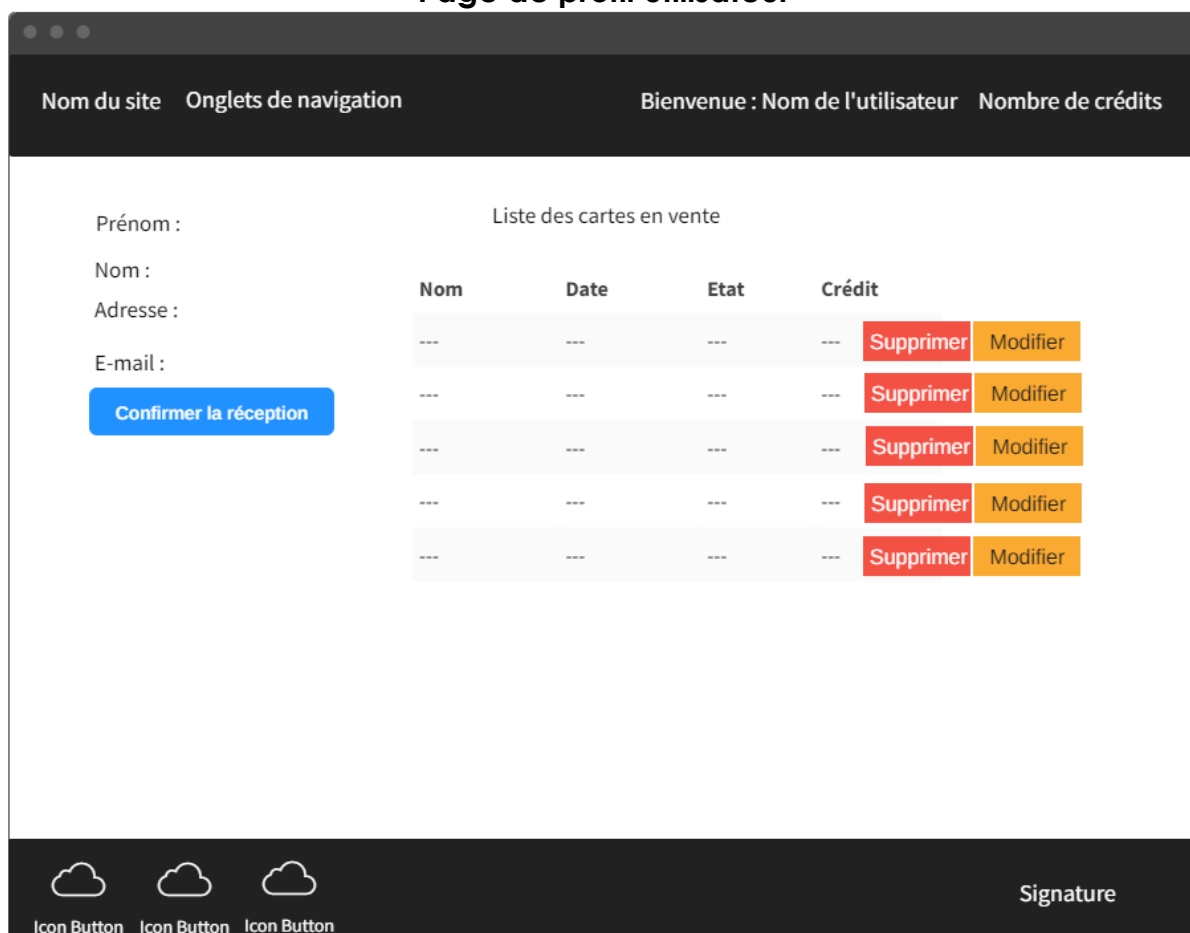
Lorsqu'un utilisateur n'a pas encore de compte, il a la possibilité d'en créer un en cliquant sur l'onglet *créer un compte*. Il doit ensuite renseigner les informations suivantes :

- Login
- Prénom
- Nom
- Adresse
- E-mail

- Mot de passe

Pour que l'inscription soit acceptée, une validation des champs est effectuée dès lors qu'il clique sur le bouton *Créer un compte*. S'il y a des erreurs, elles lui sont indiquées de façon contextuelle, sinon le compte est créé et ses informations sont enregistrées.

Page de profil utilisateur



		Nom	Date	Etat	Crédit		
Prénom :		---	---	---	---	Supprimer	Modifier
Nom :		---	---	---	---	Supprimer	Modifier
Adresse :		---	---	---	---	Supprimer	Modifier
E-mail :		---	---	---	---	Supprimer	Modifier
		---	---	---	---	Supprimer	Modifier

Figure 6 : page de profil de l'utilisateur

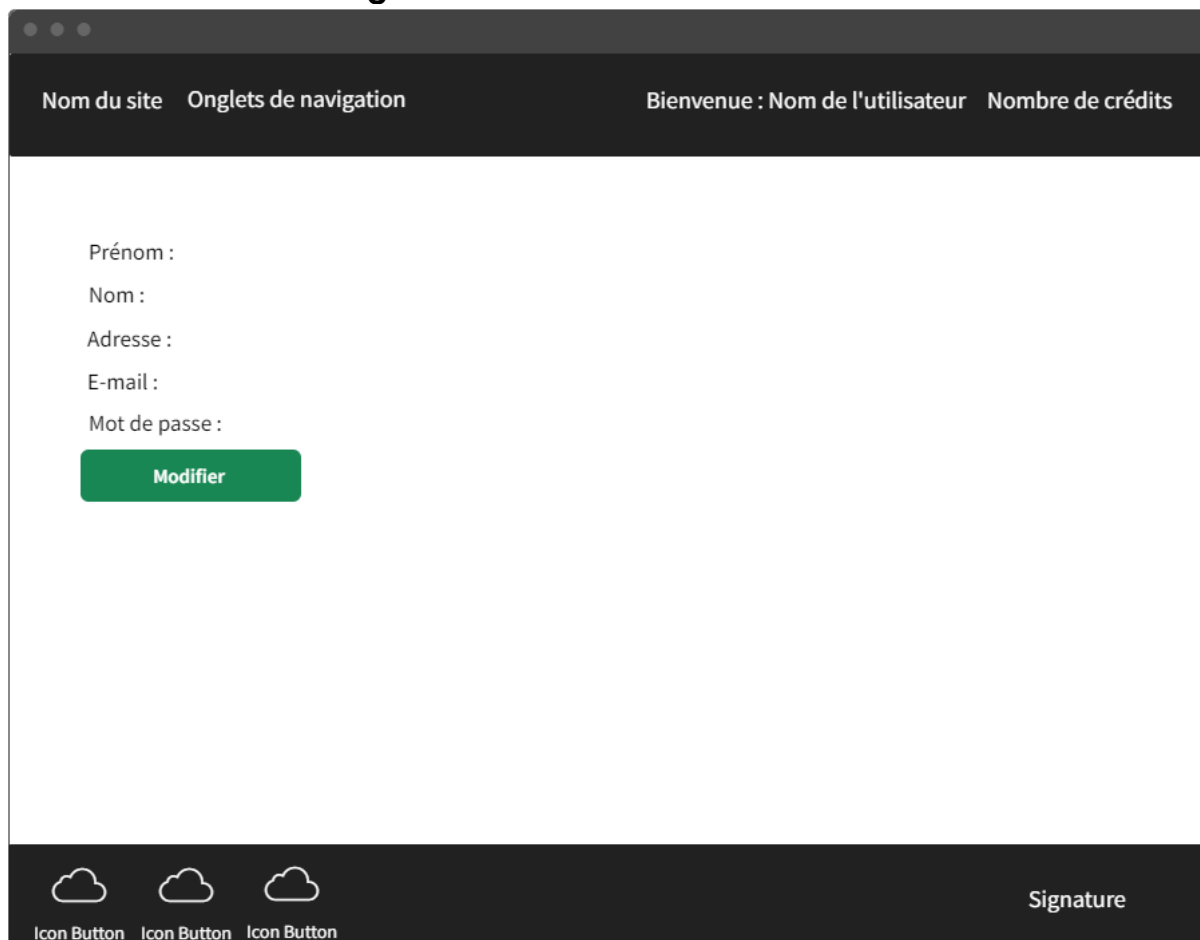
La page de profil d'un utilisateur contient toutes ses informations personnelles. Elles ne sont accessibles qu'à l'utilisateur en question en cliquant sur l'onglet *mon profil* à l'exception de l'administrateur s'il y a nécessité d'intervenir. Toutes les cartes que l'utilisateur propose à la vente sont également affichées sur un tableau et il a la possibilité de supprimer ou modifier une carte en particulier s'il clique sur le bouton correspondant. Il peut aussi consulter une carte en cliquant sur le bouton *détails* dans les options du tableau. S'il le souhaite, il peut modifier les informations de son profil depuis cette page en cliquant sur le bouton *modifier mon profil*.

