Cahier des charges

1. Introduction

Nous sommes une équipe de consultants SQL. Le groupe est constitué de 4 personnes : CHEVALIER Lucas, GENDROT Valentin, GUILLEMIN Nicolas, LEBEAU Kevin.

Ce document est un cahier des charges, consacré à notre projet sur la création d'une base de données. Nous avons à charge de proposer au client une structure solide et évolutive avec le temps. Il sera en adéquation avec les besoins changeants d'une startup. Le nom de cette dernière s'appelle TotallyNotAirbnb. La firme est spécialisée dans la mise en relation entre propriétaires de logement rustiques et vacanciers.

Nous évoquerons le contexte du projet, tel que la stack utilisé et la volumétrie de l'application. Ensuite, nous détaillerons les aspects techniques et fonctionnels de l'application. Puis, nous allons terminer par le diagramme des tables.

La restitution de ce projet sera à rendre avant le 31 mars 2023

2. Participant et rôles

Ce projet est réalisé par une équipe de consultants en SQL. Il y a 4 personnes : CHEVALIER Lucas, GENDROT Valentin, GUILLEMIN Nicolas, LEBEAU Kevin.

L'équipe a réparti les tâches en deux groupes. Le premier s'occupe et avance sur le cahier des charges en détaillant les éléments techniques et fonctionnels du projet. Les personnes assignées à cette tâche sont CHEVALIER Lucas, GENDROT Valentin.

Ensuite, le second groupe doit construire et réfléchir au diagramme des tables du projet. Ils doivent lier les tables entre elles et identifier les éléments de la table. Une fois le diagramme terminé, ils aideront à finaliser le cahier des charges avant la restitution.

3. Contexte du projet

3.1. Stack : mysql

Pour créer la base de données, nous avons choisi MySQL, afin de pouvoir répondre à de fortes demandes grâce à ses performances élevées en lecture.

Pour ce projet, nous utilisons la version 8.0.31 de MySQL.

3.2. Volumétrie

Lors de sa création, l'application a été conçue pour héberger jusqu'à 1000 logements et permettre l'inscription de 5000 comptes utilisateurs. Cependant, dans le futur, la capacité de l'application sera augmentée pour répondre à la croissance attendue de l'utilisateur et pour répondre aux besoins en évolution du client. Cela signifie que l'application sera en mesure d'accueillir plus de logements et de comptes utilisateurs, en fonction des besoins du marché et des demandes des utilisateurs.

4. Fonctionnalités

4.1. Besoins fonctionnels

Lorsqu'un utilisateur souhaite consulter un logement sur l'application, il pourra le faire sans avoir besoin de créer un compte. Toutefois, s'il souhaite réserver le logement, il devra créer un compte pour effectuer la transaction.

Pour ceux qui souhaitent mettre leur logement en location, une certification de compte sera requise pour éviter les fraudes. Les utilisateurs pourront soumettre des informations supplémentaires pour vérifier leur identité et la propriété du logement avant de le mettre en location. De plus, les propriétaires auront la possibilité de mettre plusieurs biens en location, ce qui leur permettra de gérer plusieurs propriétés sur la même plateforme.

Les locataires et les propriétaires pourront communiquer facilement grâce à une messagerie interne sur l'application. Cette fonctionnalité permettra aux locataires de poser des questions sur le logement et de discuter des détails de la location avec le propriétaire, ce qui facilitera le processus de location. Les propriétaires pourront également répondre rapidement aux demandes des locataires et gérer les réservations via l'application.

4.2 Besoins fonctionnels techniques

La création d'un compte se fait en utilisant la table "Utilisateur", cette création doit inclure certaines informations obligatoires. Plus précisément, ces informations comprennent le nom, le prénom, l'adresse e-mail, le mot de passe, le numéro de téléphone et la date de naissance. Il est important de noter que ces informations sont nécessaires pour plusieurs raisons.

