



Rapport : Projet de session

Travail présenté à Prof. Richard Khoury dans le cadre du cours *GLO-2005 : Modèles et langages des bases de données pour ingénieurs*

Travail réalisé par Claire Bouttes et Vincent Girard

Hiver 2021

Table des matières

1	Énonciation du problème et de ses exigences	1
2	Modèle entité-relation du système	1
3	Modèle relationnel	3
4	Création des relations	3
5	Requêtes et routines	4
5.1	Routines	4
6	Indexation et optimisation	4
6.1	Relations Client	5
6.2	Relation Artiste	5
6.3	Relation Oeuvre	5
6.4	Relation Commande	6
6.5	Relation Commentaire	6
6.6	Relation Facture	6
7	Normalisation des relations	6
7.1	Relation Client	6
7.2	Relation Artiste	6
7.3	Relation Oeuvre	6
7.4	Relation Commentaire	7
7.5	Relation Commande	7
7.6	Relation Facture	7
8	Logique d'affaire	7
8.1	Anonyme	7
8.2	Client	8
8.3	Artiste	8
9	Interface utilisateur	9
9.1	Accueil	9
9.2	Profil d'un artiste	9
9.3	Liste des commandes	10
10	Sécurité du système	10
10.1	Injection SQL	10
10.2	Informations de l'utilisateur	10
11	Travail d'équipe	10
A	Annexe 1 - Tables	11
B	Annexe 2 - Gachettes	13
C	Annexe 3 - Interfaces	15

1 Énonciation du problème et de ses exigences

L'application est une boutique en ligne spécialisée dans la vente d'objets d'art. Elle permet aux artistes de vendre leurs œuvres et de prendre des commandes afin de créer des objets d'art personnalisés. De plus, l'application se veut être une plateforme où les clients peuvent échanger avec les artistes à propos d'une œuvre via la création d'une commande. Cela permet aux deux parties de s'entendre sur un prix ou encore de discuter la création d'une œuvre originale basée sur les demandes particulières d'un client.

Listes des exigences :

- Chaque œuvre doit posséder une fiche détaillée
- Une galerie des objets d'art en exposition doit être visible auprès d'un visiteur du site
- Une liste des artistes qui exposent sur le site doit être visible auprès d'un visiteur du site
- Un visiteur doit pouvoir créer un compte client
- Une fois créer, le client doit pouvoir se connecter à son compte
- Un compte client doit permettre de devenir artiste sur le site
- Un compte client doit permettre de commander auprès d'un artiste
- Une commande peut permettre de réserver une œuvre ou de demander la création d'une nouvelle œuvre
- Un compte artiste doit permettre de vendre des œuvres et de recevoir des commandes
- Existence d'une fonctionnalité permettant la mise en relation des clients avec les artistes dans le cadre d'une commande
- Un artiste est un client possédant les droits d'ajouter des objets d'art sur le site, de superviser une commande et d'exposer ses œuvres
- Une fois créée, la commande ne peut être supprimée. Par contre, l'artiste peut l'annuler à sa discrétion en changeant son statut
- Les attributs d'une commande, autres que sa clé primaire, restent modifiables tant que le statut n'a pas été modifié à "En cours". Cela permet à l'artiste de modifier le prix suite à une négociation avec le client
- Une fois la commande "complétée" ou "en cours" de création, une facture doit pouvoir être créée à des fins de dépôt ou de paiement. Ainsi, une commande peut être l'objet de plusieurs factures

2 Modèle entité-relation du système

Le modèle entité-relation(ER) ci-dessous a été élaboré comme point de départ pour la modélisation de la base de données. D'abord, le lien entre **Client** et **Artiste** a été modélisée par une spécialisation à une branche, ce qui permet de relier la relation **Artiste** à d'autres relations qui lui sont spécifiques telles que **Supervise** et **Crée** . Cette spécialisation permet également à chaque artiste de commenter sur une commande étant donné qu'il est également un client. Une contrainte de participation a également été imposée sur la relation **Commande** afin que seuls l'artiste et le client concernés par une commande puissent commenter celle-ci. De manière similaire, une contrainte de cardinalité sur **Commande** assure qu'au maximum un artiste et un client puissent en faire la demande ou la superviser, respectivement. Une relation un-à-un est aussi présente entre **Commande** et **Oeuvre** ce qui assure également que les commentaires qui sont faits sur une commande doivent concerner qu'une seule œuvre. Cela est imposé par le contexte de l'application dans lequel la majorité des clients ne commanderont qu'une œuvre à la fois. Finalement, une facture n'est attachée qu'à une seule commande tandis qu'à l'inverse, une commande peut faire l'objet de plusieurs factures, tel que mentionné par les exigences ci-dessus.