Si l'utilisateur a passé une commande sur le site, un bouton *confirmer la réception* à propos de la commande en question apparaîtra sur son profil. Dès

lors qu'il clique sur ce bouton et confirme la réception, la transaction est considérée comme terminée.

Dans le cas où l'utilisateur n'a pas passé de commande, un simple message *aucune commande en attente* sera affiché.

Page de modification d'un utilisateur



Nom du site Onglets de navigation Bienvenue : Nom de l'utilisateur Nombre de crédits

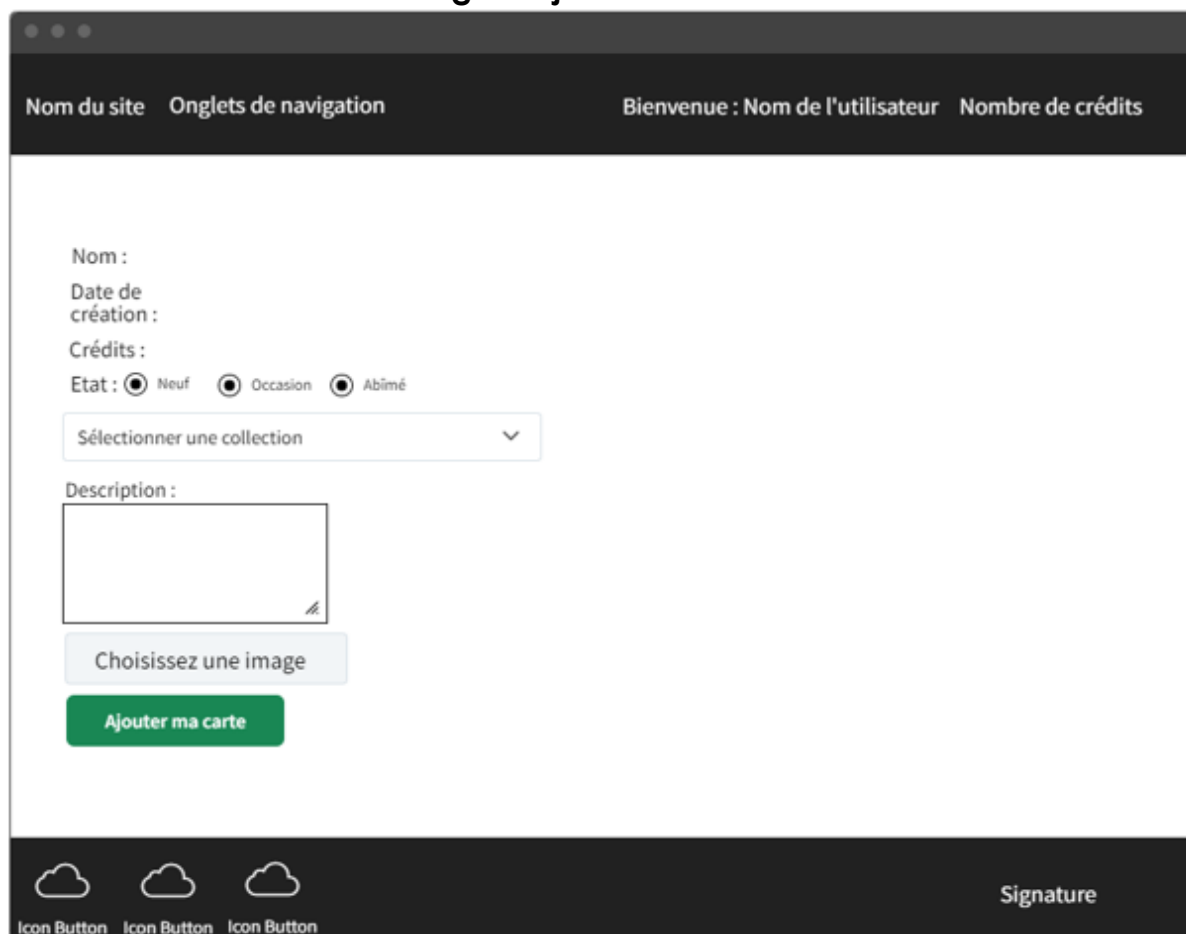
Prénom :
Nom :
Adresse :
E-mail :
Mot de passe :
Modifier

Icon Button Icon Button Icon Button Signature

Figure 7 : Page de modification d'un utilisateur

Lorsqu'un utilisateur souhaite modifier les informations de son profil, il doit à nouveau renseigner les informations qu'il souhaite modifier. Les valeurs de son profil précédemment enregistrées lui sont retournées de base dans les champs concernés. Dès lors qu'il clique sur le bouton *modifier* une validation des champs a lieu. S'il y a des erreurs, elles lui sont indiquées de façon contextuelle, sinon les modifications sont correctement enregistrées et son profil est mis à jour.

Page d'ajout d'une carte



Nom du site Onglets de navigation Bienvenue : Nom de l'utilisateur Nombre de crédits

Nom :
Date de création :
Crédits :
Etat : ☒ Neuf ☐ Occasion ☐ Abîmé
Sélectionner une collection
Description :
Choisissez une image
Ajouter ma carte

Icon Button Icon Button Icon Button Signature

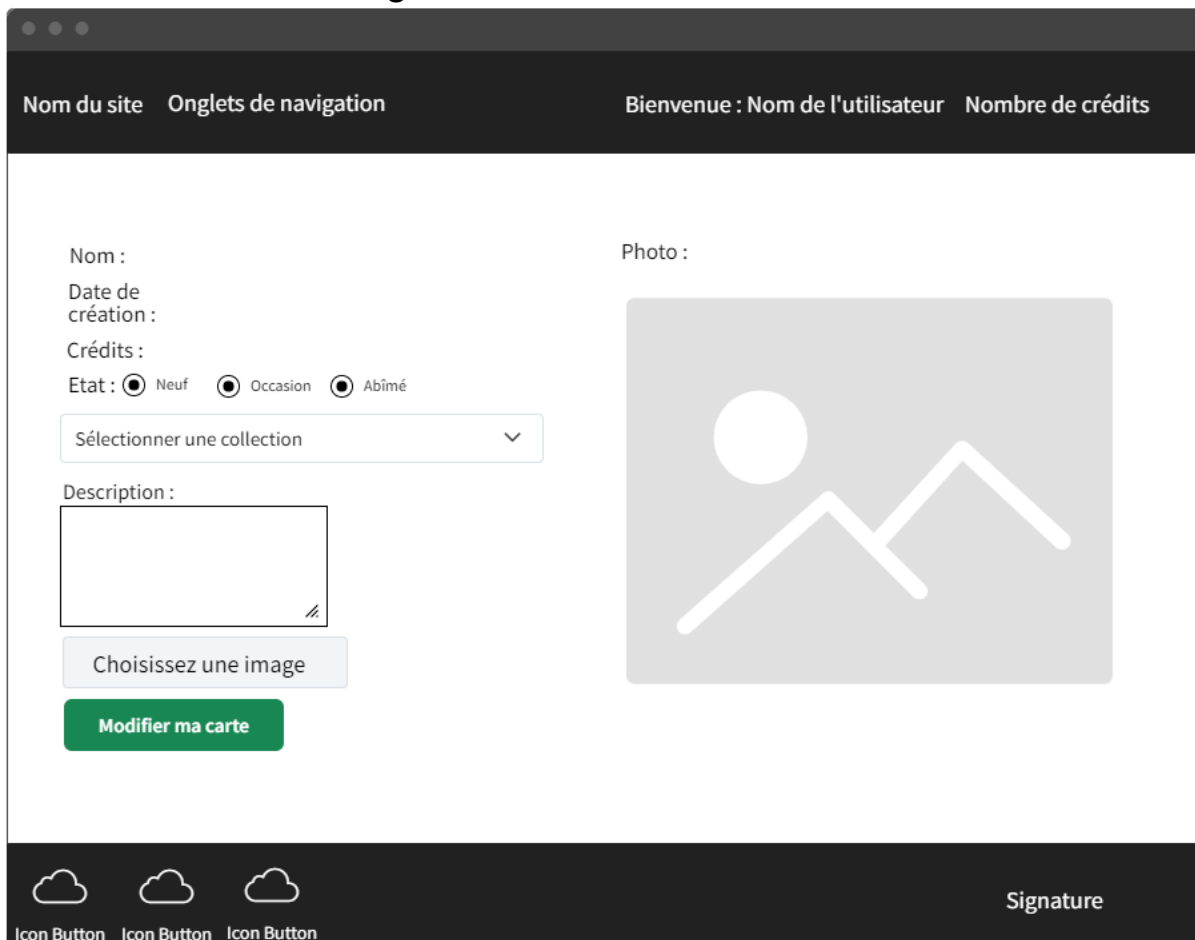
Figure 8 : Page d'ajout d'une carte

Un utilisateur connecté peut en tout temps ajouter une carte à vendre sur le site via l'onglet *Ajouter une carte*. Pour cela, il doit renseigner les informations suivantes :

- Nom
- Date de création
- Valeur en crédits
- Etat
- La collection
- Une description
- Une photo de la carte possédée

Dès lors qu'un clic est effectué sur le bouton *Ajouter une image* une validation des champs contrôle les informations renseignées. S'il y a des erreurs, elles sont affichées de façon contextuelle, sinon les informations sont bien enregistrées et la carte est mise en vente sur le site.

