**RAPPORT TRE TRES TRES TRES DESOLEE FAMILLE**

**PRATIQUE DES REGLES DE GESTION**

1. **Règles des gestions utilisateurs :**
2. **Inscription et connexion :**
3. **Diagramme de Cas d'Utilisation**

**Acteurs :**

* Utilisateur

Cas d'utilisation :

* S'inscrire
* Se connecter
* Se déconnecter

Relations :

* L'utilisateur peut s'inscrire.
* L'utilisateur peut se connecter (avec une vérification des règles de mot de passe et de connexion unique).
* L'utilisateur peut se déconnecter.

[Utilisateur] --> (S'inscrire)

[Utilisateur] --> (Se connecter)

[Utilisateur] --> (Se déconnecter)

(S'inscrire) <|-- (Se connecter)

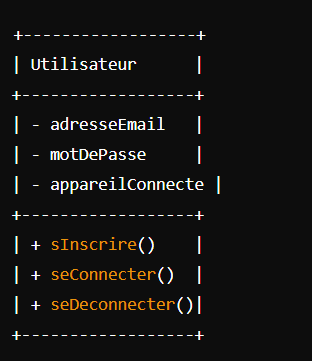
B. **Diagramme de Classes**

**Classes :**

* Utilisateur
  + Attributs : adresseEmail, motDePasse, appareilConnecte
  + Méthodes : sInscrire(), seConnecter(), seDeconnecter()

**Relations :**

* Utilisateur peut avoir un ou plusieurs appareils associés, mais ne peut être connecté qu'à un seul appareil à la fois.



**C. Diagramme de Séquence**

Scénario : Connexion d'un utilisateur

1. Utilisateur entre ses identifiants (adresse email et mot de passe).
2. Système valide l'adresse e-mail.
3. Système vérifie le mot de passe (force du mot de passe, unicité de la connexion).
4. Système connecte l'utilisateur.
5. Système enregistre l'appareil utilisé pour la connexion.

Utilisateur -> Système : entrerIdentifiants()

Système -> Système : validerAdresseEmail()

Système -> Système : verifierMotDePasse()

Système -> Utilisateur : connexionRéussie()

Système -> Système : enregistrerAppareil()

D. **Diagramme d'Activités**

**Activité : Inscription d'un utilisateur**

1. Commencer
2. Saisir adresse e-mail
3. Vérifier format de l'adresse e-mail
4. Saisir mot de passe
5. Vérifier la force du mot de passe
6. Enregistrer utilisateur dans le système
7. Terminer

[Commencer]

|

[Saisir adresse e-mail]

|

[Vérifier format e-mail]

|

[Saisir mot de passe]

|

[Vérifier force mot de passe]

|

[Enregistrer utilisateur]

|

[Terminer]

E. **Diagramme d'États**

**État : Connexion d'un utilisateur**

* **État Initial** : Non connecté
* **État** : Tentative de connexion
* **État** : Connexion réussie
* **État** : Connexion échouée
* **État Final** : Connecté

[Non connecté] --> [Tentative de connexion]

[Tentative de connexion] --> [Connexion réussie]

[Tentative de connexion] --> [Connexion échouée]

[Connexion réussie] --> [Connecté]

[Connexion échouée] --> [Non connecté]

F. **Diagramme de Communication**

**Scénario : Inscription**

* **Utilisateur** envoie une demande d'inscription au **Système**.
* **Système** vérifie les détails et envoie une confirmation à l'**Utilisateur**.

Utilisateur -> Système : demandeInscription()

Système -> Utilisateur : confirmationInscription()

G. **Diagramme de Composants**

**Composants :**

* Interface Utilisateur (pour les inscriptions, connexions)
* Service d'Authentification (valide les identifiants, vérifie la force du mot de passe)
* Base de Données (stocke les informations des utilisateurs)

**Relations :**

* Interface Utilisateur communique avec le Service d'Authentification.
* Service d'Authentification accède à la Base de Données pour vérifier et enregistrer les informations.