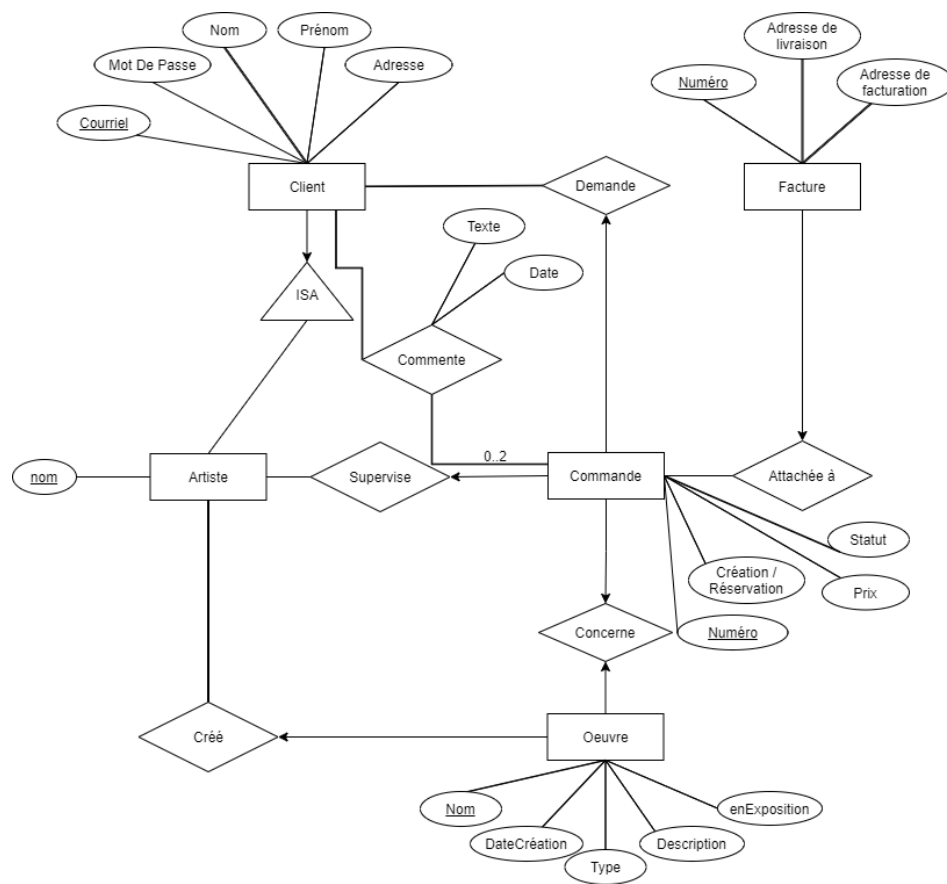


FIGURE 1 – Modèle entité-relation

3 Modèle relationnel

Certaines relations du modèle entité-relation n'ont pas été matérialisées lors du passage au modèle relationnel. En effet, les relations ne possédant pas d'attributs telles que **Crée**, **Supervise**, **Demande**, **Concerne** et **Attachée** à ont été éliminées. Cependant **Commente** a été conservée puisque ces attributs **Texte** et **Date** sont porteurs du message d'un client à l'endroit d'une commande. De plus, la table **Utilisateur** a été créée afin d'emmagasiner les mots de passe des utilisateurs de manière sécuritaire. Les relations (tables) suivantes ont été conservées :

- **Utilisateur**(courriel: varchar(64), mdp: varchar(256))
- **Client**(courriel: varchar(64), nom: varchar(32), prenom: varchar(32),
adresse: varchar(64))
- **Artiste**(courriel: varchar(64), nom: varchar(32))
- **Oeuvre**(nom: varchar(64), auteur: varchar(32), dateCreation: date, type: varchar(16),
description: varchar(256), enExposition: int(1))
- **Commande**(num: integer, superviseur: varchar(32), oeuvre: varchar(64),
demandeur: varchar(64), statut: enum('En cours', 'Complétée',
'En attente de confirmation', 'Annulée'), prix: double(6,2),
type: enum('Création', 'Réservation'), adresseLivraison: varchar(64),)
- **Commentaire**(id integer,
auteur: varchar(64), numCommande: integer, texte: varchar(128), creation datetime)
- **Facture**(numFacture: integer, numCommande: integer, adresseFacturation: varchar(64),
total: double(6,2))