Page de modification d'une carte



Nom du site Onglets de navigation Bienvenue : Nom de l'utilisateur Nombre de crédits

Nom :
Date de création :
Crédits :
Etat : ☒ Neuf ☐ Occasion ☐ Abîmé
Sélectionner une collection
Description :
Choisissez une image
Modifier ma carte

Photo :

Icon Button Icon Button Icon Button Signature

Figure 9 : Page de modification d'une carte

Un utilisateur connecté peut en tout temps modifier une carte qu'il a déjà mise en vente depuis son profil en cliquant sur le bouton *modifier*.

Lorsqu'un utilisateur souhaite modifier les informations de l'une de ses cartes, il doit à nouveau renseigner les informations qu'il souhaite modifier. Les valeurs de précédemment enregistrées de sa carte lui sont retournées de base dans les champs concernés. La photo enregistrée précédemment s'affiche également sur cette page. Dès lors qu'il clique sur le bouton *modifier ma carte* une validation des champs a lieu. S'il y a des erreurs, elles lui sont indiquées de façon contextuelle, sinon les modifications sont correctement enregistrées et sa carte est mise à jour.

Page d'affichage d'une carte en particulier

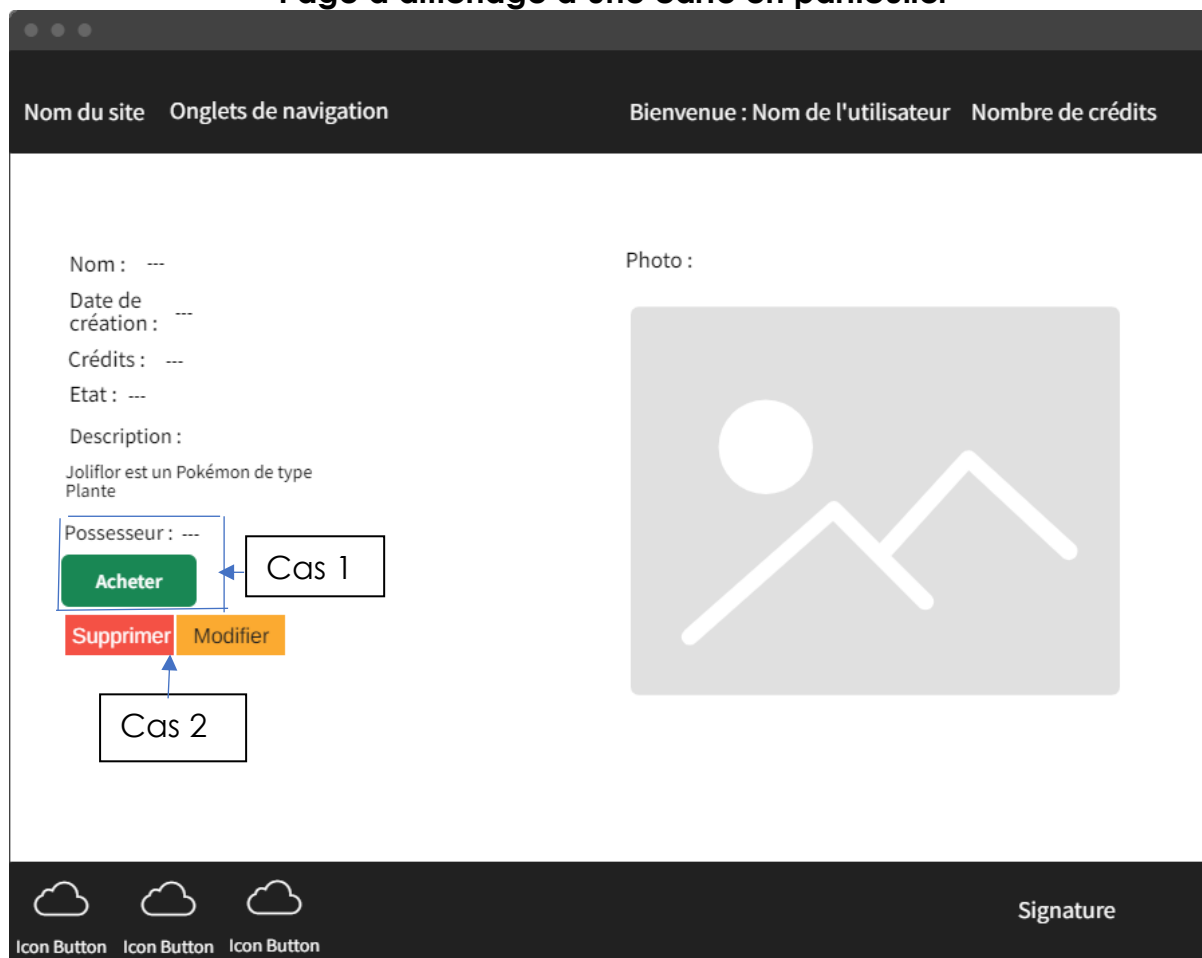


Figure 10 : Page d'affichage d'une carte en particulier

Un utilisateur connecté peut accéder en tout temps aux détails d'une carte en particulier en cliquant sur le nom de celle-ci. Il peut de cette façon consulter toutes les informations la concernant et y compris observer une photo de l'article.

Cas no1 : La carte n'appartient pas à l'utilisateur connecté

Dans le cas où l'utilisateur consulte les informations d'une carte qu'il n'a pas lui-même mise en vente, le nom du possesseur de la carte ainsi qu'un bouton *Acheter* sont visibles sur la page. En cas d'achat, la carte est ajoutée au panier de l'utilisateur.

Cas no2 : La carte appartient à l'utilisateur connecté

Dans le cas où l'utilisateur consulte les informations d'une carte qu'il a lui-même mise en vente, le champ *Possesseur* : ainsi que le bouton *Acheter* ne sont pas visibles sur la page. En lieu et place un bouton *Supprimer* permettant de supprimer la carte ainsi qu'un bouton *Modifier* redirigeant sur la page de modification d'une carte sont affichés sur la page.

Page de panier de l'utilisateur

Nom du site

Onglets de navigation

Bienvenue : Nom de l'utilisateur

Nombre de crédits




Panier

Cartes dans le panier

Nom	Possesseur	Etat	Crédit
---	---	---	---
---	---	---	---
---	---	---	---
---	---	---	---
---	---	---	---

Total : ---

Confirmer l'achat

Icon Button Icon Button Icon Button

Signature

Figure 11 : Page de panier de l'utilisateur

Un utilisateur connecté peut accéder en tout temps à son panier via l'onglet *Mon panier*. Sur cette page, il a la possibilité de consulter tous les articles qu'il a ajouté à son panier et de confirmer sa commande. Dès lors qu'il clique sur le bouton *Confirmer l'achat* la transaction débute.

Les crédits nécessaires à l'opération sont déduits du compte de l'acheteur et sont temporairement mis en attente. Une fois que le vendeur a envoyé les articles, l'acheteur confirme la bonne réception de ceux-ci depuis son profil. Les crédits mis en attente sont alors ajoutés au compte du vendeur.

2.4 Base de données

2.4.1 MCD

Pour fonctionner, notre application a besoin que la base de données puisse stocker différentes informations.