+-------------------+

| Interface Utilisateur |

+-------------------+

|

v

+-------------------+

| Service d'Authentification |

+-------------------+

|

v

+-------------------+

| Base de Données |

+-------------------+

H. **Diagramme de Déploiement**

**Nœuds :**

* Serveur Web (héberge l'application et le service d'authentification)
* Base de Données (stocke les informations des utilisateurs)

**Relations :**

* Serveur Web se connecte à la Base de Données.

+-------------------+

| Serveur Web |

+-------------------+

|

v

+-------------------+

| Base de Données |

+-------------------+

1. **Rôle et Permission :**

**A. Diagramme de Cas d'Utilisation**

Acteurs :

* Manager
* Client
* Employé
* Livreur
* Visiteur

Cas d'utilisation :

* Gérer les utilisateurs (Manager)
* Gérer les articles (Manager)
* Passer des commandes (Client)
* Consulter le menu (Client, Visiteur)
* Préparer les commandes (Employé)
* Mettre à jour le statut des commandes (Employé, Livreur)
* Consulter les commandes à livrer (Livreur)

Relations :

**[Manager] --> (Gérer les utilisateurs)**

**[Manager] --> (Gérer les articles)**

**[Client] --> (Passer des commandes)**

**[Client] --> (Consulter le menu)**

**[Employé] --> (Préparer les commandes)**

**[Employé] --> (Mettre à jour le statut des commandes)**

**[Livreur] --> (Consulter les commandes à livrer)**

**[Livreur] --> (Mettre à jour le statut des commandes)**

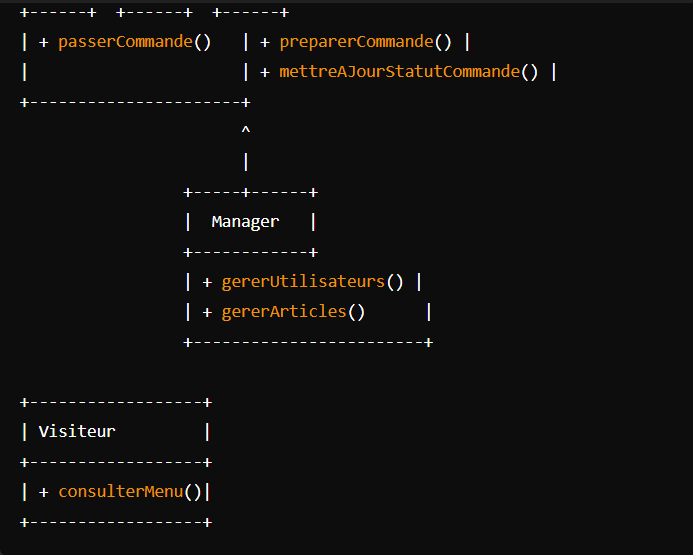
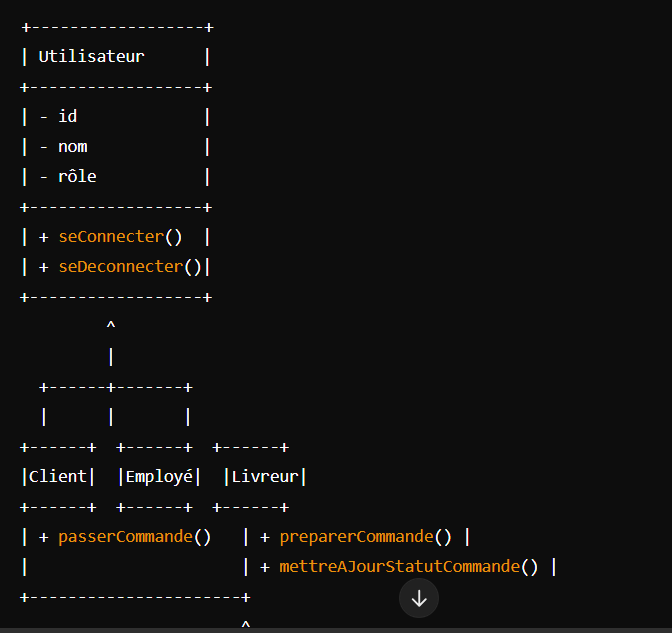
**[Visiteur] --> (Consulter le menu)**