4 Création des relations

Les schémas des relations ont été créés dans MySQL à partir du modèle relationnel avec l'ajout des contraintes d'intégrité des clés primaires et étrangères (Voir annexe 1 pour l'ensemble des tables). Pour les tables **Client** et **Utilisateur** une clé primaire a été définie sur l'attribut **courriel**. Une contrainte de clé étrangère a été appliquée sur cet attribut pour les tables **Utilisateur** et **Artiste** avec des mises à jour et des suppression en cascade afin d'assurer le maintien de l'intégrité des informations à travers ces tables. En effet, la suppression d'un compte client doit nécessairement résulter en la suppression des informations du tuple utilisateur correspondant et du compte artiste associé à ce tuple. Le nom d'artiste est l'attribut qui a été choisi comme clé primaire du schéma **Artiste** pour des raisons d'indexation. Ainsi, le mot-clé **UNIQUE** a été appliqué à l'attribut **courriel** afin de respecter la contrainte de clé primaire pour les tuples d'un client/utilisateur. Pour s'assurer de l'unicité d'une oeuvre, la clé primaire a été composée des attributs **nom** et **auteur**, ce dernier référençant le nom de l'artiste l'ayant créée. Un attribut de numérotation a été utilisé pour identifier spécifiquement une commande puisque le nom de l'oeuvre faisant l'objet de la commande peut être indéfini dans le cas où un client fait une demande de création pour une oeuvre. Une contrainte de clé étrangère a été appliquée sur le nom de l'oeuvre, sans toutefois imposer une valeur non-nulle à cet attribut, ce qui permet encore une fois d'offrir la flexibilité de faire une commande sur une oeuvre non-définie et donc ne faisant pas partie des tuples de la table **Oeuvre**. Le demandeur et le superviseur référencent le courriel client et le nom d'artiste respectivement avec la contrainte supplémentaire d'empêcher la suppression d'un tuple commande dans l'éventualité de la suppression d'un compte client afin que l'artiste puisse avoir une historique de ses commandes.

De manière similaire aux commandes, un numéro de facture sert de clé primaire puisque le numéro de commande n'est pas suffisant pour identifier une facture en raison de la plausible existence de plusieurs factures pour une même commande.

Un numéro de commentaire est aussi utilisé afin d'identifier un message car son auteur peut en écrire plusieurs sur une même commande. Ici, l'attribut auteur référence le courriel du client qui peut être celui de l'auteur de l'oeuvre ou celui du demandeur de la commande.

5 Requêtes et routines

5.1 Routines

Plusieurs gâchettes ont été implémentées afin de faire respecter les contraintes d'intégrité des relations.

- `devenirArtiste` Contrainte de spécialisation
- `artisteCmd` Contrainte pour empêcher un artiste de supprimer son compte client s'il a des commandes en cours
- `estCourriel` Contrainte de domaine pour l'attribut courriel de Client
- `clientCmd` Contrainte pour empêcher un client de supprimer son compte alors qu'il a passé au moins une commande.
- `insertOeuvreCréation` d'un tuple oeuvre lorsqu'une commande de type Création est passée
- `nouvOeuvre` Permet de créer automatiquement une oeuvre lorsque l'artiste ajoute un nom d'oeuvre lors d'une mise à jour sur une commande de création
- `commentaireAuteur` Contrainte de participation qui vérifie qu'un commentaire soit écrit par le demandeur ou l'artiste
- `oeuvreCmdLien` Contrainte qui, lors de la suppression d'une oeuvre, vérifie si elle est liée à une commande

6 Indexation et optimisation

Les principaux indexes sont automatiquement générés par le moteur de stockage utilisé (dans le cas présent **InnoDB**) avec les contraintes de clé primaire et de clé étrangère. En effet, un seul index a été déclaré explicitement et ce dernier se trouve dans la table *Oeuvre*. Les explications qui suivent prennent donc cette table en exemple afin d'illustrer efficacement les différents types d'indexes présents dans la base de donnée.

```
1 CREATE TABLE IF NOT EXISTS Oeuvre(  
2     nom varchar(64) NOT NULL,  
3     auteur varchar(32) NOT NULL,  
4     type varchar(16),  
5     ...  
6     PRIMARY KEY(nom, auteur),  
7     FOREIGN KEY (auteur)  
8         REFERENCES Artiste(nom),  
9     INDEX (type)  
10 );
```

Les clés primaires sont automatiquement indexées par mysql en arbre b groupé. Les valeurs de la table *Oeuvre* seront donc groupées selon les attributs *nom*, *auteur* dans cet ordre.

Les clés étrangères et attributs uniques sont aussi automatiquement indexés par le moteur de stockage en arbre b non-groupé. Dans l'exemple de la table *Oeuvre*, cela correspond à la variable *auteur* faisant référence à l'attribut du *nom* dans la table *Artiste*. Il n'y a pas d'attribut unique dans la table *Oeuvre*.

Sur l'ensemble de la base de donnée rendue pour ce projet, la majorité de ces indexes générés automatiquement par mysql et le moteur de stockage ont donc été suffisants pour l'optimisation des requêtes utilisées.

Seul l'attribut *type* pour la table *Oeuvre*, présentée dans l'exemple ci-dessus, utilisé à plusieurs reprises dans des requêtes de recherche est indexé en arbre b non groupé pour optimiser toutes les recherches par type à partir de la barre de recherche.

6.1 Relations Client

Analyse Après la connexion d'un utilisateur, pour chaque requête, une vérification de l'utilisateur est faite. Une requête d'égalité avec l'attribut *courriel* est donc effectuée à chaque requête au serveur.

Application Pour cette raison, un index en arbre B non-groupé sur l'attribut *courriel* est nécessaire. Déjà définie comme clé primaire après réflexion sur le modèle relationnel, la contrainte de *PRIMARYKEY* ajoutée à l'attribut est donc suffisante. Les autres attributs de la relation sont secondaires et ne nécessitent donc pas d'index.