Tout d'abord, le nom et le prénom sont utilisés pour identifier le titulaire du compte et pour personnaliser les communications avec lui. L'adresse e-mail est utilisée pour permettre au titulaire du compte de récupérer son mot de passe en cas de besoin et pour lui envoyer des informations liées à une réservation par exemple.

Le mot de passe est utilisé pour protéger l'accès au compte et pour empêcher toute personne non autorisée d'accéder aux informations du titulaire.

Le numéro de téléphone peut être utilisé pour envoyer des notifications, contacter un propriétaire et récupérer son compte en cas de perte du mot de passe. Enfin, la date de naissance est utilisée pour vérifier si l'utilisateur est majeur.

La réservation d'un logement sur l'application sera liée aux tables logement et utilisateur. Chaque réservation contiendra des informations importantes sur l'utilisateur et le logement.

Les informations de l'utilisateur incluses dans une réservation peuvent inclure son nom, prénom, adresse email et numéro de téléphone. Ces informations aideront à identifier l'utilisateur et à le recontacter en cas de besoin.

Les informations relatives au logement comprennent le titre du logement, l'adresse, la ville et le code postal. Le prix de la nuitée sera également inclus dans la réservation pour permettre aux utilisateurs de connaître le coût total de la location avant de finaliser leur réservation.

Lorsqu'il s'agit de la gestion des échanges entre le propriétaire et le client, les tables impliqués seront utilisateur, réservation et message. Tout d'abord, il y a la table avec les informations du client (utilisateur), contenant des détails tels que le nom et prénom. La même table est utilisée pour récupérer les informations du propriétaire puisque pour rappel un propriétaire est un utilisateur. La table réservation sert à récupérer l'ID de cette dernière ce qui permettra l'ouverture des échanges.

La mise en location est possible grâce à une annonce mentionnant un titre, l'adresse du logement, le prix par nuitée, une description, le nombre de pièces de celui-ci ainsi que le nombre de personnes pouvant y séjourner simultanément, l'annonce sera modifiable et la date de création ainsi que la date de modification seront enregistrées.

Après leur séjour, les utilisateurs auront la possibilité de laisser un avis sur le logement, impliquant les tables utilisateur et logement. La table utilisateur permettra de récupérer les noms et prénoms des utilisateurs, tandis que la table logement sera utilisée pour identifier le logement en question via son ID. Les utilisateurs pourront noter et commenter leur expérience.

4.3. Description techniques

4.3.1. Champs/tables

nous aurons donc un total de 11 tables :

1. Logement :
    - id : int(11)
    - titre : varchar(150)
    - adresse : varchar(255)
    - ville : varchar(100)
    - codePostal : int(5)
    - prix : decimal(5,2)

- description : text(3000)
- dateDeModification : timestamp(6) current_timestamp(6)
- dateDeCreation : timestamp(6)
- dateDeSuppression : timestamp(6) current_timestamp(6)
- nombreDePieces : int(2)
- nombreDePlaces : int(3)

2. Utilisateur :
    - id : int (11)
    - nom : varchar(100)
    - prenom : varchar(100)
    - mail : varchar(300)
    - motDePasse : varchar(25)
    - telephone : varchar(30)
    - dateDeNaissance : date
    - dateDeCreation : timestamp(6)
    - compteCertif : tinyint

3. Réservation :
    - id : int (11)
    - LogementId : int(11)
    - UtilisateurId : int(11)
    - dateReservation : date

4. typeLogement :
    - id : int (11)
    - nom : varchar(45)
    - LogementId : int(11)
5. Disponibilité :
    - id : int (11)
    - dateDispoDebut : date
    - dateDispoFin : date
    - LogementId : int(11)

6. Services :
    - id : int (11)
    - nom : varchar(45)
    - Logement-id : int(11)

7. Message :
    - id : int (11)
    - contenu : varchar(2000)
    - dateEnvoie : timestamp(6) current_timestamp(6)
    - dateModif : timestamp(6)

