Plante, Samuel
111 183 751

Maude Paradis
111 178 289

Vanessa Godon
111 156 057

Modèles et langages des bases de données pour ingénieurs
GLO-2005 – Section A

Projet de session

Travail présenté à
M. Richard Khoury

Département d'informatique et de génie logiciel
Université Laval
Hiver 2018

Problème et exigences

Contexte

Nous allons développer une boutique en ligne pour une artiste locale. Mme Nathalie Côté, notre cliente, est une graphiste qui fabrique des produits artisanaux.

Nous avons donc comme mandat de développer un site web qui met en valeur les produits de notre cliente et qui informe les utilisateurs sur les produits et sur l'artiste. Comme le site web doit être personnalisé pour Mme Côté, nous devons déterminer certaines contraintes afin qu'il respecte les objectifs de notre cliente.

Exigences

Le site web comporte donc les pages suivantes : l'accueil, la boutique qui comporte autant de catégories que de types de produits vendus, le formulaire de création et de connexion du compte, le panier de l'acheteur, la page personnelle de l'acheteur, une page pour décrire les travaux de la cliente ainsi que son parcours et une page qui énumère les façons d'entrer en contact avec la cliente.

L'accueil doit comporter quelques réalisations faites par l'artiste afin de donner à l'acheteur un aperçu du style de notre cliente.

La boutique, quant à elle est divisée en sous-sections : une pour chaque type de produit confectionné par Mme Côté. Quand l'utilisateur sélectionne une catégorie de produit, un aperçu de tous les produits vendus est disponible, puis, en cliquant, l'acheteur a accès à une fiche détaillée du produit. Il est important de mettre de l'avant les produits de la cliente et le soin avec lequel elle choisit les matières premières, les procédés d'impression, etc. Chaque fiche décrit donc, de manière exhaustive, le produit afin que l'utilisateur ait le plus de détails sur son futur achat.

Les catégories de produits disponibles pour le moment sont des cartes, des coussins et des serviettes de table.

En ce qui concerne les cartes, il est important de préciser le prix à l'unité, le type de carton, le message à l'intérieur de la carte, le format, le procédé d'impression, le motif et le nom du produit. Il faut également qu'une photo soit disponible afin que l'acheteur puisse voir le design de la carte.

Pour les coussins, il faut décrire le type de tissu, le procédé d'impression, la peinture utilisée pour imprimer l'image, le prix à l'unité, le format, le motif et le nom. Une image du produit est également présentée.

Ensuite, pour les serviettes de table, il faut préciser le procédé d'impression, le type de tissu, la peinture utilisée pour imprimer l'image, le motif imprimé, le prix à l'unité, le format et le nom. Une image donne un aperçu des serviettes de table.

La disponibilité des produits n'est pas un enjeu comme notre cliente prend en considération les commandes pour créer son stock de produits.

Les acheteurs peuvent ajouter au panier les produits à partir de leur fiche détaillée.

Le panier présente les produits qu'il contient ainsi que le total de la facture. Les produits qui ne sont plus désirés peuvent être retirés. Les acheteurs peuvent confirmer leur achat et ensuite ils sont

redirigés pour confirmer les informations nécessaires au paiement et à la livraison si l'acheteur possède un compte. Sinon, il sera invité à s'en créer un.

Il est à noter que les livraisons se font seulement au Québec et à domicile.

Un formulaire est donc disponible pour l'acheteur afin qu'il se crée un compte. Cela lui permet d'acheter des produits et de consulter ses factures antérieures.

Spécifications et responsabilités

D'abord, il va sans dire que notre application a une architecture à trois niveaux. Afin de nous faciliter la tâche de décision sur les responsabilités de chacun, nous suivons les lignes directrices suivantes :

- Toutes manipulations concernant des données privées auxquels l'usager ne doit pas avoir accès sont exécutés sur le serveur. (Accès base de données, connexion, gestion de compte, etc.)
- Le client a pour but d'effectuer des validations sur les champs d'un formulaire avant l'envoi et d'afficher l'information reçue de la couche applicative.

1. Naviguer dans les catalogues de produits

Serveur

Pour cette fonctionnalité, le serveur doit récupérer la liste des produits répertoriés dans la base de données. Il doit appliquer les filtres sélectionnés par l'utilisateur dans sa recherche, pour retourner seulement l'information pertinente. Une fois qu'il a recueilli le tout, il transforme chaque entrée et l'envoie au client.