Remarque En principe, un index de hashage aurait été plus pertinent dans cette situation. Seulement, le moteur de stockage permet uniquement de définir des indexes en arbre B. De plus, il apparaît qu'un index hashé et un index en arbre B d'hauteur 1 aient des temps similaires sur l'ensemble des comparaisons de structure.

6.2 Relation Artiste

Analyse Avec la relation de **Client**, la relation **Artiste** est sûrement celle qui sera la plus interrogée dans la base de donnée.

Premièrement, pour l'authentification, un artiste a plus de droit qu'un client mais reste avant tout un client. Notamment, il lui est possible de créer, modifier et supprimer des oeuvres. Ainsi, pour chaque requête, une requête d'égalité avec le courriel à la table des artistes est d'abord effectuée avant celle des clients.

Deuxièmement, l'attribut de nom d'artiste est une clé de recherche fréquente pour la requête sur la page d'accueil qui effectue un balayage de la table.

Application Étant donné qu'il n'y a que des requêtes d'égalité sur le courriel, un index non-groupé hashé suffit, contrairement au nom pour lequel il est important d'avoir un index **groupé**. Pour cette raison, il est nécessaire que ce soit l'attribut *nom* qui obtienne la contrainte de *PRIMARYKEY*.

6.3 Relation Oeuvre

Analyse Les attributs de *nom*, *auteur* et *type* sont des clés de recherche fréquentes pour la requête sur la page d'accueil qui effectue un balayage de la table. De plus le nom est aussi une clé de recherche pour une requête d'égalité pour retourner l'ensemble des oeuvres d'un artiste.

Application La clé primaire de la relation *nom, auteur* permet d’avoir un index en arbre B groupé. La clé étrangère *auteur* bénéficie d’un index non-groupé en arbre B. Enfin, le seul attribut primaire nécessitant un index non-groupé en arbre B est *type*.

6.4 Relation Commande

Analyse Les attributs de la relation *Commande* *superviseur* et *demandeur* sont des clé pour la requête d’égalité qui permet de retourner l’ensemble des commandes pour un artiste ou pour un client.

Application Les 2 attributs cités ci-dessus sont des clés étrangères et sont donc automatiquement indexés par mysql de manière non-groupé en arbre B.

6.5 Relation Commentaire

Analyse Chaque commande possède un certain nombre de commentaires qu’il faut afficher les uns à la suite des autres dans l’ordre.

Application La clé primaire indexée est un integer qui incrémente. Étant la clé primaire, un index groupé en arbre B lui est automatiquement attribué. Ainsi pour chaque requête, les commentaires seront affichés dans l’ordre de leur création. De plus l’index la clé étrangère sur l’attribut *numCommande* permet de traiter efficacement la requête d’égalité retournant l’ensemble des commentaires pour une commande.

6.6 Relation Facture

Analyse Les factures sont directement rattachées à une commande. Ainsi, la seule manière dont on les récupérera sera en effectuant une requête d’égalité sur l’attribut *numCommande*

Application Cet attribut étant une clé étrangère, un index non-groupé lui est attribué et suffit pour l’ensemble de la relation.

7 Normalisation des relations

7.1 Relation Client

L’absence de dépendances autres que la dépendance de la clé primaire, qui permet de prédire tous les attributs de la relation, confirme l’atteinte de la forme normale Boyce-Codd (FNBC).

7.2 Relation Artiste

La table étant déjà décomposée au minimum, FNBC est atteint.

7.3 Relation Oeuvre

Seul la combinaison des attributs *nom* et *auteur* permettent de prédire d’autres attributs. Cela respecte les critères de la FNBC.

7.4 Relation Commentaire

Hormis l'identifiant du commentaire, la seule combinaison d'attribut permettant de prédire d'autres attributs est une clé candidate qui comprend les quatres attributs autres que la clé primaire. La FNBC est donc respectée.

7.5 Relation Commande

En apparence, il existe une dépendance fonctionnelle $Oeuvre \rightarrow Superviseur$, soit d'un attribut faisant partie d'une clé candidate vers un attribut non clé. Or, certaines commandes de type création peuvent être Null pour l'attribut oeuvre dans le cas où un artiste prend une commande personnalisée sur laquelle aucun titre d'oeuvre n'a été spécifié. Ainsi, la dépendance fonctionnelle mentionnée précédemment n'est pas représentative du monde réel et la forme normale FNBC est atteinte.

7.6 Relation Facture

La relation **Facture** a été modifiée suite à la normalisation. Initialement, l'adresse de livraison était un attribut de cette relation. Cependant, l'analyse de la relation a révélé la dépendance transitive $numCommande \rightarrow adresseLivraison$ qui violait la 3e forme normale et la forme normale Boyce-Codd (FNBC). En effet, une oeuvre n'est livrée qu'à un seul endroit, et cela même si elle est payée en plusieurs factures. Afin de ne pas créer une table supplémentaire, l'adresse de livraison a été ajoutée comme attribut de la relation **Commande**, sans toutefois modifier la forme normale de cette relation.