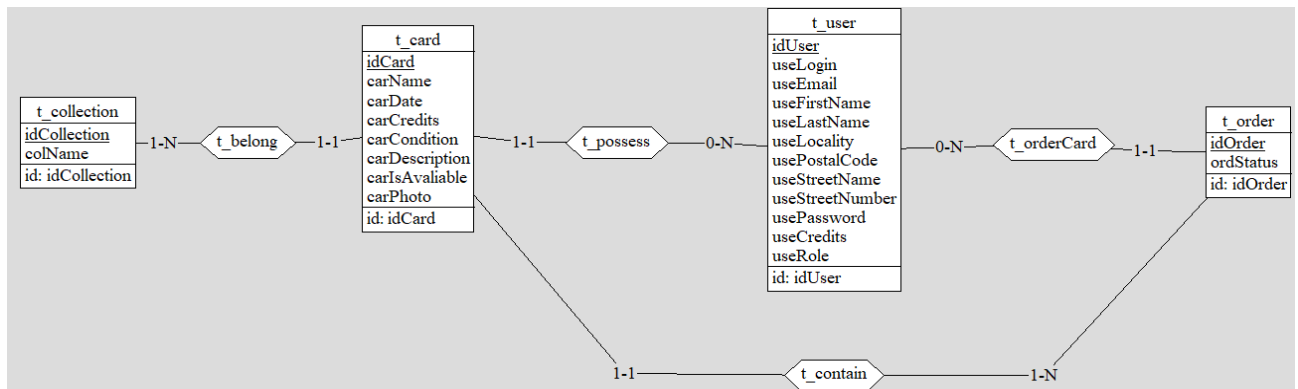


Figure 12 : MCD de la base de données

2.4.2 Entités

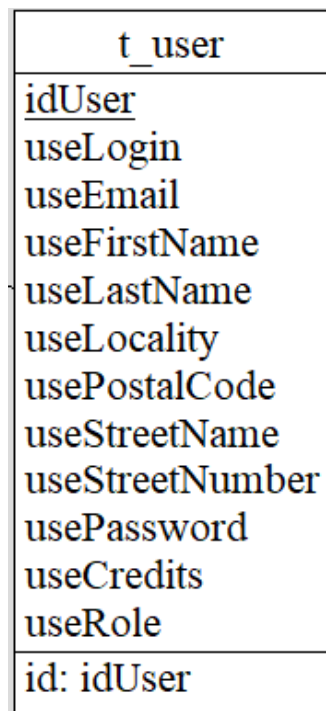


Figure 13 : entité t_user du MCD

L'entité **t_user** contient toutes les informations d'un utilisateur donné. Elle permet par exemple de définir si l'utilisateur est connecté ou non et si oui de quel type d'utilisateur il s'agit. Elle est composée des propriétés suivantes :

- *idUser* : Identifiant unique de l'utilisateur.
- *useLogin* : Login de l'utilisateur.
- *useEmail* : Email de l'utilisateur.
- *useFirstName* : Prénom de l'utilisateur.
- *useLastName* : Nom de famille de l'utilisateur.
- *useLocality* : Localité dans laquelle vit l'utilisateur.
- *usePostalCode* : Code postal de l'endroit où vit l'utilisateur.
- *useStreetName* : Nom de la rue où vit l'utilisateur.
- *UseStreetNumber* : Numéro de la rue où vit l'utilisateur.
- *usePassword* : Mot de passe de l'utilisateur pour se connecter sur le site.
- *useCredits* : Valeur en crédits sur le compte de l'utilisateur
- *useRole* : Rôle de l'utilisateur sur le site (profil utilisateur/administrateur).

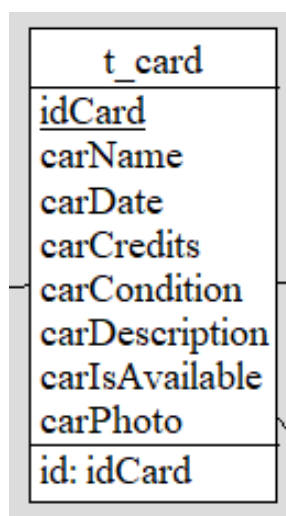


Figure 14 : entité *t_card* du MCD

L'entité *card* contient toutes les informations d'une carte donnée. Elle permet par exemple de définir la valeur en crédits, l'état ou le nom d'une carte. Elle est composée des propriétés suivantes :

- *idCard* : Identifiant unique de la carte.
- *carName* : Nom de la carte.
- *carDate* : Année de création de la carte.
- *carCredits* : Valeur en crédits de la carte.
- *carCondition* : Etat de la carte.
- *carDescription* : Description de la carte.
- *carIsAvailable* : Indicateur permettant de savoir si la carte est disponible à la vente ou non.
- *carPhoto* : Photo de la carte.
-

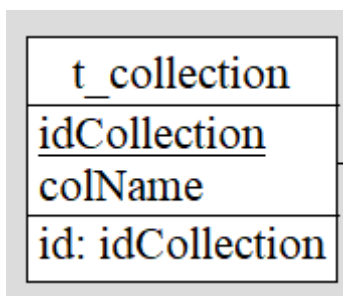


Figure 15 : entité *t_collection* du MCD

L'entité *t_collection* contient les informations relatives à la collection d'une carte. Elle est composée des propriétés suivantes :

- *idCollection* : Identifiant unique de la collection.
- *colName* : Nom de la collection.
-

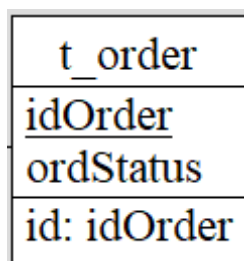


Figure 16 : entité *t_order*

L'entité *t_order* contient les informations relatives aux commandes réalisées par les utilisateurs et permet de gérer les crédits en attente lorsqu'une commande est passée. Elle est composée des propriétés suivantes :

- *idOrder* : Identifiant unique de la commande.
- *ordStatus* : Statut de la commande (en cours/terminée).

2.4.3 Cardinalités

Les cardinalités permettent d'établir le type de relation entre deux entités via une association.

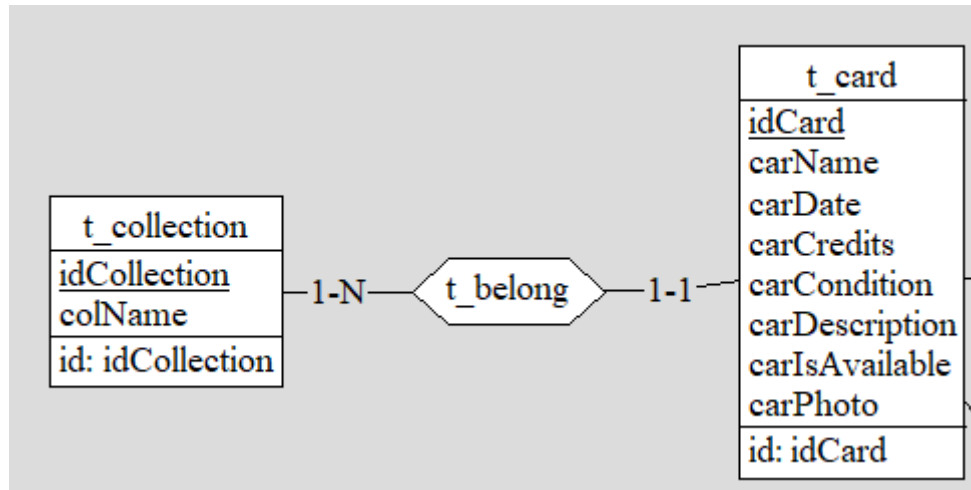


Figure 17 : Association *t_belong* entre les entités *t_collection* et *t_card*

L'association *t_belong* lie les entités *t_card* et *t_collection*. La cardinalité **1-1** indique qu'une carte ne peut appartenir qu'à une et une seule collection. En revanche, la cardinalité **1-N** indique qu'une ou plusieurs cartes peuvent appartenir à la même collection.

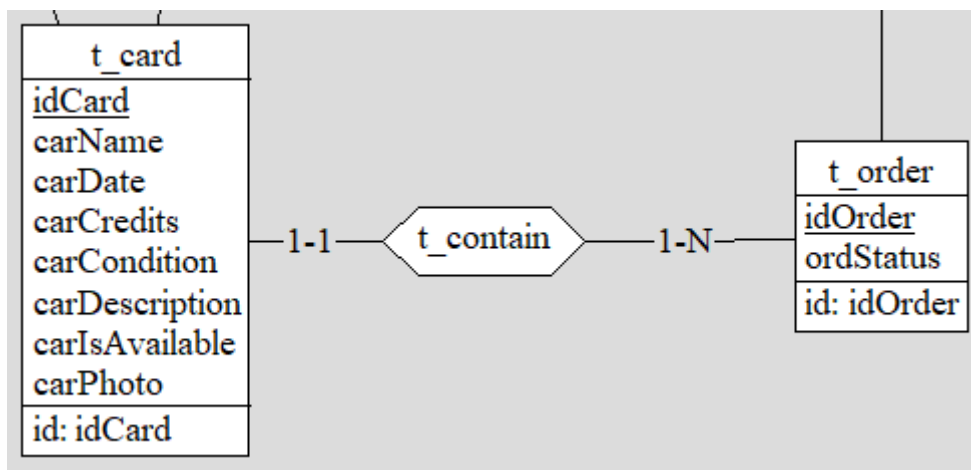


Figure 18 : Association *t_contain* entre les entités *t_card* et *t_order*

L'association *t_contain* lie les entités *t_card* et *t_order*. La cardinalité **1-1** indique que chaque carte, étant unique, ne peut être contenue que dans une et une seule commande à la fois. La cardinalité **1-N** indique quant à elle qu'une commande doit contenir au moins une carte mais peut également contenir plusieurs cartes différentes.