**B. Diagramme de Classes**

**Classes :**

* Utilisateur
  + Attributs : id, nom, rôle
  + Méthodes : seConnecter(), seDeconnecter()
* Client (hérite de Utilisateur)
  + Méthodes : passerCommande()
* Employé (hérite de Utilisateur)
  + Méthodes : preparerCommande(), mettreAJourStatutCommande()
* Livreur (hérite de Utilisateur)
  + Méthodes : consulterCommandesALivrer(), mettreAJourStatutLivraison()
* Manager (hérite de Utilisateur)
  + Méthodes : gererUtilisateurs(), gererArticles()
* Visiteur (hérite de Utilisateur)
  + Méthodes : consulterMenu()

**Diagramme de Classes :**

****

**C. Diagramme de Séquence**

Scénario : Passage d'une commande par un client

1. Client sélectionne les articles.
2. Client passe la commande.
3. Système enregistre la commande.
4. Système notifie l'Employé pour la préparation.

Client -> Système : sélectionnerArticles()

Client -> Système : passerCommande()

Système -> Système : enregistrerCommande()

Système -> Employé : notifierCommande()

**D. Diagramme d'Activités**

Activité : Préparation de commande

1. Début
2. Recevoir la commande
3. Préparer la commande
4. Mettre à jour le statut
5. Terminer

[Début]

|

[Recevoir la commande]

|

[Préparer la commande]

|

[Mettre à jour le statut]

|

[Terminer]

E. **Diagramme d'États**

**État : Gestion des commandes**

* **État Initial** : Nouvelle commande
* **État** : En préparation
* **État** : Prête à être livrée
* **État** : Livrée
* **État Final** : Commande terminée

[Nouvelle commande] --> [En préparation]

[En préparation] --> [Prête à être livrée]

[Prête à être livrée] --> [Livrée]

[Livrée] --> [Commande terminée]

F. **Diagramme de Communication**

**Scénario : Mise à jour du statut de la commande**

* **Employé** met à jour le statut de la commande.
* **Système** enregistre la mise à jour et informe le **Livreur** si nécessaire.

Employé -> Système : mettreAJourStatutCommande()

Système -> Système : enregistrerMiseAJour()

Système -> Livreur : notifierMiseAJour()

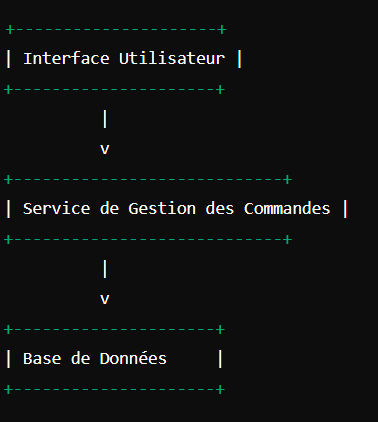
**H. Diagramme de Composants**

**Composants :**

* Interface Utilisateur (pour chaque rôle : Manager, Client, Employé, Livreur, Visiteur)
* Service de Gestion des Commandes (traitement des commandes, mise à jour des statuts)
* Base de Données (stocke les informations sur les utilisateurs, les commandes, les articles)

Relations :

* Interface Utilisateur communique avec le Service de Gestion des Commandes.
* Service de Gestion des Commandes accède à la Base de Données pour les opérations.



H. **Diagramme de Déploiement**

**Nœuds :**

* Serveur d'Application (héberge le Service de Gestion des Commandes et l'Interface Utilisateur)
* Base de Données (stocke les données)

**Relations :**

* Le Serveur d'Application se connecte à la Base de Données pour accéder aux informations nécessaires.