8 Logique d'affaire

Le serveur d'application est l'interface entre le client (ou page web) et les données nécessaires à son fonctionnement. Il est donc notamment l'interface entre la base de donnée relationnelle et le client.

L'approche choisie pour ce projet cherche à traduire le système de relation avec des classes qui fournissent un ensemble de requêtes sql permettant de traiter une relation chacune. Ces dernières seront par la suite utilisées pour mettre en place les réponses du serveur.

Ainsi les restrictions sur les types ainsi que la vérification de la validité des valeurs est laissé au serveur de base de donnée (mysql). La logique applicative se contentera d'exécuter une suite de requêtes disponibles pour chaque relation.

8.1 Anonyme

Malgré qu'un utilisateur ne soit pas identifié, de la même manière que sur Amazon ou d'autre magasin de vente, il lui est possible de parcourir le catalogue des oeuvres. Ainsi le serveur d'application permet sans aucune authentification nécessaire un certain nombre d'action.

8.1.1 Authentification

En premier lieu, les requêtes d'authentification permettent d'avoir les droits d'un client (voir d'artiste) en plus de ceux d'anonyme. Le système de session (stockage, chiffrement et décription du token notamment) est géré par la librairie externe flask-login.

- La connection : `’/connection’` (POST) prend un courriel et un mot de passe `’mdp’` et ouvre une session artiste ou client en fonction de la présence ou de l’absence d’une spécialisation artiste de l’utilisateur
- La création de compte : `’/creer/_compte’` (POST) prend un courriel et un mot de passe comme pour la connection mais prend en plus un nom, prénom et adresse pour pouvoir créer un nouveau client et un nouvel utilisateur. Cette requête ouvre une session client (artiste n’est pas possible).

8.1.2 Artiste et oeuvres

Il est possible de récupérer un certain nombre d’informations sur les artistes qui sont les vendeurs du site ainsi que leur oeuvres. Ainsi il est possible de :

- Récupérer la liste des artistes : `’/artiste’` (GET)
- Récupérer la liste des oeuvres : `’/oeuvre’` (GET)
- Faire une recherche parmi les oeuvres : `’/search’` (GET) avec des *query parameters type* (Artiste, Oeuvre ou Type) qui sélectionne l’attribut de la table pertinent et *recherche* qui contient le texte saisi dans la barre de recherche.

8.2 Client

Le client a accès au requête de la section précédente en plus d’autres qui lui sont propres.

8.2.1 Authentification

- Devenir un artiste : `’/artiste/devenir’` (POST) prend un nom. La requête traduit la session de client en session d’artiste.
- Déconnexion : `’/deconnection’` (PUT) Supprime la session enregistrée.

8.2.2 Commandes

En tant que client, l’utilisateur peut maintenant prendre commande auprès d’un artiste pour une oeuvre créée ou à créer.

- Récupération de tous les commentaires d’une commande : `’/commande/<numCommande>/commentaires’` (GET) avec numCommandes qui correspond au numéro de la commande pour laquelle on recherche les commentaires
- Créer un commentaire `’/commande/<numCommande>/ajouteCommentaire’` (POST) permet de créer un nouveau commentaire pour la commande avec un numéro numCommande, l’auteur correspond au client connecté avec le texte qui correspond au texte donnée dans le corps de la requête

8.3 Artiste

Artiste étant un client possédant des oeuvres qu’il peut vendre, il lui est possible de faire les mêmes requêtes que le client. Il peut supprimer son compte avec `’/artiste/finir’` (DELETE) ce qui supprimera son compte artiste si celui-ci n’a jamais supervisé de commandes. Il redeviendra alors simple client.

8.3.1 Oeuvres

- Créer une oeuvre : `’/oeuvre/creer’` (POST) permet de créer une nouvelle oeuvre qui prend un nom, une date de création (`dateCreation`), un type et une description
- Supprimer une oeuvre : `’/oeuvre/supprimer’` (DELETE) supprime une oeuvre dont l’auteur est l’artiste connecté et le nom est le nom donné dans le corps de la requête

9 Interface utilisateur

Les différentes pages d’interface se trouvent dans la troisième annexe.

L’interface utilisateur possède 3 pages principales. Chacune de ses pages affichent une entête de page qui est commune à toutes les pages. Cette dernière est composée principalement d’un bouton déroulant ou non de connexion. En fonction du type de connexion utilisateur, soit en tant que Client ou Artiste ou seulement anonyme, ce bouton n’est pas le même et ne permet pas les mêmes actions.

Sans être connecté Seul un bouton permettant de se connecter est présent. Ce bouton ouvre alors une modale contenant un formulaire de connexion et un formulaire de création de compte.

En tant que client le bouton devient un menu déroulant comptenant un bouton de déconnexion, un bouton redirigeant vers sa page de commande et enfin un bouton pour devenir artiste.