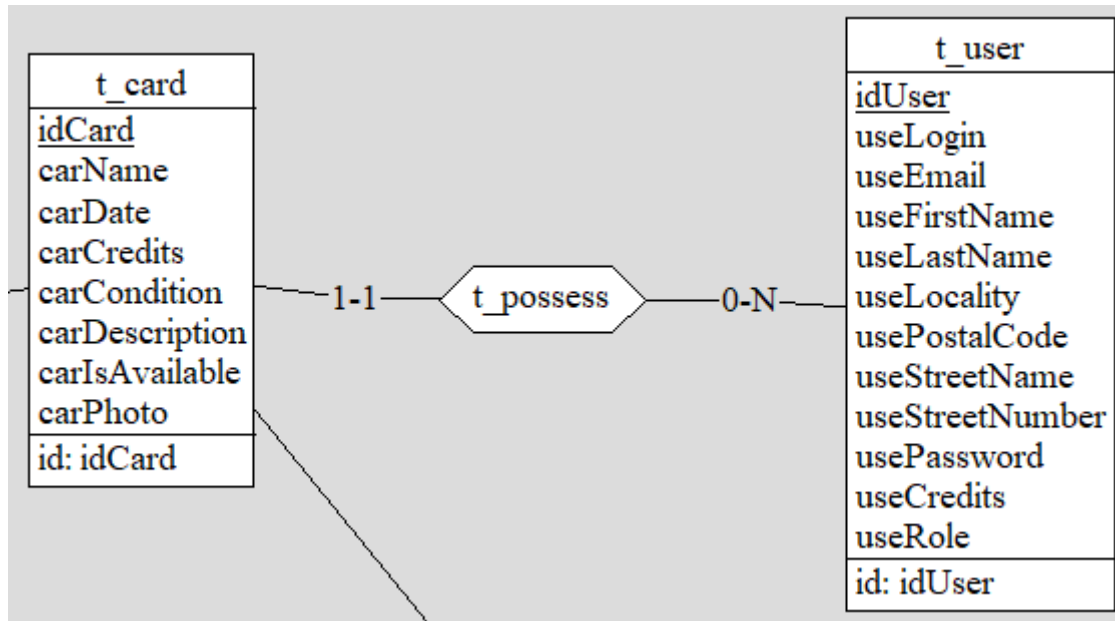


Figure 19 : association *t_possess* entre les entités *t_card* et *t_user*

L'association *t_possess* lie les entités *t_card* et *t_user*. La cardinalité **1-1** indique qu'une carte, étant un objet unique, ne peut être possédée qu'une et une seule fois par un utilisateur. La cardinalité **0-N** quant à elle indique qu'un utilisateur n'a pas l'obligation d'échanger une carte mais peut en échanger autant qu'il le souhaite.

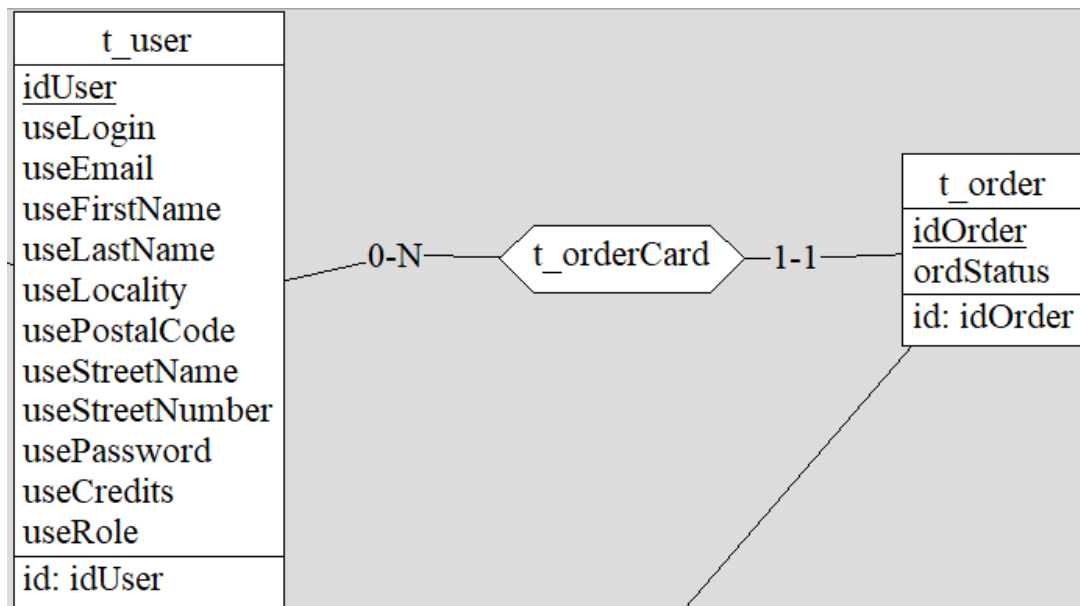


Figure 20 : association *t_orderCard* entre les entités *t_user* et *t_order*

L'association *t_orderCard* lie les entités *t_user* et *t_order*. La cardinalité **0-N** indique qu'un utilisateur n'a pas l'obligation de passer une commande mais qu'il a la possibilité d'en passer autant qu'il le souhaite. La cardinalité **1-1** indique quant à elle qu'une commande, étant unique, ne peut être passée que par un et un seul utilisateur.

2.4.4 MLD

Selon notre MCD, voici le résultat du MLD. Des contraintes référentielles ont été créées et l'intégrité de nos données entre les différentes tables est maintenant assurée.

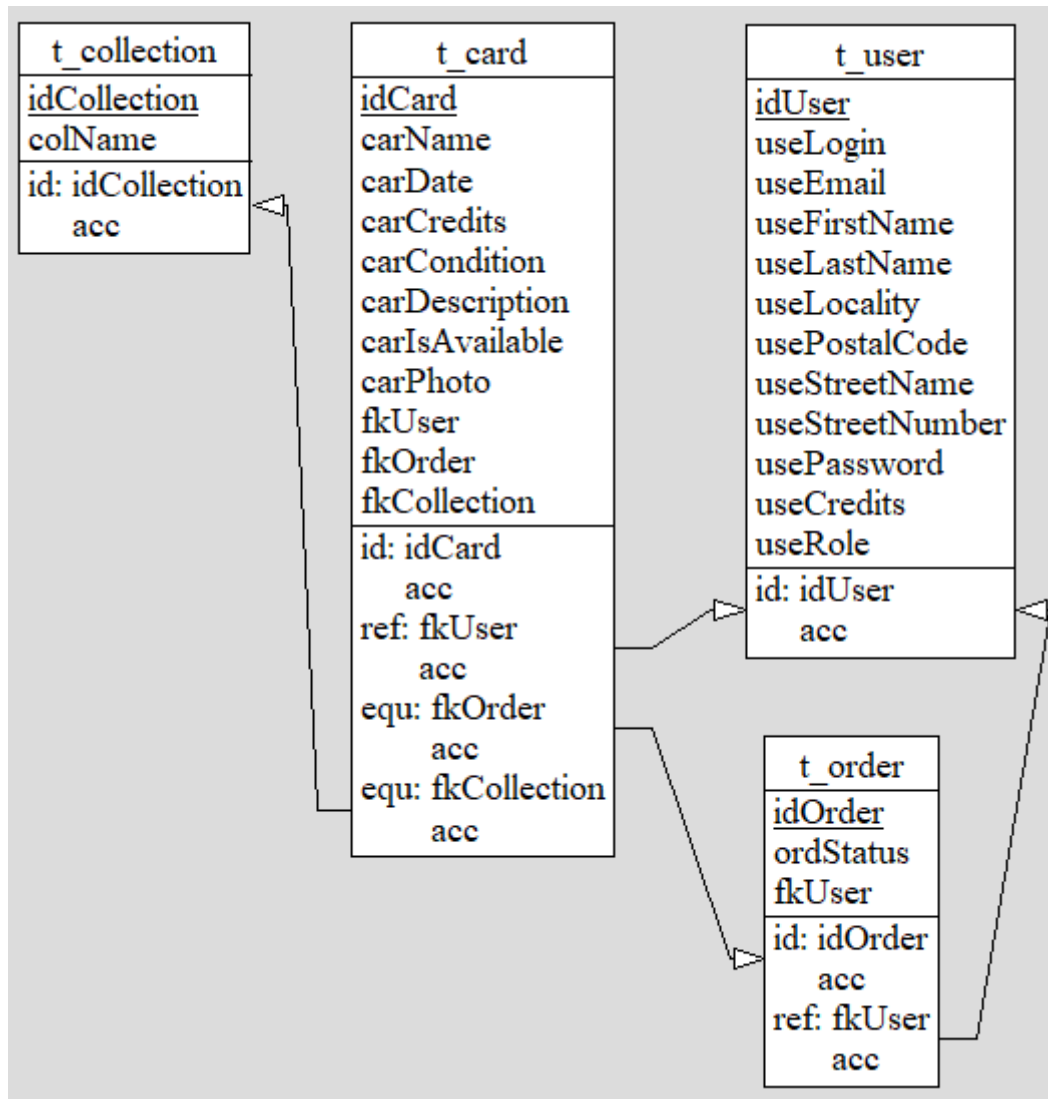


Figure 21 : MLD de la base de données

Nous pouvons constater que la table *t_card* récupère quant à elle les clés étrangères des tables *t_order*, *t_user* et *t_collection*. Cela nous permet de lier une carte à une commande lorsqu'elle est passée, à un utilisateur ainsi qu'à une collection.

