Welift

Modèles et langages des bases de données pour ingénieurs

GLO-2005

Présenté à

Richard Khoury

Par
Équipe 28

Matricules	Noms
111 237 153	Jean-Christophe Drouin
111 239 483	Vincent Lambert
111 239 017	Yu Xuan Zhao

Faculté des Sciences et génies

Université Laval

05 mai 2020

Table des matières

Problème et exigences

Dans le cadre de ce projet, nous avons décidé de créer une application de covoiturage qui servira aux voyageurs à travers le Québec de se déplacer de manière simple et écologique. L'utilisateur pourra soit bénéficier du transport de quelqu'un d'autre ou décider de conduire avec son propre véhicule et embarquer des passagers. Le contexte de cette application sera uniquement réservé aux voyageurs seuls ou en petits groupes puisque le nombre de passagers maximum pour un transport ne doivent pas dépasser le nombre de places dans le véhicule du chauffeur. Chaque utilisateur possède un compte qui lui servira pour les transactions. Ces derniers pourront également laisser des rétroactions sur leurs expériences de covoiturage avec d'autres utilisateurs. Les utilisateurs possédant une voiture devront se charger de planifier des trajets tout en spécifiant le type et l'apparence de la voiture qu'ils utiliseront.

Spécifications du système et des responsabilités des trois niveaux

Les fonctionnalités principales du logiciel sont les suivantes: le client commence par créer son compte en y entrant les informations. Ces informations pourront être aperçues dans la page "profil" du site. Ensuite, il pourra décider entre créer un trajet ou chercher ceux qui sont déjà offerts par d'autres conducteurs. S'il décide de créer un trajet, il devra d'abord entrer les informations du voyage, puis les informations de sa voiture. S'il décide de chercher un trajet, le client pourra manipuler des filtres et trouver les trajets qui correspondent à ses demandes. Lorsqu'il confirme qu'il désire embarquer avec un covoitureur, il devra effectuer une transaction. Finalement, à la fin du voyage, le client pourra décider ou non de publier une rétroaction à son conducteur.

Les serveurs et les routes du logiciel seront créés en Python (Flask). La création du compte insèrera dans la base de données les informations du client. La fonction d'authentification utilisera une logique Python (Flask) combinée d'une requête avec la base de données pour

valider l'utilisateur. Cette logique sera également appliquée sur toutes les pages du site où l'accès est seulement autorisé s'il y a une session qui est active. Les fonctions de recherche, de création de trajets, de paiement et de "reviews" utilisent Flask qui exécutera des requêtes vers la base de données. Le "Frontend" du site sera codé en HTML avec des extensions Bootstrap pour le style et l'esthétique.

Modèle entité relation du système

Le modèle entité-relation de la base de données de notre logiciel est la suivante, nous pouvons y apercevoir les 6 tables dont 3 avec 6 attributs ou plus qui feront fonctionner le site:

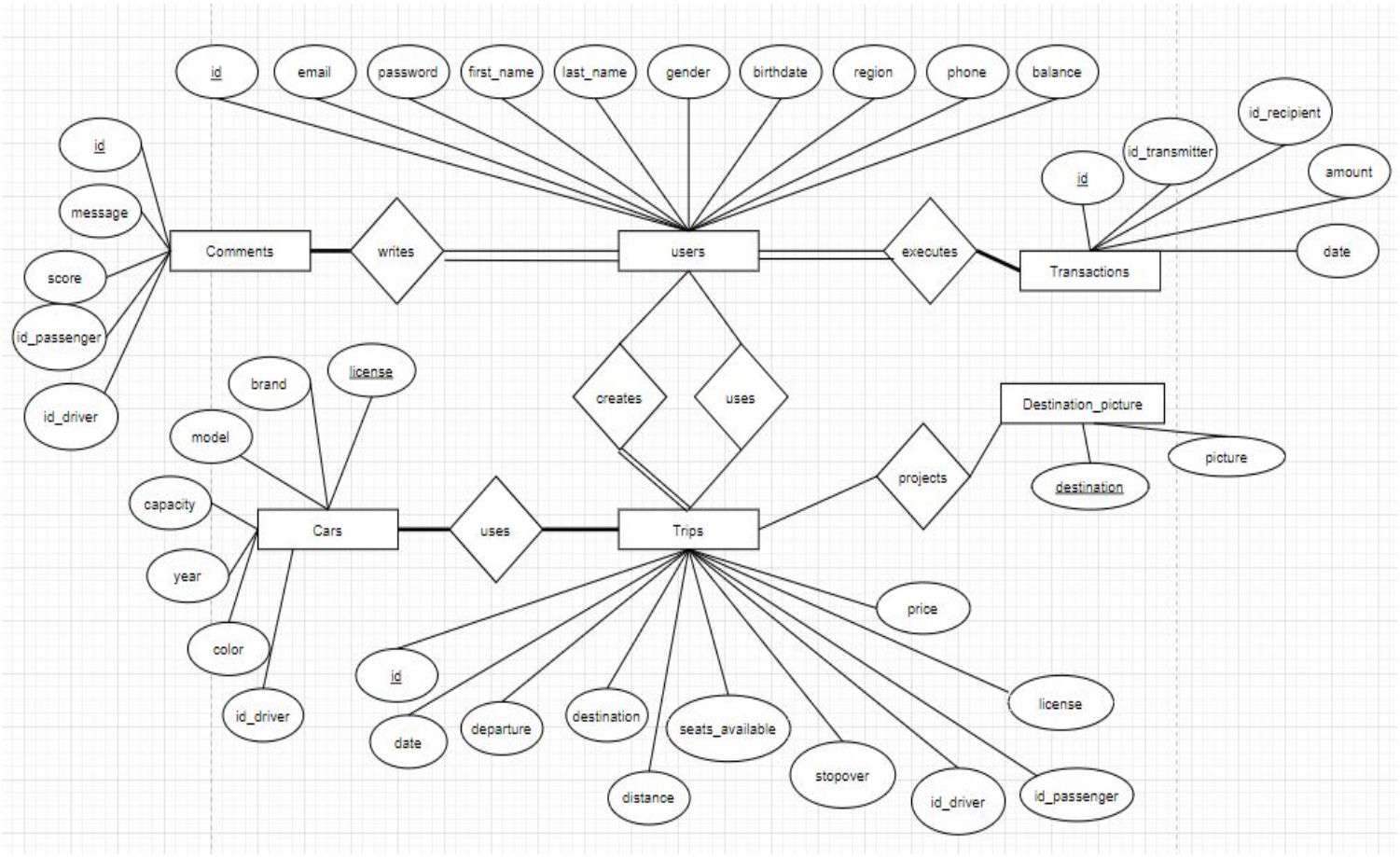
Les entités les plus importantes seront "users" et "trips" puisqu'ils interagissent beaucoup entre eux ainsi qu'avec les autres entités.
Les attributs de la table "users" sont un identifiant, qui est la clé primaire, son adresse courriel, son mot de passe, son prénom, son nom de famille, son sexe, sa date de naissance, la région d'où il vient, son numéro de téléphone et le solde dans son compte.
Un "user" peut exécuter plusieurs transactions et une transaction est faite une fois. L'entité "transaction" possède un identifiant de la transaction comme clé primaire , l'identifiant de l'émetteur, l'identifiant du récipient, le montant et la date de la transaction. Les clés étrangères sont les identifiants des utilisateurs qui font la transaction.
L'entité "user" peut aussi interagir avec l'entité "comments" en laissant des commentaires et des rétroactions. L'entité "comments" possède un identifiant comme clé primaire, un message, une note, l'id du passager (émetteur), l'id du chauffeur (destinataire) et une date. Les clés étrangères sont les identifiants de l'utilisateur qui commentent et le id du chauffeur qui reçoit les critiques.
Les "users" peuvent décider de créer ou utiliser l'entité "trips" (les voyages). Cette dernière possède un identifiant comme clé primaire, un lieu de départ, une destination, une date, une distance, qui est le nombre approximatif de kilomètres que le chauffeur conduira tenant compte des détours, le nombre de places disponibles, une entrée pour savoir si le trajet se fait avec des escales (pauses) ou d'un coup, l'identifiant du chauffeur, une chaîne de caractères

contenant les identifiants des passagers et la licence du véhicule du chauffeur. Les clés étrangères seront le id du chauffeur ainsi que la licence du véhicule.

Ce qui nous amène à notre prochaine relation: celle entre “trips” et “cars”. Un véhicule est relié à un voyage. L’entité “cars” a comme attributs sa licence comme clé primaire, une marque , un modèle, une capacité maximale, une année et une couleur et un identifiant du conducteur.

Finalement, une entité d’utilité contenant des images des destinations sert de projection à l’entité “trips” pour afficher des images représentatives des lieux de destination des voyages.

Il y a une identification qui sert de clé primaire, le nom de la destination et l’URL de l’image.