En tant qu’artiste de la même manière que pour le client, le bouton est alors un menu déroulant. Cette fois-ci les options sont le profil d’artiste qui redirige vers la page de son profil et le bouton de déconnection.

9.1 Accueil

La page d’accueil présente une barre de recherche et deux onglets (Oeuvres et Artistes). Le premier onglet affiche la liste des oeuvres avec l’ensemble de leur attribut, que les artistes ont choisi d’exposer. On peut faire une recherche parmi ces oeuvres via la barre de recherche et le bouton déroulant qui permet de sélectionner le niveau de la recherche (par type, nom ou encore par auteur). Le second onglet correspond à la liste des artistes. Cette liste s’actualise pour la liste des artistes dont les oeuvres sont affichées dans le premier onglet après la première recherche.

9.2 Profil d’un artiste

Le profil d’artiste s’affiche lorsque l’utilisateur clique sur la carte d’un des artistes ou lorsqu’un artiste clique sur son profil. Les fonctionnalités présentent sur le profil de l’artiste dépendent de l’utilisateur qui y accèdent. En effet, si le client n’est pas l’artiste du profile d’artiste en question, alors ne il pourra voir que le nom de l’artiste et la liste de ces oeuvres. Autrement, si l’artiste est connecté et accède à son profil d’artiste, il aura la possibilité d’interagir avec deux boutons qui lui permettent d’ajouter une oeuvre et de supprimer son compte d’artiste, respectivement. Il peut également voir la liste de ces oeuvres et voir si celles-ci sont exposées sur la page d’accueil. Il peut aussi les supprimer via un bouton situé sur chaque carte d’oeuvre.

9.3 Liste des commandes

La page correspondant à la liste des commandes n'est atteignable que par un utilisateur connecté en tant que client. Elle affiche la liste des commandes à gauche du site et le contenu de la commande sélectionnée à droite.

10 Sécurité du système

10.1 Injection SQL

Afin de prévenir les injections SQL, certaines mesures ont été mises en place. D'abord, les requêtes exécutées par le curseur de la connection à la base de donnée ont été écrites de manière à ce que les variables de requête soient passées comme paramètres, ce qui permet d'échapper aux caractères qui créent une porte d'entrée pour les injections SQL en mettant fin à une requête et en insérant une requête non désirée. D'autres part, des limites de caractères ont été imposées aux champs de texte du côté client afin de limiter les chances que le texte passé par le client contienne une requête SQL.

10.2 Informations de l'utilisateur

Afin d'assurer un minimum de sécurité, les mots de passe des utilisateurs sont hashés par un algorithme de hashage utilisant une clé de hashage de 256-bit (Sha256) et sont ensuite stockés dans une table contenant uniquement le courriel permettant d'identifier d'associer le mot de passe associé à son possesseur. De cette manière, le mot de passe saisi par le client lors de l'authentification est hashé sur le serveur d'application avant d'être comparé au mot de passe associé au tuple de l'adresse courriel de l'utilisateur.

11 Travail d'équipe

Pour pouvoir avancer sur le projet de manière efficace et régulière, chaque semaine avait lieu une réunion. Cette dernière permettait en premier de présenter les avancements et de mettre en commun le travail fait pendant la semaine. Ensuite, cela nous permettait de prendre les décisions de design importante à l'aide de la matière du cours et de l'intégrer à l'application. Finalement, les rencontres se concluaient sur une distribution des tâches à faire pour la semaine suivante.

En utilisant un outil mis à notre disposition par github, nous avons été en mesure de lier nos "issues" (mot propre à github pour désigner des sujets liés à un repository) à un tableau de tâches qui nous permettait de visualiser l'avancement du projet.

La distribution des tâches s'est également effectuée de manière à optimiser le temps à disposition en exploitant les forces de chaque membre de l'équipe, tout en permettant aux deux membres de toucher à tous les aspects du projet.