Enfin, la table *t_order* récupère la clé étrangère de la table *t_user*, ce qui nous permet de lier une commande à un utilisateur. Cela nous sera utile lors d'une réalisation de commande puisque même si elle contient plusieurs cartes appartenant au même utilisateur, il ne s'agira que d'une seule et même commande.

2.5 Stratégie de test

Afin d'effectuer des vérifications testant le fonctionnement de nos différentes fonctionnalités, il est nécessaire d'établir une stratégie qui permettra de réaliser ces tests selon différents scénarios en anticipant le résultat escompté.

Nom du test	Scénario	Résultat attendu
Ajout d'une carte avec succès.	<p>Un utilisateur connecté clique sur l'onglet <i>Ajouter une carte</i>. Il renseigne les informations demandées et télécharge une image au format .JPG ne dépassant pas 2mo.</p> <p>Il clique ensuite sur le bouton <i>Ajouter une carte</i> lorsqu'il est satisfait des informations renseignées.</p>	<p>La validation des champs s'effectue sans rencontrer d'erreurs et un message affiche : <i>Votre carte a bien été enregistrée.</i></p> <p>L'utilisateur peut consulter sa carte en cliquant sur l'onglet <i>Mon profil</i> puis en parcourant le tableau des cartes qu'il a mis en vente.</p>
Ajout d'une carte sans respecter le format de l'image.	<p>Un utilisateur connecté clique sur l'onglet <i>Ajouter une carte</i>. Il renseigne les informations demandées, respecte la taille maximum de 2mo mais envoie un fichier au format .PNG.</p> <p>Il clique ensuite sur le bouton <i>Ajouter une carte</i> lorsqu'il est satisfait des informations renseignées.</p>	<p>La validation des champs s'effectue et repère que le fichier envoyé n'est pas au format attendu.</p> <p>Une erreur contextuelle affiche : <i>Merci de n'envoyer que des photos au format .JPG.</i></p> <p>Les informations renseignées dans les champs qui n'ont pas déclenché d'erreur sont retournés à l'utilisateur.</p>
Accès à une page non autorisée.	Un utilisateur non connecté essaye d'accéder à la page <i>Ajouter une carte</i> via l'URL sans qu'il ne dispose des droits pour y accéder.	Une page d'erreur 403 s'affiche et empêche l'utilisateur d'accéder à la page web. Celle-ci l'invite à rejoindre l'accueil ou à se connecter.
Téléchargement du dossier de l'application depuis Github sans avoir	Un utilisateur télécharge le dossier de l'application depuis le répertoire Github dans le but de	Le fichier secrets.json contenant les informations sensibles de l'application est absent

accès au fichier secrets.json.	l'utiliser dans son environnement.	des fichiers du dossier téléchargé.
Tri des cartes affichées sur la page d'accueil.	Un utilisateur connecté souhaite trier les cartes par ordre alphabétique à partir du nom. Dans ce but, il clique sur l'icone au-dessus de la colonne permettant de réaliser le tri.	Les cartes s'affichent correctement par ordre alphabétique croissant ou décroissant après le clique sur l'icône.
Filtrage des cartes affichées sur la page d'accueil selon leur état.	Un utilisateur souhaite filtrer les cartes affichées sur la page d'accueil selon leur état. Il clique sur le bouton <i>plus de filtres</i> et indique un état <i>Neuf</i> sur le filtre concerné qui est correctement apparu. Il clique ensuite sur le bouton <i>Rechercher</i> .	Seules les cartes renseignées comme étant dans un état <i>Neuf</i> s'affichent dans le tableau des cartes en vente de la page d'accueil.
Crédits mis en attente lors d'une transaction	Un utilisateur connecté clique sur le bouton <i>Acheter</i> de la carte qui l'intéresse depuis le tableau de la page d'accueil. Après avoir cliqué sur l'onglet <i>Mon panier</i> il constate que la carte a correctement été ajoutée à son panier. Il confirme l'achat en cliquant sur le bouton <i>Confirmer l'achat</i> .	Les crédits sont bien débités du compte de l'acheteur mais ne sont pas crédités sur le compte du vendeur avant que l'acheteur ne confirme la réception sur son profil.
Réussite de l'utilisation du manuel d'installation	Un collègue ou un enseignant disponible télécharge l'application depuis le dépôt git et consulte le manuel d'installation présent dans le dossier sous forme de README.md. Il essaie ensuite de le reproduire méticuleusement en l'adaptant à son environnement si nécessaire.	La personne qui a suivi les instructions du manuel d'installation a réussi à lancer l'application dans son propre environnement.

2.5.1 Logiciels et outils supplémentaires

Dans cette section se trouvent les différents logiciels et outils qui n'ont pas encore été mentionnés et qui seront utilisés afin de mener à bien la réalisation de ce projet, ainsi qu'une justification de leur utilisation.

- **Docker**, plateforme permettant de lancer des applications dans des conteneurs. C'est grâce à cet outil que seront stockés la base de données, le site web ainsi que PHPMyAdmin.
- **PHPMyAdmin**, application Web de gestion pour les systèmes de base de données MySQL. Grâce à cet outil, il sera facile de visualiser la base de données et d'y ajouter des données manuellement si besoin.
- **ChatGPT**, modèle de langage.
- **DBMain**, logiciel pour la conception du MCD et du MLD de la base de données.

2.6 Risques techniques

2.6.1 Mise en place de l'environnement Docker

- *risques techniques (complexité, manque de compétences, ...).*

Décrire aussi quelles solutions ont été appliquées pour réduire les risques (priorités, formation, actions, ...).

Fournir tous les document de conception:

- *le choix du matériel HW*
- *le choix des systèmes d'exploitation pour la réalisation et l'utilisation*
- *le choix des outils logiciels pour la réalisation et l'utilisation*
- *site web: réaliser les maquettes avec un logiciel, décrire toutes les animations sur papier, définir les mots-clés, choisir une formule d'hébergement, définir la méthode de mise à jour, ...*
- *bases de données: décrire le modèle relationnel, le contenu détaillé des tables (caractéristiques de chaque champs) et les requêtes.*
- *programmation et scripts: organigramme, architecture du programme, découpage modulaire, entrées-sorties des modules, pseudo-code / structogramme...*

Le dossier de conception devrait permettre de sous-traiter la réalisation du projet !

3 Réalisation

INTRO + VERSIONS OUTILS

3.1

3.2 Base de données

3.2.1 Importation du fichier .SQL

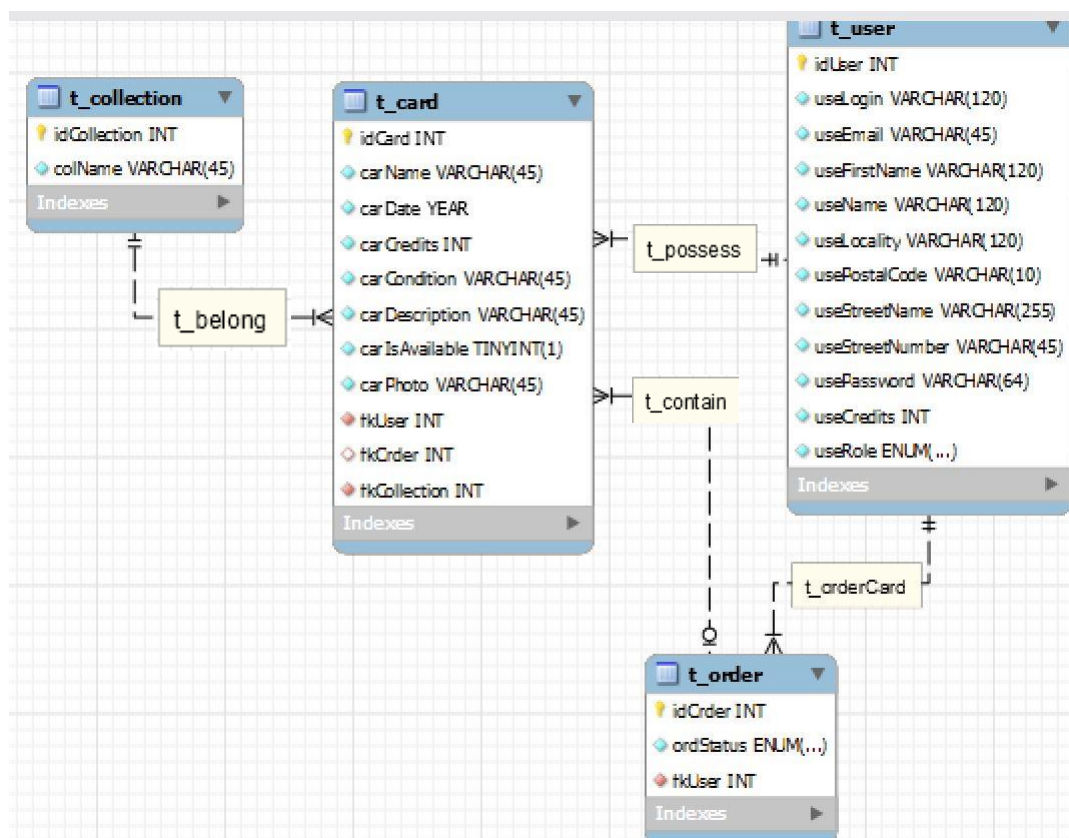


Figure 22 : MLD sur MySQL Workbench

Pour la génération et l'importation du fichier .SQL, le programme MySQL Workbench a été préféré à DB-MAIN car il propose une définition beaucoup plus fine des types pour les attributs ce qui permet de générer un fichier .SQL déjà optimisé pour notre application.