Le modèle relationnel du système

Users									
id	email	password	first_name	last_name	gender	birthdate	Région	phone	balance
1	jean@gmail.com	2c81b5b205088b281f0ac332bbe2d3cf	Jean	Tremblay	Male	1970-02-23	Québec	418-654-2020	20
2	claude@live.ca	21d6f40cfb511982e4424e0e250a9557	Claude	Vincent	Male	1980-05-20	Saguenay	418-656-1111	100
3	chris@outlook.ca	6b34fe24ac2ff8103f6fce1f0da2ef57	Chris	Boucher	Male	1965-08-12	Mauricie	418-652-2745	50
4	jeanne@live.ca	b2e82448982e0e04e929d0af411e40a2	Jeanne	Fortin	Female	1985-01-22	Drummondville	418-658-3653	60
5	anne@gmail.com	e8bf7b49eda7d9a0a8b8506104dbd6e2	Anne	Laporte	Female	1987-12-23	Rouyn	418-652-2065	156
6	joe@hotmail.com	fa4c365ab3c71531b4ff153623134ad2	Joe	Deer	Male	1999-02-04	Hamilton	418-652-2524	37

Trips									
id	date	departure	destination	distance	capacity	stopover	id_driver	id_passengers	price
1	2020-05-01 13:10	Québec	Saguenay	150	5	1	1	35,2,4	20
2	2020-06-01 09:10	Montréal	Gaspésie	250	4	0	4	1,2,3	30
3	2020-05-26 15:10	Québec	Rouyn	300	3	1	2	6	50
4	2020-07-01 12:15	Saguenay	Québec	170	6	2	35	8,36	32
5	2020-05-09 14:30	Shawinigan	Drummondville	185	2	0	67	9	14
6	2020-05-15 16:20	Rimouski	St-Tite	150	3	1	21	2,1	60
7	2020-05-04 13:00	Québec	Montréal	220	1	2	5	13	24
8	2020-05-05 08:10	Montréal	Saguenay	200	5	2	23	15,26	42

Cars					
license	brand	model	capacity	year	color
AAA AAA	Mazda	3	4	2000	Black
ABC 6G3	Toyota	Tercel	2	2003	Green
6GG 54G	Honda	Accord	3	2010	Red
123 GHT	Tesla	model x	2	2019	White
BWF T25	Toyota	Yaris	4	2013	Red
G36 8K0	Nissan	Altima	4	2006	Yellow
G25 52D	Jeep	Wrangler	2	2009	Black
H35 366	Mazda	2	4	2005	Black

Transactions				
id	id_transmitter	id_recipient	amount	date
1	4	3	29	2020-05-01 13:10
2	3	5	30	2020-05-02 13:10
3	6	34	40	2020-05-03 13:10
4	35	23	34	2020-05-04 13:10
5	23	15	22	2020-05-05 13:10
6	15	35	10	2020-05-06 13:10

comments				
id	message	score	id_passenger	id_driver
1	amazing	5	4	5
2	enjoyed it	4	23	16
3	perfect	5	12	35
4	car was stinky but ok	3	24	63

destination_pictures	
destination	url
Québec	../../static/images/quebec.jpg
Montréal	../../static/images/montreal.jpg
Saguenay	../../static/images/saguenay.jpg

Les fonctionnalités du niveau serveur et de la BD:

Création des relations

Selon les objectifs et la portée de notre site web, nous avons décidé que la plateforme, de nature simple, allait nécessiter des relations qui ne sont pas trop compliquées à manipuler. C'est pourquoi avec les fonctionnalités que notre site offre, nous nous sommes limités à six tables.

Les deux tables au centre des relations existantes dans notre base de données sont "users" et "trips". "Users" a une relation avec la table des transactions où deux utilisateurs s'échangent de l'argent lorsque l'un embarque dans le "lift" à l'autre. Cette relation peut aussi s'appliquer quand un utilisateur dépose des fonds. Cette table possède également une relation avec la table "comments" où un utilisateur peut faire une rétroaction en notant son chauffeur après un voyage. Les utilisateurs sont aussi liés avec la table des voyages où ils pourront soit créer ou être passager dans un "lift". Cette relation est la plus importante puisqu'en interagissant ainsi, la table des voyages peut mettre à jour ses entrées de "id_driver" et "id_passenger".

Pour ce qui est des relations de la table "trips", mis à part ses relations d'usage avec "users", elle est connectée avec la table "destination_pictures" afin que cette dernière store les liens des images de chaque région de destination respective et aussi pour éviter les dépendances (voir plus tard dans la section normalisation). La table "trips" possède également une relation d'utilisation avec la table "cars", où chaque voyage nécessite les informations d'une voiture en particulier pour que les passagers puissent reconnaître le véhicule du chauffeur lors du point de rendez-vous.