A Annexe 1 - Tables

```
1 CREATE TABLE Client(  
2     courriel varchar(64) PRIMARY KEY,  
3     nom varchar(32),  
4     prenom varchar(32),  
5     adresse varchar(64)  
6 );  
7  
8 CREATE TABLE Utilisateur(  
9     courriel varchar(64) PRIMARY KEY,  
10    mdp varchar(256) NOT NULL,  
11    FOREIGN KEY(courriel)  
12        REFERENCES Client(courriel)  
13        ON DELETE CASCADE  
14        ON UPDATE CASCADE  
15 );  
16 CREATE TABLE Artiste(  
17     courriel varchar(64) UNIQUE NOT NULL,  
18     nom varchar(32) PRIMARY KEY,  
19     FOREIGN KEY(courriel)  
20        REFERENCES Client(courriel)  
21        ON DELETE CASCADE  
22        ON UPDATE CASCADE  
23 );  
24 CREATE TABLE Oeuvre(  
25     nom varchar(64) NOT NULL,  
26     auteur varchar(32) NOT NULL,  
27     dateCreation date,  
28     type varchar(16),  
29     description varchar(256),  
30     enExposition int(1) DEFAULT FALSE,  
31     PRIMARY KEY(nom, auteur),  
32     FOREIGN KEY (auteur)  
33        REFERENCES Artiste(nom),  
34     INDEX (type)  
35 );  
36 CREATE TABLE Commande(  
37     num integer AUTO_INCREMENT PRIMARY KEY,  
38     oeuvre varchar(64),  
39     superviseur varchar(64) NOT NULL,  
40     demandeur varchar(64) NOT NULL,  
41     statut enum('En cours', 'Completee', 'En attente de confirmation', 'Annulee')  
42     DEFAULT 'En cours',  
43     prix double(6,2),  
44     type enum('Creation', 'Reservation') NOT NULL,  
45     adresseLivraison varchar(64),  
46     FOREIGN KEY(oeuvre)  
47        REFERENCES Oeuvre(nom),  
48     FOREIGN KEY(superviseur)  
49        REFERENCES Artiste(nom),  
50     FOREIGN KEY(demandeur)  
51        REFERENCES Client(courriel)  
52        ON DELETE NO ACTION  
53        ON UPDATE CASCADE  
54 );  
55 CREATE TABLE Facture(  
    numFacture integer AUTO_INCREMENT PRIMARY KEY,
```

```

56     numCommande integer NOT NULL,
57     adresseFacturation varchar(64) NOT NULL,
58     total double(6,2) NOT NULL,
59     FOREIGN KEY(numCommande)
60         REFERENCES Commande(num)
61 );
62 CREATE TABLE Commentaire(
63     id integer AUTO_INCREMENT PRIMARY KEY,
64     auteur varchar(64) NOT NULL,
65     numCommande integer NOT NULL,
66     texte varchar(128) NOT NULL,
67     creation date DEFAULT (CURRENT_DATE),
68     FOREIGN KEY(auteur)
69         REFERENCES Client(courriel),
70     FOREIGN KEY(numCommande)
71         REFERENCES Commande(num)
72 );

```

B Annexe 2 - Gachettes

```
1 DELIMITER //
2 — Contrainte de specialisation
3 CREATE TRIGGER devenirArtiste
4 BEFORE INSERT ON Artiste
5 FOR EACH ROW
6 BEGIN
7     IF NOT EXISTS (SELECT courriel FROM Client WHERE courriel = NEW.courriel)
8     THEN
9         SIGNAL SQLSTATE '45000 '
10        SET MESSAGE_TEXT = 'Vous devez creer un compte client avant de devenir un
11        artiste';
12        END IF;
13    END; //
14 — Contrainte de domaine pour attribut courriel de Client
15 CREATE TRIGGER estCourriel
16 BEFORE INSERT ON Client
17 FOR EACH ROW
18 BEGIN
19     IF (NEW.courriel NOT LIKE '%@%')
20     THEN
21         SIGNAL SQLSTATE '45000 '
22         SET MESSAGE_TEXT = 'Le courriel que vous avez entre est invalide';
23         END IF;
24     END; //
25 — Contrainte pour verifier qu'un commentaire soit ecrit par le demandeur ou l'
26     artiste
27 CREATE TRIGGER commentaireAuteur
28 BEFORE INSERT ON Commentaire
29 FOR EACH ROW
30 BEGIN
31     IF NEW.auteur NOT IN (SELECT C.demandeur FROM Commande C WHERE C.num = NEW.
32     numCommande) — Si l'auteur du commentaire est le demandeur
33     AND NEW.auteur NOT IN (SELECT A.courriel FROM Commande C, Artiste A WHERE
34     NEW.numCommande = C.num AND C.superviseur = A.nom) — Si l'auteur du commentaire
35     est le superviseur
36     THEN
37         SIGNAL SQLSTATE '45000 '
38         SET MESSAGE_TEXT = 'Ajout du commentaire interdit: auteur invalide';
39         END IF;
40     END; //
41 — Contrainte pour empecher un artiste de supprimer son compte client s'il a des
42     commandes en cours
43 CREATE TRIGGER artisteCmd
44 BEFORE DELETE ON Artiste
45 FOR EACH ROW
46 BEGIN
47     IF OLD.nom IN (SELECT C.superviseur FROM Commande C)
48     THEN
49         SIGNAL SQLSTATE '45000 '
50         SET MESSAGE_TEXT = 'Supression du compte artiste interdite: commandes en
51         cours';
52         END IF;
53     END; //
```

```

50
51 — Contrainte pour empecher un client de supprimer son compte alors qu'il a passe au
    moins une cmd
52 CREATE TRIGGER clientCmd
53     BEFORE DELETE ON Client
54     FOR EACH ROW
55     BEGIN
56         IF OLD.courriel IN (SELECT C.demandeur FROM Commande C)
57         THEN
58             SIGNAL SQLSTATE '45000'
59             SET MESSAGE_TEXT = 'Supression du compte client interdite: commandes en
    cours';
60         END IF;
61     END; //
62
63 — Creation d'un tuple oeuvre lorsqu'une commande de type Creation est passee
64 CREATE TRIGGER nouvOeuvre
65     BEFORE UPDATE ON Commande
66     FOR EACH ROW
67     BEGIN
68         IF OLD.oeuvre IS NULL AND NEW.oeuvre IS NOT NULL AND NEW.oeuvre NOT IN (
69         SELECT O.nom FROM Oeuvre O) AND NEW.type = 'creation'
70         THEN
71             INSERT INTO Oeuvre(nom, auteur) VALUES (NEW.oeuvre, NEW.superviseur);
72             END IF;
73         END; //
74
75 CREATE TRIGGER insertOeuvre
76     BEFORE INSERT ON Commande
77     FOR EACH ROW
78     BEGIN
79         IF NEW.oeuvre IS NOT NULL AND NEW.oeuvre NOT IN (SELECT O.nom FROM Oeuvre O
80         ) AND NEW.type = 'creation'
81         THEN
82             INSERT INTO Oeuvre(nom, auteur) VALUES (NEW.oeuvre, NEW.superviseur);
83             END IF;
84         END; //
85
86 — Contrainte lors de la suppression d'une oeuvre, si elle est liee a une commande
87 CREATE TRIGGER oeuvreCmdLien
88     BEFORE DELETE ON Oeuvre
89     FOR EACH ROW
90     BEGIN
91         IF EXISTS (SELECT * FROM Commande C WHERE C.superviseur = OLD.auteur AND C.
92         oeuvre = OLD.nom)
93         THEN
94             SIGNAL SQLSTATE '45000'
95             SET MESSAGE_TEXT = 'Suppression de l\'oeuvre interdite: liee a une
    commande';
96         end if;
97     END; //
    DELIMITER ;

