

## ▀ ▀ ▀ Modèle de copie :

**Créer et administrer une base de données**

**GDWFSCAUBDDEXAIII1A**

**Ceci est un modèle de copie. N’oubliez pas de renseigner vos prénom/nom, ainsi que le nom et le lien vers le projet.**

**Vous pouvez bien sûr agrandir les cadres pour répondre aux questions sur la description du projet si nécessaire.**

**Prénom :**Mickaël

**Nom : Desclaux-Arramond**

**ATTENTION ! PENSEZ À RENSEIGNER VOS NOM ET PRÉNOM DANS LE TITRE DE VOS FICHIERS / PROJETS !**

Nom du projet : ……….

Lien Github du projet : ……………

Lien Drive du projet (si nécessaire) : ………

URL du site (si vous avez mis votre projet en ligne) : ……….

**Description du projet**

1. Décrivez les tâches ou opérations que vous avez effectuées, et dans quelles conditions. Dans cette rubrique, le jury cherche à voir comment vous procédez : comment vous organisez votre travail, comment vous réalisez concrètement la tâche ou l’opération pas à pas.

Utiliser un langage professionnel. Employez le « je », car vous parlez en votre nom. Vous pouvez écrire au temps présent.

Je commence par créer le diagramme de classe afin de déterminer les tables que je vais devoir créer, ainsi que toutes les colonnes nécessaires à chaque table et leur type de données. Une fois mon diagramme terminé, je décide de créer 9 tables au total :

-Table User : Stocke les informations de connexion des utilisateurs.

-Table Rôle : Stocke les différents rôles existants sur le site.

-Table Cinema : Stocke les informations des différents cinémas du groupe avec un nom et une adresse.

-Table Room : Stocke les informations des différentes salles de cinéma ainsi que leurs nombres de places maximum. Chaque salle est associée à un cinéma.

-Table Movie : Stocke les informations des films qui sont diffusés par les cinémas.

-Table Movie\_genre : Stocke les différents genres de films (comédie, action etc..) afin de stocker uniquement les ID dans la table Movie et éviter les répétions de texte dans la table Movie.

-Table Ticket : Stocke les tickets réservés par les clients du cinéma et le type de ticket acheté.

-Table Ticket\_category : Stocke les différentes catégories de tickets et leurs prix, afin de stocker uniquement un ID du type de ticket dans la table Ticket.

-Table Session : Permet de stocker les séances de diffusions d'un film, en spécifiant la salle de cinéma, le film projeté et la date et l'heure de la séance.

Je crée la base de données "cinema\_studi" dans pgadmin, puis je me connecte à la base sur DataGrip et je crée le schéma "cinema".

Je crée ensuite mes tables avec la méthode CREATE TABLE :

Une image contenant texte, capture d’écran, Police, nombre

Description générée automatiquement

Avant chaque création de table, j'insère une ligne de code pour supprimer la table (DROP TABLE IF EXISTS), cela me permet de les recréer plus facilement si je dois faire des modifications. Je répète cette méthode pour la création de toutes mes tables.

Je passe ensuite à la création des clés secondaires afin d'associer mes tables. Par exemple, pour associer la table cinéma à la table room, afin que chaque salle appartienne à un cinéma, j'écris le code suivant :

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

J'ajoute de la même manière les autres associations entre les tables :

Une image contenant texte, capture d’écran, menu, Police

Description générée automatiquement

Ensuite, j'insère des données dans chaque table avec le code INSERT INTO … VALUES.

Une image contenant texte, capture d’écran, Police, ligne

Description générée automatiquement

Enfin, je crée trois fonctions SQL pour la réservation des tickets, l'ajout d'un film et l'ajout d'une session de cinéma.

Je crée ces 3 fonctions en langage plpgsql, qui est un langage de procédures SQL.

Je crée la fonction buy\_ticket(), qui permet aux clients de réserver un ou plusieurs tickets pour une session.

Pour cela, je définis les paramètres qui vont devoir être passés à la fonction : l'ID de la session, la catégorie du ticket et la quantité de tickets réservés. Je déclare ensuite deux variables, v\_max\_places et v\_sold\_tickets qui vont me permettre de vérifier que le nombre de tickets sur le point d'être réservés ne dépassent pas le nombre maximum de places dans la salle.

Je récupère le nombre de sièges dans la salle, puis je compte le nombre de tickets déjà réservés pour cette session. Si le nombre de tickets sur le point d'être réservés ne dépassent pas le maximum, alors un ticket est créé dans la table "Ticket" pour chaque ticket réservé. Si la séance est déjà complète, ou que la réservation dépasserait le nombre maximum de places, alors un message d'erreur s'affiche à l'écran : "Désolé, la session est complète".

Afin de tester cette fonction, je fais un "SELECT cinema.buy\_ticket()" où je réserve deux places pour une session. Une fois cette fonction validée, je vérifie le comportement de la fonction au cas où je réserve plus de places que le maximum de la salle :

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

Je passe ensuite sur la création de la fonction permettant aux utilisateurs d'ajouter des films.

Je crée une fonction add\_movie(), qui passe en paramètre un nom, une durée, un genre et une description.

La fonction récupère ces paramètres et les insèrent dans la table "Movie".

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

Enfin, je crée la fonction add\_session qui permettra aux utilisateurs d'ajouter la programmation de la diffusion des films.

Cette fonction demande en paramètre le cinéma concerné, sa salle, le film qui va être diffusé et le jour et l'heure de la projection.

Avant d'insérer les données, la fonction doit vérifier si l'utilisateur qui exécute la fonction est bien le gérant du cinéma (le cinema\_ID renseigné à la ligne de cet utilisateur doit être égal à la valeur du cinéma passée en paramètre) ou s'il est un admin.

Je déclare pour cela deux variables, v\_user\_id et v\_admin : l'identifiant de l'utilisateur qui exécute la fonction est stocké dans la variable v\_user\_id, et la vérification du rôle de l'utilisateur modifie la valeur de v\_admin (true si l'utilisateur est un admin, false dans le cas contraire).

Si l'utilisateur est un admin ou si la fonction a identifié qu'il s'agit bien du gérant du cinéma (cinema\_id de l'utilisateur = cinema\_id passé en paramètre de la fonction), alors les données sont ajoutées dans la table. Si ces conditions ne sont pas validées, alors la fonction lève une erreur.

1. Précisez les moyens utilisés. Expliquez tout ce dont vous avez eu besoin pour réaliser vos tâches : langages de programmation, frameworks, outils, logiciels, documentations techniques, etc...

J'ai codé ma base de données en SQL sur l'IDE DataGrip. J'ai aussi utilisé le logiciel pgAdmin 4 pour visualiser ma base postgreSQL. Je me suis servi de la documentation postgres. J'ai utilisé le langage PL/pgSQL pour créer mes fonctions SQL.

1. Contexte. Les noms des organismes, entreprises ou associations, dans lesquels vous avez exercé vos pratiques

NB: Pour le cas des exercices et évaluations demandées sur la plateforme Studi, il s'agit de...Studi.

Studi

1. Informations complémentaires (*facultatif*)