Requêtes et routines

Lorsqu'un utilisateur désire créer un nouveau compte sur notre site web, les informations qu'il écrit sont insérés dans la table "user". Une entrée représentant le solde est à sa disposition pour verser de l'argent fictif. Par contre, une routine (trigger) est implémentée pour cette entrée afin d'empêcher les utilisateurs de rentrer des montants négatifs.

Ensuite, ayant la possibilité de créer un trajet de covoiturage, l'utilisateur remplit un formulaire contenant les détails de son trajet et celui-ci envoie une requête d'insertion à la table "trips" dans la base de données. Une requête d'insertion est également envoyé à la table "cars" au moment où le chauffeur remplit les informations sur son véhicule. (Juste après avoir créé son trajet.) Cependant, le chauffeur a également le droit d'utiliser un des véhicules qu'il a déjà inscrit. Dans ce cas, il n'y aura pas de requêtes. Puis, lorsque le véhicule est confirmé, la table "trips" est mise à jour en y insérant la licence du véhicule.

Au contraire, un client qui cherche des voyages effectue une requête de recherche (select) dans cette table en appliquant les conditions "where" selon les filtres qu'il a remplies. Les filtres les plus utilisés sont les mots-clés pour la destination, le départ et la date. Lorsqu'un client confirme et paie un voyage, l'entrée "balance" de la table "user" du client se fait mettre à jour en enlevant le montant du voyage et est ajouté dans la même entrée de la table du chauffeur. De plus, une requête d'insertion est exécutée dans la table "transactions" en y inscrivant les informations de la transaction. Un client qui dépose des fonds dans son compte en banque déclenche aussi une insertion dans la table de transactions où l'identifiant de l'émetteur et du récipient seront le même.

Quand un utilisateur décide de laisser une rétroaction au chauffeur, une entrée est insérée dans la table "comments" avec la note, le message, et les identifiants nécessaires.

À la création d'un voyage, une requête de sélection est faite dans la table "destination_pictures" en utilisant la colonne de la destination de la table "trips" comme condition pour trouver la photo de la région de destination du voyage pour l'affichage sur le site.

Finalement, une des fonctionnalités dans la page d'accueil est d'afficher les trajets créés les plus récents. Étant donné que la page d'accueil permet d'afficher au maximum 18 trajets, la requête de recherche dans "trips" de cette page est donc limitée à 18 entrées pour optimiser la rapidité.

Indexation et optimisations

Pour la table des "users" , il n'est pas nécessaire d'ajouter un index sur l'identifiant de l'utilisateur puisque cette information est utilisée en tant que variable de session de celui qui

se connecte. Ainsi, les requêtes sur l'id des "users" lors de la création ou la recherche des voyages sont remplacées par la récupération de la variable de session.

La table "trips" de notre base de données possède plusieurs index pour permettre l'optimisation des requêtes. Tout d'abord, un index d'arbre est implémenté sur les entrées "departure", "destination", et "date" pour permettre une recherche plus rapide des voyages. En effet, comme mentionné plus haut, les filtres les plus utilisés par les utilisateurs lorsqu'ils recherchent un trajet sont ces entrées. Ensuite, la colonne "id" des voyages possède un index de hachage pour permettre des requêtes de recherches afin de localiser le bon trajet. (Par exemple: mettre à jour l'entrée de "trips" correspondant au bon identifiant pour pouvoir y insérer la licence du véhicule.) Finalement, pour une raison similaire, la colonne "id_driver" est également hachée pour pouvoir trouver le bon compte usager où transmettre l'argent et écrire des rétroactions.

D'un côté plus esthétique, la table "desination_pictures" possède un index de hachage sur la colonne "destination" pour faire des requêtes de recherche plus rapidement lorsque le système veut savoir quelle photo publier pour un nouveau trajet créé.

Pour finir, la table "comments" a un index de hachage sur le "id_driver" afin de pouvoir récupérer de façon optimale les commentaires sur le bon conducteur et les afficher dans le profil de celui-ci.