Client

Ici, le client a pour tâche de recevoir et d'afficher l'information pertinente du serveur.

2. Enregistrer un compte utilisateur

Serveur

Le serveur reçoit les informations entrées dans le formulaire par l'utilisateur. Il doit ensuite valider que l'utilisateur n'existe pas déjà et ajouter ensuite son compte à la base de données. Pour plus de sécurité, le mot de passe de l'utilisateur est haché avant l'insertion dans la base de données. On y ajoute un grain de sel connu uniquement par le serveur pour renforcer encore la sécurité. Une fois que le compte est créé, le serveur redirige l'usager vers la page de connexion du site.

Client

Le client a comme tâche de recueillir l'information entrée dans le formulaire et de s'assurer que chacun des champs est une entrée valide. Une fois que toutes les données sont valides, il envoie le formulaire au serveur qui va prendre en charge la suite du processus de création du compte.

3. Se connecter à un compte utilisateur

Serveur

Ici, la tâche du serveur est de valider l'information reçue du client et de s'assurer que les informations entrées sont bien enregistrées dans la base de données. Pour ce faire, il doit calculer la valeur de hachage du mot de passe entré et valider qu'il concorde avec la valeur inscrite dans le tuple de la base de données associé à l'utilisateur. Si ça concorde, il crée la variable de session pour l'utilisateur et le redirige vers la page de son compte. Sinon, il envoie une erreur de connexion au client.

Client

Le client doit valider que le code d'usager et le mot de passe sont bien remplis dans le formulaire avant d'envoyer l'information au serveur.

4. Modifier les informations du compte

Serveur

Ici, le serveur a le même rôle que précédemment, soit de mettre à jour la base de données en cas de besoin. Seul l'identifiant du compte ne peut pas être mis à jour.

Client

Encore une fois, le client valide les entrées du formulaire de mise à jour pour s'assurer que toute l'information est valide et envoie les modifications au serveur.

5. Ajouter un article au panier

Serveur

Le serveur ajoute une entrée dans la base de données pour répertorier le produit dans le panier de l'utilisateur. Ici, l'utilisateur a toujours un panier associé à son compte, mais celui-ci peut être vide.

Client

Le client transmet l'information sur le produit à ajouter au panier de l'utilisateur et la quantité de celui-ci.

6. Visualiser les entrées du panier

Serveur

Le serveur recueille la liste des objets dans le panier de l'utilisateur et l'information sur ces produits. Il génère l'information pour qu'elle puisse être affichée par le client et ajoute des liens vers les pages d'informations des objets.

Client

Affiche l'information sur les entrées du panier.

7. Retirer un produit du panier

Serveur

Le serveur retire le tuple associé au produit reçu de la table du panier et actualise l'affichage de l'utilisateur.

Client

Envoie le produit à supprimer du panier au serveur.

8. Modifier la quantité d'un produit du panier

Serveur

Le serveur met à jour le tuple associé au produit reçu de la table du panier et actualise l'affichage de l'utilisateur.

Client

Envoie le produit et la nouvelle quantité du produit au serveur.

9. Passer une commande

Serveur

Après avoir validé les informations de paiement, le serveur enregistre l'information de la commande et génère un numéro de commande. Il enregistre aussi les informations de livraison et de paiement associées à la commande. Il crée une nouvelle livraison à envoyer qui va pointer sur le bon de commande créé auparavant.

Client

Le client valide les informations de livraison et les informations de paiement avant d'envoyer l'information au serveur.