1. **Profil Utilisateur :**

**A. Diagramme de Cas d'Utilisation**

**Acteurs :**

* Utilisateur
* Manager

Cas d'utilisation :

* Consulter le profil
* Mettre à jour les informations personnelles (nom, adresse, numéro de téléphone, etc.)
* Modifier le rôle (réservé au Manager)

**Relations :**

**[Utilisateur] --> (Consulter le profil)**

**[Utilisateur] --> (Mettre à jour les informations personnelles)**

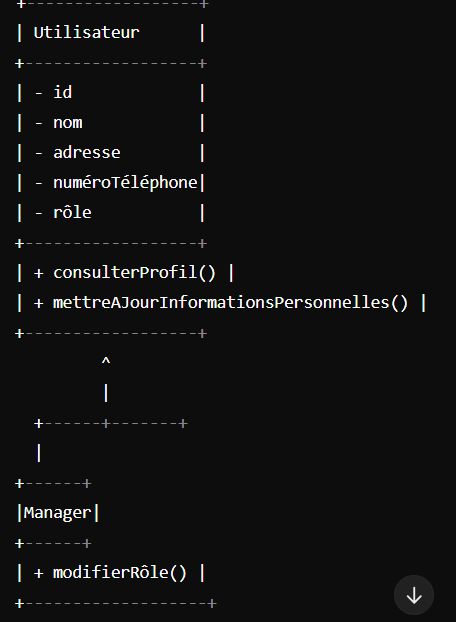
**[Manager] --> (Modifier le rôle)**

**B. Diagramme de Classes**

Classes :

* Utilisateur
  + Attributs : id, nom, adresse, numéroTéléphone, rôle
  + Méthodes : consulterProfil(), mettreAJourInformationsPersonnelles()
* Manager (hérite de Utilisateur)
  + Méthodes : modifierRôle()

**Diagramme de Classes** :



**C. Diagramme de Séquence**

**Scénario : Mise à jour des informations personnelles**

1. Utilisateur demande à mettre à jour ses informations.
2. Système affiche le formulaire de mise à jour.
3. Utilisateur soumet les nouvelles informations.
4. Système met à jour les informations dans la base de données.

Utilisateur -> Système : demanderMiseAJourInformations()

Système -> Utilisateur : afficherFormulaireMiseAJour()

Utilisateur -> Système : soumettreNouvellesInformations()

Système -> Base de Données : mettreAJourInformations()

Système -> Utilisateur : confirmationMiseAJour()

**D. Diagramme d'Activités**

**Activité : Mise à jour des informations personnelles**

1. Début
2. Accéder au profil
3. Remplir les nouvelles informations
4. Soumettre les informations
5. Valider la mise à jour
6. Terminer

[Début]

|

[Accéder au profil]

|

[Remplir les nouvelles informations]

|

[Soumettre les informations]

|

[Valider la mise à jour]

|

[Terminer]

**E. Diagramme d'États**

**État : Profil utilisateur**

* État Initial : Informations non modifiées
* État : En cours de modification
* État : Informations mises à jour
* État Final : Informations validées

**[Informations non modifiées] --> [En cours de modification]**

**[En cours de modification] --> [Informations mises à jour]**

**[Informations mises à jour] --> [Informations validées]**

**F. Diagramme de Communication**

**Scénario : Consultation et mise à jour du profil**

* Utilisateur consulte son profil.
* Système renvoie les informations du profil.
* Utilisateur met à jour ses informations.
* Système met à jour les informations dans la base de données.

**Utilisateur -> Système : consulterProfil()**

**Système -> Utilisateur : afficherInformationsProfil()**

**Utilisateur -> Système : mettreAJourInformationsPersonnelles()**

**Système -> Base de Données : mettreAJourInformations()**

**Système -> Utilisateur : confirmationMiseAJour()**

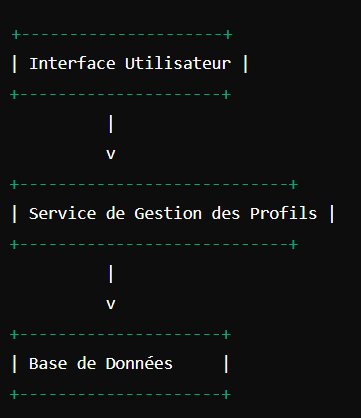
**G. Diagramme de Composants**

**Composants :**

* Interface Utilisateur (permet aux utilisateurs de consulter et mettre à jour leur profil)
* Service de Gestion des Profils (gère les informations des utilisateurs)
* Base de Données (stocke les informations personnelles des utilisateurs)

Relations :

* Interface Utilisateur communique avec le Service de Gestion des Profils.
* Service de Gestion des Profils accède à la Base de Données pour mettre à jour et récupérer les informations.