Normalisation des relations

Les tables de la base de données du système contiennent majoritairement des colonnes qui ne peuvent être normalisées en raison de leur dépendances et de leur utilisation. Par exemple, dans la table "users" tous les informations dépendent de l'identifiant de l'utilisateur. Il est donc inutile de sous-diviser la table en plusieurs autres tables. Il en est de même pour les tables "cars", "transactions" et "comments" où l'identifiant contrôle les autres clés. Au sujet de la table "trips", il y avait au début une colonne où les liens des images des destinations étaient affichés. En raison de cette dépendance vers une clé qui n'est pas primaire (transitive), une nouvelle table ("destination_pictures") est créée avec destination comme clé primaire et étrangère et le lien des images.

Implémentation de la logique d'affaire

Comme mentionné plus haut, la logique de ce logiciel est programmé en Python/Flask. Le contenu du serveur, la connexion à la base de données, les sessions et les routes sont implémentés avec Flask alors que la logique et les validations sont en Python. Pour ce qui est de la connexion avec le "FrontEnd", les routes faites en Flask offrent des redirections simples vers des pages HTML, stockées dans la couche présentation. Chaque page HTML est accompagnée d'un fichier Python contenant des fonctions utiles qui se retrouve dans la couche application. Pour plus d'informations, le code du projet est à votre disposition.

Fonctionnalités de l'interface utilisateur

L'interface de notre site web est codé en HTML avec des templates Bootstrap. Un style simpliste et épuré peut être remarqué par l'apparence de notre site.
Une barre de recherche qui est toujours présente possède les onglets les plus utiles (login, logout, profil, chercher un trajet, créer un trajet) . Une page d'accueil offre les fonctions de création et de recherche de trajets. Ces fonctionnalités sont évidemment seulement disponibles à condition d'être connecté. Les pages d'authentification sont conventionnelles.
La page de recherche de trajets affiche dynamiquement les trajets existants selon les filtres utilisés par le client. Une fois que celui-ci choisit un trajet qui lui convient, il est amené à une page de confirmation et de paiement du trajet puis, à une page d'information du trajet affichant le lieu de départ, la destination, la date, le prix, etc.

D’un autre côté, un chauffeur peut aller créer son trajet de covoiturage et est ensuite amené à une page où il peut entrer les informations sur son véhicule. En confirmant, il est redirigé à une page d’informations pareil que celle mentionnée dans le paragraphe précédent.
Les utilisateurs peuvent également utiliser la page de rétroaction pour noter leur expérience de covoiturage.

Sécurité du système

Dans notre table des utilisateurs, nous avons haché la colonne des mots de passe afin de rendre les données de compte sécuritaires. De plus, nos “cookies” et nos données de session utilisent l’id de l’utilisateur, qui est une méthode sécuritaire puisque la variable ne révèle aucune information sur l’utilisateur pendant sa connexion.

Tests du système

Pour ce qui est des tests du système et la gestion des erreurs, certaines méthodes de vérification proviennent des “input” de code HTML qui limitent le type d’entrée. Ces derniers apparaissent habituellement dans les formulaires de nos pages HTML. D’autres outils de validation proviennent de conditions programmées en Python dans le côté logique de notre application.
Il y a évidemment le système d’authentification qui est une validation clé du système. Un usager ne peut créer un compte avec un courriel existant de même qu’il doit franchir une barrière de “login”.

Ces fonctions valident les entrées avant de permettre des requêtes SQL. En somme, chaque détail tel que ce soit un ajout de fonds négatifs, la réutilisation d'une plaque de licence d'un véhicule, les dates de trajets du passé sont tous bien gérés.

Organisation et la gestion d'équipe, et division des tâches

L'équipe a pris des moyens de gestion adéquats pour la réalisation du projet. Un ou deux mois avant la remise du projet, plusieurs petites réunions se sont faits en planifiant le côté technique du projet, qui comprend le "brainstorming" pour le choix du sujet du logiciel. Ensuite, le sujet au coeur des rencontres est devenu la conception du système qui inclut entres autres la création de prototypes de diagrammes et des tables. Puis, quelques semaines avant la remise, l'architecture et les outils pour la réalisation du projet ont été choisis. À ce moment, chaque membre de l'équipe travaillait sur des aspects communs pour l'amorce du logiciel. Les implémentation de bases de données, du serveur et des routes en Flask, ainsi que les pages de "FrontEnd" ont été faits ensemble. Puis, les membres se sont séparés le travail selon les forces et préférences. Certains ont travaillés plus sur l'esthétique et l'interface utilisateur, et d'autres davantage sur la logique. Des membres se sont aussi occupés des aspects auxiliaires du projet comme le rapport, les diagrammes et l'enregistrement de la vidéo. Pour des coéquipiers qui se connaissent bien, l'équipe a avancé de façon dynamique où il n'y avait pas de conflits.

Démonstration