10. Visualiser une commande

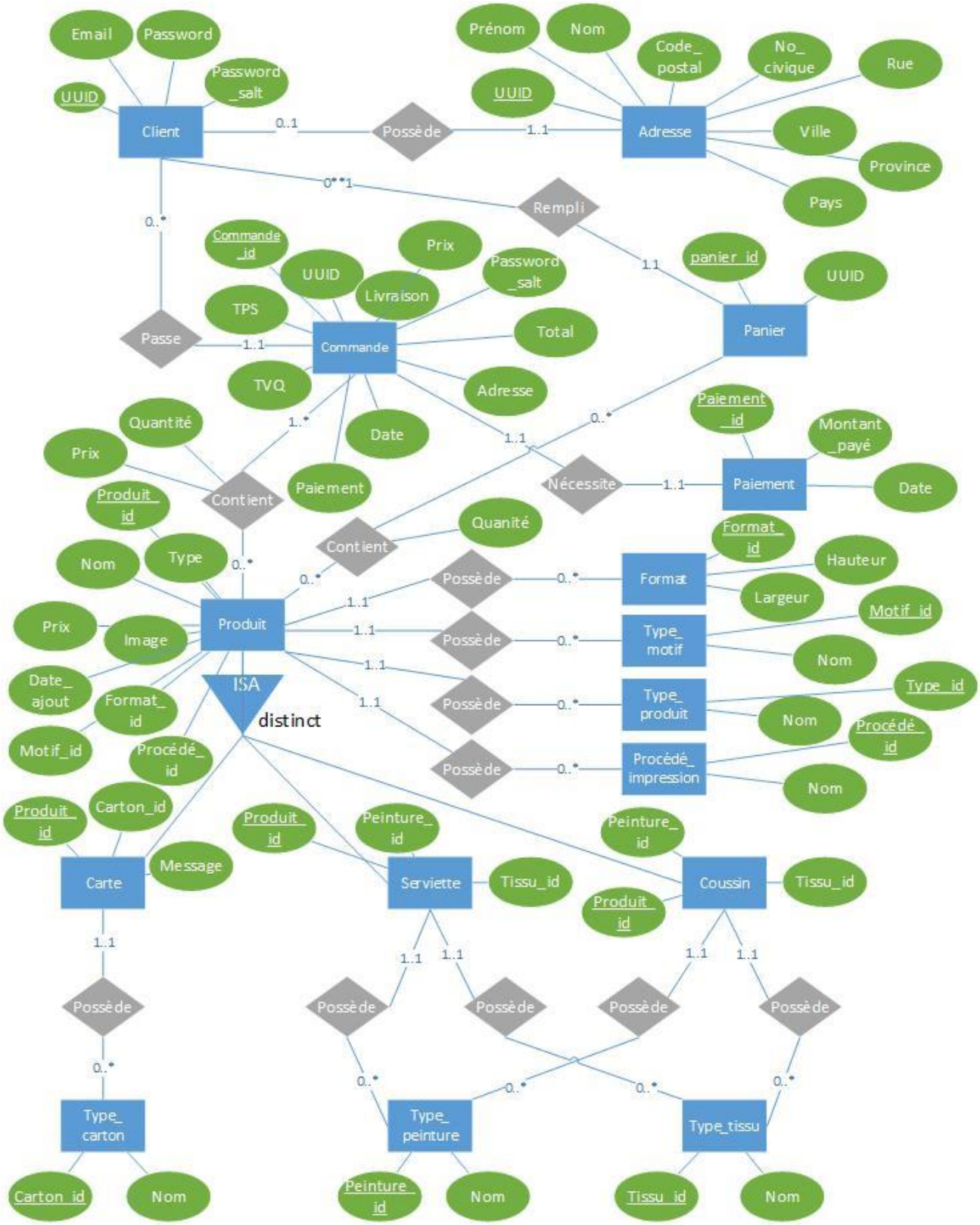
Serveur

À partir du numéro de commande sélectionné, le serveur récupère toutes les informations concernant la commande et envoi l'information au client.

Client

Le client affiche les informations de la commande reçues du serveur.

Modèle entité-relation de la base de données



Modèle relationnel de la base de données

Schémas

- Client (UUID : varchar(36), email : varchar(254), password : varchar(64), password_salt : varchar(64))
- Adresse (UUID : varchar(36), prenom : varchar(20), nom : varchar (20), code_postal : varchar(7), no_civique : integer, rue : varchar(20), ville : var char(50), province : varchar(50), pays : varchar(20))
- Produit (produit_id : integer, nom : varchar(40), type : integer, prix : decimal(5,2), image : char(70), date_ajout : date, format_id : integer, motif_id : integer, procede_id : integer)
- Carte (produit_id: integer, carton_id : integer, message : varchar(1000))
- Coussin (produit_id: integer, peinture_id : integer, tissu_id : integer)
- Format (format_id : integer, hauteur : decimal(5, 2), largeur : decimal(5, 2))
- Serviette (produit_id: integer, peinture_id : integer, tissu_id : integer)
- TypeMotif (motif_id : integer, nom : char(50))
- TypeCarton (carton_id : integer, nom : char(20))
- TypeTissu (tissu_id:integer, nom char(70))
- ProcedeImpression (procede_id : integer, nom : char(20))
- TypePeinture (peinture_id : integer, nom : char(20))
- Panier (panier_id : integer auto increment, UUID: char(36))
- Produit_Panier (panier_id : integer, produit_id: integer, quantite : integer)
- Commandes (commande_id: integer auto increment, UUID : char(36), prix : decimal(5, 2), livraison : decimal(5,2), tps : decimal(5, 2), tvq : decimal(5, 2), total : decimal(8, 2), date : date, paiement : integer, adresse : varchar(100))
- Paiement (paiement_id : integer auto_increment, montant_paye : decimal(6,2), date : dateTime)
- Produit_Commande (commande_id : integer, produit_id: integer, quantite : integer, prix : decimal(5, 2))