**H. Diagramme de Déploiement**

**Nœuds :**

* Serveur d'Application (héberge le Service de Gestion des Profils et l'Interface Utilisateur)
* Base de Données (stocke les informations des utilisateurs)

Relations :

* Le Serveur d'Application se connecte à la Base de Données pour accéder et mettre à jour les informations des utilisateurs.

**+---------------------+**

**| Serveur d'Application |**

**+---------------------+**

**|**

**v**

**+---------------------+**

**| Base de Données |**

**+---------------------+**

1. **Gestions commandes :**

**Création des commandes, Modification des commandes  et des**

**Statut des commandes :**

* 1. **Diagramme de Cas d'Utilisation**

Acteurs :

* Client
* Manager
* Employé
* Livreur

Cas d'utilisation :

* Créer une commande
* Modifier une commande
* Mettre à jour le statut des commandes

Relations :

**[Client] --> (Créer une commande)**

**[Client] --> (Modifier une commande)**

**[Manager] --> (Modifier une commande)**

**[Employé] --> (Modifier une commande)**

**[Employé] --> (Mettre à jour le statut des commandes)**

**[Livreur] --> (Mettre à jour le statut des commandes)**

* 1. **Diagramme de Classes**

**Classes :**

* **Commande**
  + **Attributs : id, dateHeureCréation, statut, listeArticles**
  + **Méthodes : ajouterArticle(), modifierCommande(), annulerCommande(), mettreAJourStatut()**
* **Article**
  + **Attributs : id, nom, prix**
* **Utilisateur**
  + **Attributs : id, nom, rôle**
* **Client (hérite de Utilisateur)**
  + **Méthodes : créerCommande(), modifierCommande()**
* **Manager (hérite de Utilisateur)**
  + **Méthodes : modifierCommande()**
* **Employé (hérite de Utilisateur)**
  + **Méthodes : mettreAJourStatutCommande()**
* **Livreur (hérite de Utilisateur)**
  + **Méthodes : mettreAJourStatutCommande()**
  1. **Diagramme de Séquences :**

**Scénario : Création et modification de commande**

1. Client sélectionne les articles et crée une commande.
2. Système enregistre la commande.
3. Client demande à modifier la commande.
4. Système vérifie si la commande peut être modifiée et applique les modifications.
5. Manager/Employé (si nécessaire) ou le Client peut également modifier la commande.

**Client -> Système : créerCommande()**

**Système -> Base de Données : enregistrerCommande()**

**Client -> Système : modifierCommande()**

**Système -> Système : vérifierModificationPossible()**

**Système -> Base de Données : mettreAJourCommande()**

* 1. **Diagramme d’activité :**

**Activité : Gestion des commandes**

1. Début
2. Créer une commande
3. Vérifier la possibilité de modification
4. Modifier la commande (si possible)
5. Mettre à jour le statut de la commande (si applicable)
6. Annuler la commande (si dans les 10 minutes)
7. Terminer

[Début]

|

[Créer une commande]

|

[Afficher commande créée]

|

[Vérifier possibilité de modification]

|

[Modifier la commande (si possible)]

|

[Mettre à jour le statut de la commande]

|

[Annuler la commande (si dans les 10 minutes)]

|

[Terminer]

* 1. **Diagramme d’Etat :**

État : Cycle de vie d'une commande

* État Initial : Commande créée
* État : En préparation
* État : Prête
* État : En livraison
* État : Livrée
* État Final : Annulée

[Commande créée] --> [En préparation]

[En préparation] --> [Prête]

[Prête] --> [En livraison]

[En livraison] --> [Livrée]

[Commande créée] --> [Annulée] (si dans les 10 minutes)

* 1. Diagramme de communication :

**Scénario : Mise à jour du statut de la commande**

* **Employé** ou **Livreur** met à jour le statut de la commande.
* **Système** enregistre la mise à jour et notifie le client si nécessaire.