```


C Annexe 3 - Interfaces

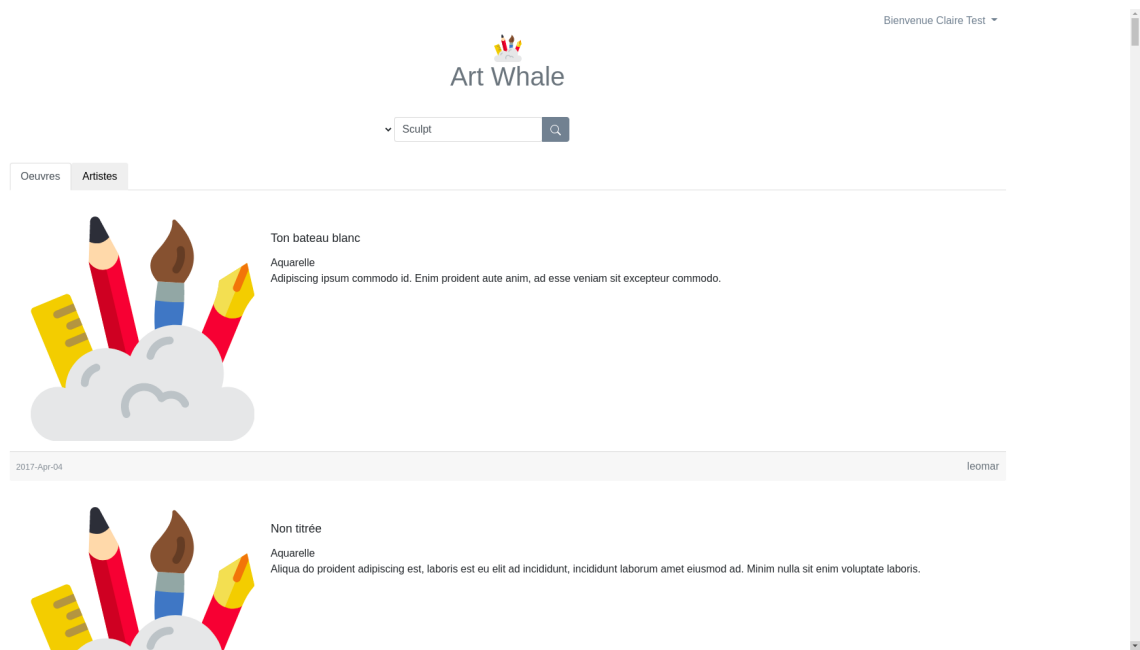




FIGURE 2 – Page d'accueil

 Art Whale
 Bienvenue Claire Test ▾

Test
Ajouter une oeuvre
Quitter le monde artistique



La Joconde


exposée

Tableau

La Joconde, ou Portrait de Mona Lisa, est un tableau de Léonard de Vinci représente un portrait mi-corps, probablement celui de la Florentine Lisa Gherardini, épouse de Francesco del Giocondo.

Supprimer cette oeuvre

1503-Jan-01Test




La luna

non exposée


Musique

Cette oeuvre est pas exposée

FIGURE 3 – Profil d'artiste

 Art Whale
 Bienvenue Claire Test ▾

Mes commandes




151
rayfou

En cours

Commenter

Ajouter



152
joslac

Complètee

FIGURE 4 – Mes commandes

16