Il est important de mentionner que le montant du paiement est équivalent au montant de la commande puisque nous tenons pour acquis que le client paye tout dès qu'il passe sa commande. Nous ne gardons pas et ne vérifions pas les données de paiement comme aucun vrai paiement n'est passé.

De plus, les relations Produit_Commande et Produit_Panier correspondent aux relations qui unissent Commandes et Produit, Panier et Produit dans le modèle entités-relations.

Implémentation de la base de données et des requêtes, et indexation et optimisation.

Nous devons effectuer différentes requêtes pour récupérer les tuples nécessaires pour que l'utilisateur puisse naviguer dans le magasin, se connecter, etc. Voici donc les principales contraintes et requêtes qui sont implémentées dans le site web pour les différentes fonctionnalités ainsi que leur index :

1. Requête sur les relations reliées aux produits

Des requêtes de gammes sont effectuées pour récupérer les tuples correspondant aux produits à afficher. Les tuples retrouvés sont différents selon la catégorie de produits sélectionnée. Donc, un

arbre B+ semble tout indiqué. Cependant, il faut également afficher un produit spécifique, et retrouver le tuple du produit pour l'ajouter à la table Produit_Panier. Un index de hachage sur la clé primaire de la relation Produit est tout de même préféré.

En ce qui concerne la table Produit_Panier, nous faisons un index de hachage sur la clé primaire de la relation, soit Produit_id et Panier_id. En ce qui concerne la relation Panier, un index de hachage sur l'attribut UUID est plus pratique étant donné que la recherche se fait sur cet attribut dans les requêtes sur cette relation.

Du côté de la relation Commandes, un index de hachage sur l'attribut UUID est le plus bénéfique pour les requêtes effectuées, car c'est cet attribut qui est utilisé pour retrouver une commande dans les requêtes.

Finalement, les relations qui contiennent de l'information sur des spécifications du produit comme Type_Carton contiennent un index de hachage sur la clé primaire, soit l'identifiant de la relation. Effectivement, aucune requête de gamme ne s'effectue pour ces relations, car pour chaque tuple, il n'y a qu'une valeur associée à la spécification.

2. Requête sur les relations reliées au compte utilisateur

Un index ralentit l'enregistrement du compte d'un utilisateur comme c'est une mise à jour de la base de données, mais il est tout de même nécessaire, un index de hachage plus précisément sur l'attribut UUID de la table Client. Effectivement, il faut récupérer le tuple nécessaire pour la connexion de l'utilisateur. Il en va de même pour la modification des informations du compte. Aucune requête de gamme n'est effectuée sur cette relation, donc c'est pourquoi l'option d'un index en arbre B+ a été mise de côté.

La même approche est employée pour la table Adresse. En effet, il faut retrouver le tuple correspondant à l'adresse du client lors de la transaction, donc un index de hachage sur l'attribut UUID est la meilleure option.

3. Contraintes

Lorsqu'un produit dans la table Produit est supprimé, le tuple référencé dans la relation du type de produit doit être supprimé.

Lorsqu'un Client est supprimé, les tuples référencés dans Adresse doivent être supprimés en cascade. Dans la relation Commande, la valeur de l'identifiant du client est remplacée par la valeur nulle. La suppression du tuple n'est pas désirée, car l'inventaire des commandes est utile pour Mme Côté.

Lorsqu'un tuple d'une relation concernant la spécification d'un produit est supprimé, la relation où le tuple est référencé défend la suppression, car sinon l'état du produit deviendrait incohérent.

De plus, un trigger sur la relation client est implémenté afin qu'il crée, lorsqu'un client est ajouté, un tuple dans la table Panier relié à ce client.

Implémentation de la logique d'affaireS

La logique d'affaires est implémentée en suivant les spécifications énoncées plus haut. Le lien est fait entre la base de données et l'interface utilisateur. Les données sont recueillies de l'interface utilisateur et ajoutées à la base de données et vice-versa.