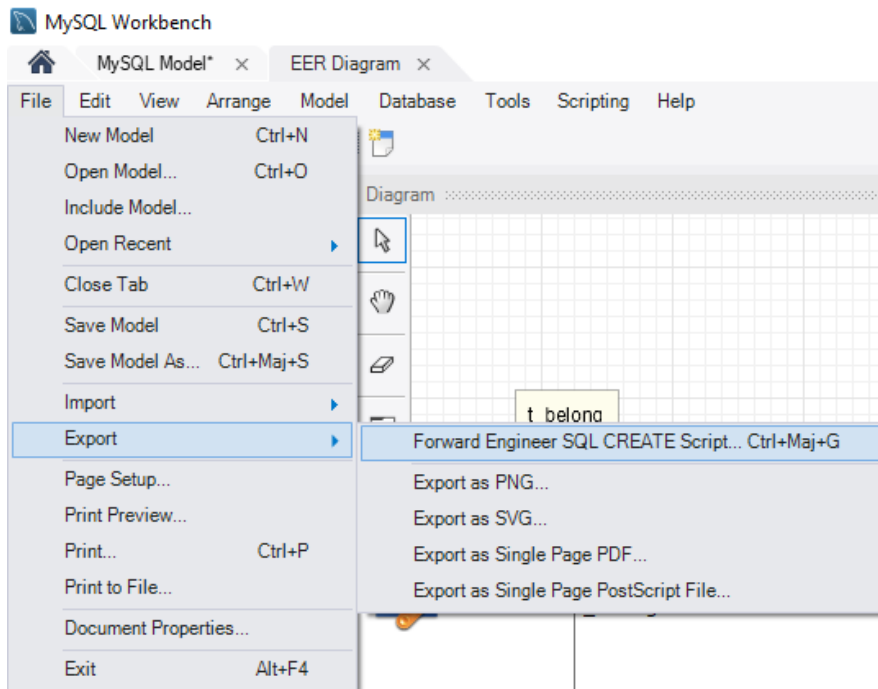


Figure 23 : Export du code .SQL depuis le MLD

Nous exportons le code correspondant à notre MLD dans un fichier .SQL avant de l'importer dans PHPMyAdmin.



Figure 24 : Importation du fichier .SQL dans phpMyAdmin

Une fois notre fichier .SQL généré et adapté à nos besoins nous l'importons sur PHPMyAdmin.

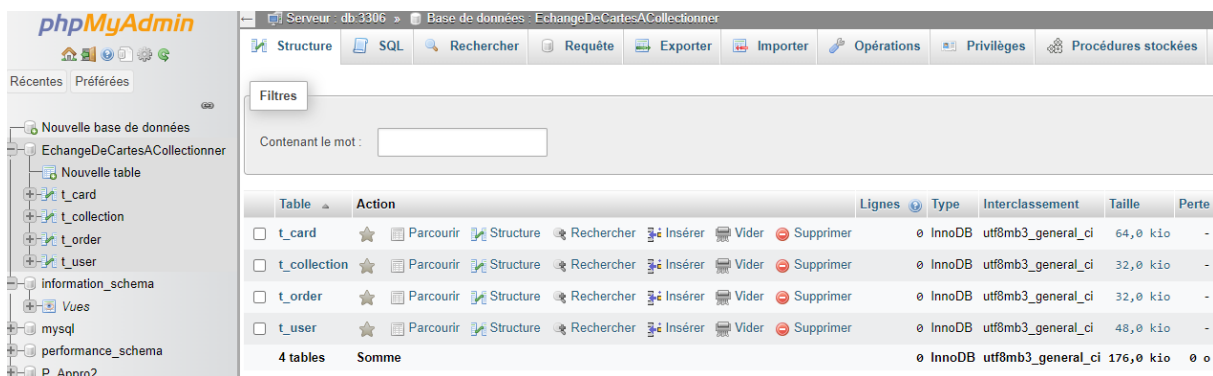


Figure 25 : Interface de la base de données sur phpMyAdmin

Nous pouvons constater que la base de données a correctement été importée sur phpMyAdmin.

3.2.2 MPD

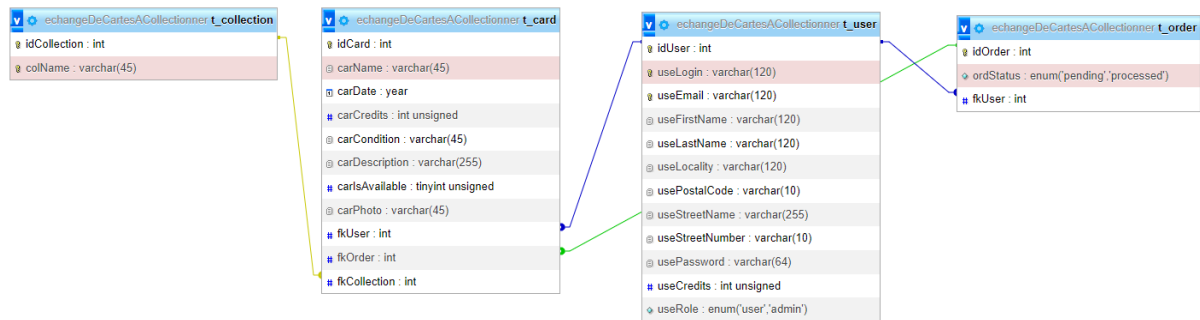


Figure 26 : Vue du MPD de l'application sur phpMyAdmin

A présent que notre base de données a été importée sur phpMyAdmin, nous pouvons avoir une vue d'ensemble sur la logique de notre base de données via notre MPD.

3.2.3 Tables et type des attributs

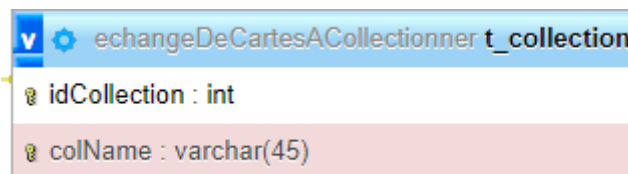
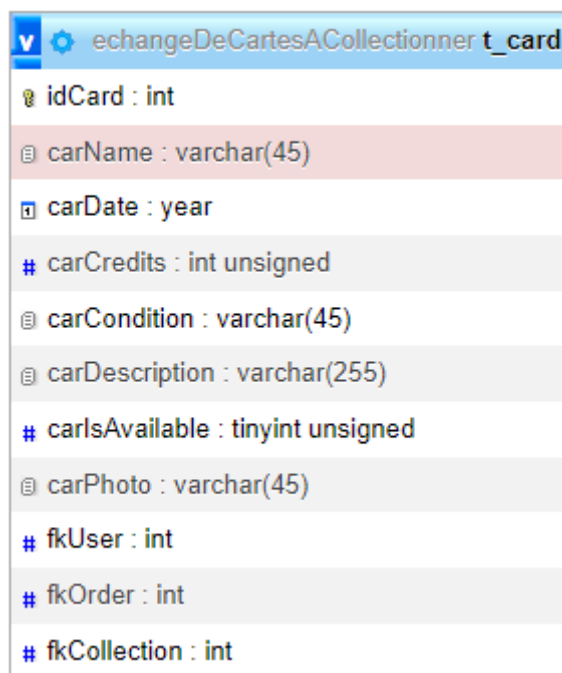


Figure 27 : table t_collection du MPD

Nom	Type	Attributs	Null	Unique	Valeur Par défaut	Extra
idCollection	int	-	Non	Oui	Aucun(e)	Auto-Increment
colName	varchar(45)	-	Non	Oui	Aucun(e)	-

Dans la table *t_collection* :

- *idCollection*, en tant que clé primaire, est en *auto-increment*.
- *ColName* est unique car il n'est pas possible d'enregistrer plusieurs fois la même collection.