Employé -> Système : mettreAJourStatutCommande()

Système -> Base de Données : mettreAJourStatut()

Système -> Client : notifierChangementStatut()

* 1. Diagramme de composants :

**Composants :**

* Interface Utilisateur (pour la gestion des commandes : création, modification, mise à jour du statut)
* Service de Gestion des Commandes (gère la logique des commandes : création, modification, mise à jour du statut)
* Base de Données (stocke les informations sur les commandes et les articles)

**Relations :**

* Interface Utilisateur communique avec le Service de Gestion des Commandes.
* Service de Gestion des Commandes accède à la Base de Données pour les opérations.
  1. Diagramme de déploiement :

**Nœuds :**

* Serveur d'Application (héberge le Service de Gestion des Commandes et l'Interface Utilisateur)
* Base de Données (stocke les informations sur les commandes et les articles)

**Relations :**

* Le Serveur d'Application se connecte à la Base de Données pour accéder et mettre à jour les informations des commandes.

+---------------------+

| Serveur d'Application |

+---------------------+

|

v

+---------------------+

| Base de Données |

+---------------------+

1. **Gestion articles :**
2. **Gestion des menus :**

**1. Diagramme de Cas d'Utilisation**

Acteurs :

* Manager

Cas d'utilisation :

* Ajouter un article au menu
* Modifier un article du menu
* Supprimer un article du menu

**Relations :**

**[Manager] --> (Ajouter un article au menu)**

**[Manager] --> (Modifier un article du menu)**

**[Manager] --> (Supprimer un article du menu)**

**2. Diagramme de Classes**

**Classes :**

* Article
  + Attributs : id, nom, description, prix, disponibilité
  + Méthodes : ajouterArticle(), modifierArticle(), supprimerArticle()
* Manager (hérite de Utilisateur)
  + Méthodes : ajouterArticle(), modifierArticle(), supprimerArticle()

**Diagramme de Classes :**

**+------------------+**

**| Article |**

**+------------------+**

**| - id |**

**| - nom |**

**| - description |**

**| - prix |**

**| - disponibilité |**

**+------------------+**

**| + ajouterArticle() |**

**| + modifierArticle() |**

**| + supprimerArticle() |**

**+------------------+**

**+------------------+**

**| Manager |**

**+------------------+**

**| + ajouterArticle() |**

**| + modifierArticle() |**

**| + supprimerArticle() |**

**+------------------+**

**3. Diagramme de Séquence**

**Scénario : Gestion des articles**

1. Manager ajoute un nouvel article.
2. Système enregistre l'article dans la base de données.
3. Manager modifie un article existant.
4. Système met à jour les détails de l'article dans la base de données.
5. Manager supprime un article.
6. Système vérifie si l'article est dans une commande en cours avant de le supprimer, puis supprime l'article.

**Manager -> Système : ajouterArticle()**

**Système -> Base de Données : enregistrerArticle()**

**Manager -> Système : modifierArticle()**

**Système -> Base de Données : mettreAJourArticle()**

**Manager -> Système : supprimerArticle()**

**Système -> Base de Données : vérifierArticleDansCommande()**

**Système -> Base de Données : supprimerArticle()**

**4. Diagramme d'Activités**

Activité : Gestion des articles

1. Début
2. Ajouter un article
3. Modifier un article
4. Supprimer un article
   * Vérifier si l'article est dans une commande en cours
5. Terminer

**[Début]**

**|**

**[Ajouter un article]**

**|**

**[Modifier un article]**

**|**

**[Supprimer un article]**

**|**

**[Vérifier si l'article est dans une commande en cours]**

**|**

**[Terminer]**

**5. Diagramme d'États**

État : Gestion des articles

* État Initial : Article non ajouté
* État : Article ajouté
* État : Article modifié
* État : Article prêt à être supprimé
* État Final : Article supprimé

**[Article non ajouté] --> [Article ajouté]**

**[Article ajouté] --> [Article modifié]**

**[Article modifié] --> [Article prêt à être supprimé]**

**[Article prêt à être supprimé] --> [Article supprimé]**

**6. Diagramme de Communication**

**Scénario : Gestion des articles**

* Manager demande à ajouter/modifier/supprimer un article.
* Système exécute les actions nécessaires et interagit avec la Base de Données pour la persistance des données.