Implémentation de l'interface utilisateur

L'interface utilisateur est implémentée pour répondre au besoin du client et pour donner une image attirante des produits et de la marque de Mme Côté.

L'interface affiche les données nécessaires pour que le client puisse naviguer sur le site, prendre connaissance des produits, acheter, etc.

De plus, nous avons adopté un style appliqué à toutes les pages afin que le site soit attrayant et représente bien notre cliente. Le design choisi est épuré et simple. Les lignes droites sont privilégiées ainsi que la clarté, donc le blanc est la couleur de fond de tout le site web. Huit couleurs sont privilégiées par notre cliente pour les différents éléments des pages comme les formulaires ou les boutons.

Sécurité du système

Pour assurer un minimum de sécurité, quelques mesures sont implémentées.

Premièrement, nous mettons une limite de caractères pour chaque champ de saisie. De cette manière nous évitons la possibilité d'injection SQL. Aussi, les données critiques ne sont pas passées par le biais d'un formulaire, car il serait simple de modifier la valeur de la page HTML avant d'envoyer l'information au serveur. Donc, pour toutes les données critiques, les valeurs sont recalculées par le serveur (ex : calcul du prix d'une commande).

Deuxièmement, nous appliquons une fonction de hachage sur le mot de passe pour ne pas avoir le mot de passe en clair dans la base de données. Afin d'éviter les collisions, nous utilisons l'algorithme SHA1. Il n'y a pas de fonction de hachage orienté vers les mots de passe dans Python, mais si nous avions pu utiliser un module pour en ajouter nous aurions utilisé une méthode qui a pour but précis de hacher un mot de passe. L'avantage avec ces fonctions spécifiques, c'est que le délai de hachage est plus long qu'un algorithme normal, afin de complexifier les attaques de type force brute.

De plus un grain de sel est ajouté au mot de passe avant le hachage, afin d'empêcher quelqu'un qui récupérerait les données d'utiliser une « Rainbow Table » pour retrouver la valeur du mot de passe. Aussi, toujours dans le but de complexifier les attaques force brute, nous générons un gain de sel différent pour chaque compte et nous l'inscrivons dans la base de données avec le mot de passe. En effet, par sa nature, il n'y a pas d'inconvénient à ce que l'attaquant récupère le grain de sel, car il devra tout de même faire une attaque force brute pour trouver le mot de passe. Comme ce grain de sel est différent pour tous les utilisateurs, il devra lancer une nouvelle attaque pour tous les mots de passe qu'il veut récupérer, ce qui peut prendre des milliers d'années.

Instructions de configuration de la base de données

Afin de pouvoir avoir une version de notre système pour la correction, il faut décompresser l'archive .zip remise avec le projet. Cette archive comporte tous les fichiers nécessaires au fonctionnement du serveur web FLASK ainsi qu'un script python pour la création de la base de données et le chargement des données dans celle-ci.

Configuration

Prérequis

Pour s'exécuter, les deux scripts nécessitent que les modules Flask et PyMySQL soient installés dans python. Il va sans dire que le poste du correcteur doit aussi contenir un serveur MySQL et la dernière version de Python.

Création de la base de données

Il faut tout d'abord utiliser le script nommé « Create database.py ». Ce script crée sur le serveur local du correcteur une base de données intitulée « Lieco » et un utilisateur intitulé « lieco-dbuser ». Si une de ces deux entrées existe déjà sur le serveur, il faudrait penser à les supprimer d'abord, sinon l'exécution du script ne fonctionnera pas.

Une fois cette validation faite, il faut remplir dans le script les identifiants du compte « root » du serveur dans les variables définies à cet effet au début du script. Une fois que c'est fait, on roule le script et chacune des étapes, de la création de la base de données et des différentes tables au chargement des données et terminant par la création d'un trigger devrait s'exécuter. Il est recommandé de lancer le script à partir d'une invite de commandes afin de voir les informations s'afficher. Le script indiquera un message au correcteur si les identifiants entrés ne sont pas corrects.

Pour accéder à tout ce qui se rapporte aux utilisateurs comme leur profil, voici l'identifiant et le mot de passe de l'un de ceux qui font partie de la base de données :

Identifiant : cafe1214@hotmail.com

Mot de passe : v6hZxpX[rev

Lancement du serveur FLASK

Il suffit par la suite de lancer le script « __init__ » pour lancer le serveur.