8. Photo :
    - id : int (11)
    - url : varchar(1000)
    - logementId : int(11)

9. Avis :
    - id : int (11)
    - note : int(1)
    - commentaire : text(3000)
    - LogementId : int(11)
    - UtilisateurId : int(11)

10. Echange :
    - id : int (11)
    - dateDeCreation : timestamp(6) current_timestamp(6)
    - UtilisateurId : int(11)
    - ReservationId : int(11)
    - MessageId : int(11)

11. Facture :
    - id : int (11)
    - dateDeDelivrance :timestamp(6) current_timestamp(6)
    - dateEcheance : date

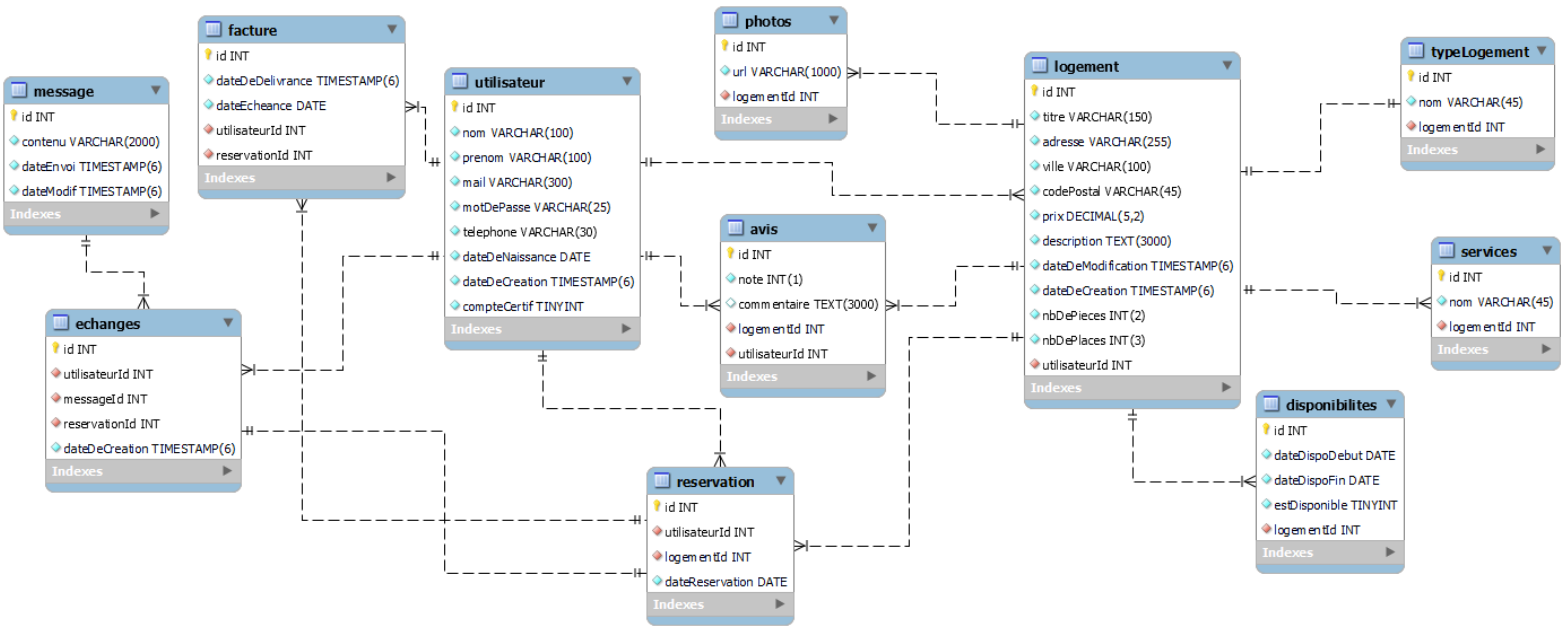
- utilisateurId : int(11)
- reservationId : int(11)

4.3.2. Contraintes

contrainte du projet du client ?

- structure solide et scalable avec le temps
- besoins changeants du fondateur

5. Diagramme des tables



6. Conclusion