échangeDeCartesACollectionner t_card	
idCard	int
carName	varchar(45)
carDate	year
carCredits	int unsigned
carCondition	varchar(45)
carDescription	varchar(255)
carlsAvailable	tinyint unsigned
carPhoto	varchar(45)
fkUser	int
fkOrder	int
fkCollection	int

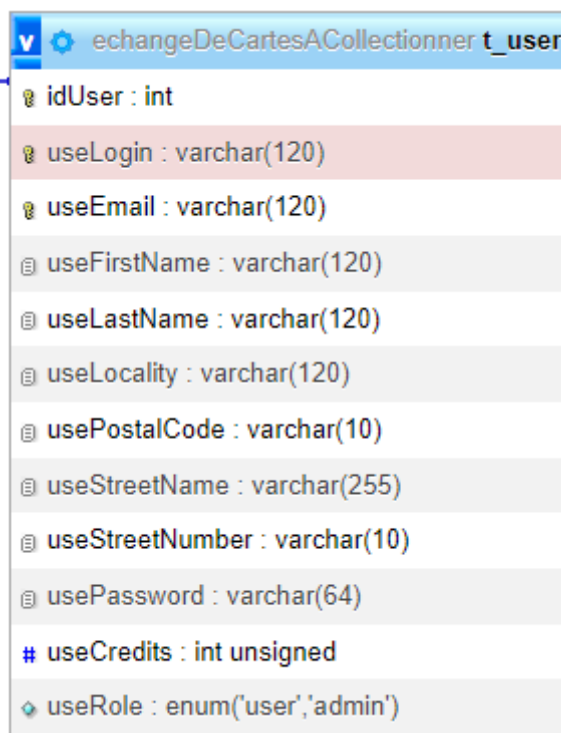
Figure 28 : table t_card du MPD

Nom	Type	Attributs	Null	Unique	Valeur Par défaut	Extra
idCard	int	-	Non	Oui	Aucun(e)	Auto-Increment
carName	varchar(45)	-	Non	Non	Aucun(e)	-
carDate	year	-	Non	Non	Aucun(e)	-
carCredits	int	unsigned	Non	Non	Aucun(e)	-
carCondition	varchar(45)	-	Non	Non	Aucun(e)	-
carDescription	varchar(255)	-	Non	Non	Aucun(e)	-
carlsAvailable	tinyint	unsigned	Non	Non	1	-
carPhoto	Varchar(45)	-	Non	Non	Aucun(e)	-
fkUser	int	-	Non	Non	Aucun(e)	-
fkOrder	int	-	Oui	Non	NULL	-
fkCollection	int	-	Non	Non	Aucun(e)	-

Dans la table *t_card* :

- *idCard*, en tant que clé primaire, est en *auto-increment*.
- *CarDate* est de type *year*, ce qui nous permet d'extraire l'année de création de la carte.
- Les attributs de *carCredits* sont *unsigned* car le montant de crédits ne peut pas être négatif.
- *carlsAvailable* est de type *tinyint* car il nous faut uniquement définir un état disponible ou indisponible, la valeur peut passer de 0 à 1 mais dès sa création, la carte a une valeur par défaut à 1 car elle est disponible sur le marché tant que personne ne l'a achetée.

- *fkOrder* peut être NULL car tant qu'une carte n'a pas été achetée, elle n'appartient à aucune commande.



échangeDeCartesACollectionner t_user	
idUser	int
useLogin	varchar(120)
useEmail	varchar(120)
useFirstName	varchar(120)
useLastName	varchar(120)
useLocality	varchar(120)
usePostalCode	varchar(10)
useStreetName	varchar(255)
useStreetNumber	varchar(10)
usePassword	varchar(64)
useCredits	int unsigned
useRole	enum('user','admin')

Figure 29 : table t_user du MPD

Nom	Type	Attributs	Null	Unique	Valeur Par défaut	Extra
idUser	int	-	Non	Oui	Aucun(e)	Auto-Increment
useLogin	varchar(120)	-	Non	Oui	Aucun(e)	-
useEmail	varchar(120)	-	Non	Oui	Aucun(e)	-
useFirstName	varchar(120)	-	Non	Non	Aucun(e)	-
useLastName	varchar(120)	-	Non	Non	Aucun(e)	-
useLocality	varchar(120)	-	Non	Non	Aucun(e)	-
usePostalCode	varchar(10)	-	Non	Non	Aucun(e)	-
useStreetName	varchar(255)	-	Non	Non	Aucun(e)	-
useStreetNumber	varchar(10)	-	Non	Non	Aucun(e)	-
usePassword	varchar(64)	-	Non	Non	Aucun(e)	-
useCredits	int	unsigned	Non	Non	Aucun(e)	-
useRole	enum('user', 'admin')	-	Non	Non	user	-

Dans la table *t_user* :

- *idUser* , en tant que clé primaire, est en *auto-increment*.
- *useLogin* est unique car un utilisateur peut utiliser la valeur de cet attribut pour se connecter à son compte via le login.
- *useEmail* est unique car un utilisateur peut utiliser la valeur de cet attribut pour se connecter à son compte via le login.
- *usePassword* est de type *varchar(64)* car le mot de passe sera hashé.
- Les attributs de *useCredits* sont *unsigned* car la valeur des crédits ne peut pas être négatifs.
- *useRole* est de type *enum('user', 'admin')*, ce qui nous permet de définir le rôle de l'utilisateur lors de la création du compte. Le rôle admin n'était définissable que par l'administrateur lui-même via l'interface de phpMyAdmin, la valeur par défaut de cet attribut est *user*.

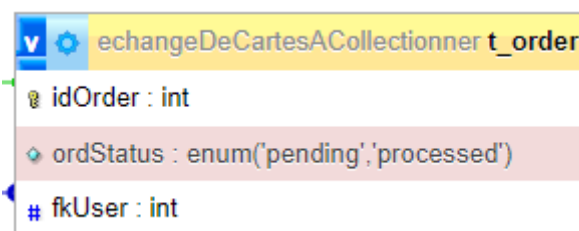


Figure 30 : table *t_order* du MPD

Nom	Type	Attributs	Null	Unique	Valeur Par défaut	Extra
idOrder	int	-	Non	Oui	Aucun(e)	Auto-Increment
ordStatus	enum('pending', 'processed')	-	Non	Non	Aucun(e)	-
fkUser	int	-	Non	Non	Aucun(e)	-

Dans la table *t_order* :

- *IdOrder*, en tant que clé primaire, est en *auto-increment*.
- *ordStatus* est de type *enum('pending', 'processed')* ce qui nous permet de gérer l'état de la commande. Tant qu'un utilisateur n'achète pas de cartes, la commande n'existe pas. Dès qu'il achète une carte, le statut passe à *pending*. Et lorsque l'acheteur confirme la réception, le statut passe à *processed*.

3.3

Décrire la réalisation "physique" de votre projet

- les répertoires où le logiciel est installé
- la liste de tous les fichiers et une rapide description de leur contenu (des noms qui parlent !)
- les versions des systèmes d'exploitation et des outils logiciels
- la description exacte du matériel
- le numéro de version de votre produit !
- programmation et scripts: librairies externes, dictionnaire des données, reconstruction du logiciel - cible à partir des sources.

NOTE : Evitez d'inclure les listings des sources, à moins que vous ne désiriez en expliquer une partie vous paraissant importante. Dans ce cas n'incluez que cette partie...

4 Tests

4.1.1

LISTE DES FONCTIONNALITÉS QUI DOIVENT FONCTIONNER EN TABLEAU AVEC NOM DU TEST, SCENARIO, RESULTAT ATTENDU ET OBTENU

4.1.2

LISTE DES FONCTIONNALITÉS QUI DOIVENT FONCTIONNER

4.1.3

4.2 Erreurs restantes

S'il reste encore des erreurs:

- *Description détaillée*
- *Conséquences sur l'utilisation du produit*
- *Actions envisagées ou possibles*

4.3 Liste des documents fournis

Lister les documents fournis au client avec votre produit, en indiquant les numéros de versions

- *le rapport de projet*
- *le manuel d'Installation (en annexe)*
- *le manuel d'Utilisation avec des exemples graphiques (en annexe)*
- *autres...*

5 Conclusions

5.1 Bilan des fonctionnalités

5.2 Comparaison de la planification

5.3 Critiques / Finalité du projet

5.4 Difficultés particulières

5.5 Conclusion personnelle

Développez en tous cas les points suivants:

- Objectifs atteints / non-atteints
- Points positifs / négatifs
- Difficultés particulières
- Suites possibles pour le projet (évolutions & améliorations)

6 Lexique

7 Table d'illustrations

FIGURE 1 : SCHEMA DE LA METHODE WATERFALL.....	6
--	---

8 Annexes

8.1 Résumé du rapport du TPI / version succincte de la documentation

8.1.1 Situation de départ

8.1.2 Mise en œuvre

8.1.3 Résultats

8.2 Sources – Bibliographie

9 Bibliographie

9.1 Manuel d'Utilisation

9.2 Archives du projet

Media, ... dans une fourre en plastique