**Manager -> Système : ajouter/modifier/supprimerArticle()**

**Système -> Base de Données : enregistrer/mettreAJour/supprimerArticle()**

**7. Diagramme de Composants**

Composants :

* Interface Utilisateur (pour la gestion des articles : ajout, modification, suppression)
* Service de Gestion des Articles (gère la logique des articles)
* Base de Données (stocke les informations des articles)

Relations :

* Interface Utilisateur communique avec le Service de Gestion des Articles.
* Service de Gestion des Articles accède à la Base de Données pour les opérations.

**+---------------------+**

**| Interface Utilisateur |**

**+---------------------+**

**|**

**v**

**+-----------------------------+**

**| Service de Gestion des Articles |**

**+-----------------------------+**

**|**

**v**

**+---------------------+**

**| Base de Données |**

**+---------------------+**

**8. Diagramme de Déploiement**

**Nœuds :**

* Serveur d'Application (héberge le Service de Gestion des Articles et l'Interface Utilisateur)
* Base de Données (stocke les informations sur les articles)

Relations :

* Le Serveur d'Application se connecte à la Base de Données pour accéder et mettre à jour les informations des articles.

**+---------------------+**

**| Serveur d'Application |**

**+---------------------+**

**|**

**v**

**+---------------------+**

**| Base de Données |**

**+---------------------+**

1. **Stock de articles :**

**1. Diagramme de Cas d'Utilisation**

Acteurs :

* Employé

Cas d'utilisation :

* Mettre à jour le stock des articles
* Marquer un article comme indisponible

Relations :

**[Employé] --> (Mettre à jour le stock des articles)**

**[Employé] --> (Marquer un article comme indisponible)**

**2. Diagramme de Classes**

**Classes :**

* Article
  + Attributs : id, nom, description, prix, disponibilité, stock
  + Méthodes : mettreAJourStock(), marquerIndisponible()
* Employé (hérite de Utilisateur)
  + Méthodes : mettreAJourStock()

**Diagramme de Classes :**

**diff**

**Copier le code**

**+------------------+**

**| Article |**

**+------------------+**

**| - id |**

**| - nom |**

**| - description |**

**| - prix |**

**| - disponibilité |**

**| - stock |**

**+------------------+**

**| + mettreAJourStock() |**

**| + marquerIndisponible() |**

**+------------------+**

**+------------------+**

**| Employé |**

**+------------------+**

**| + mettreAJourStock() |**

**+------------------+**

**3. Diagramme de Séquence**

Scénario : Mise à jour du stock et gestion de la disponibilité

1. Employé met à jour le stock d'un article.
2. Système vérifie le niveau de stock de l'article.
3. Système marque l'article comme indisponible si le stock est à zéro.
4. Système met à jour la disponibilité de l'article dans le menu.

**Employé -> Système : mettreAJourStock()**

**Système -> Système : vérifierStock()**

**Système -> Système : marquerIndisponible() (si stock = 0)**

**Système -> Menu : mettreAJourDisponibilité()**

**4. Diagramme d'Activités**

Activité : Mise à jour du stock et gestion de la disponibilité

1. Début
2. Mettre à jour le stock de l'article
3. Vérifier le niveau de stock
4. Marquer l'article comme indisponible (si stock = 0)
5. Mettre à jour le menu
6. Terminer

**[Début]**

**|**

**[Mettre à jour le stock de l'article]**

**|**

**[Vérifier le niveau de stock]**

**|**

**[Marquer l'article comme indisponible (si stock = 0)]**

**|**

**[Mettre à jour le menu]**

**|**

**[Terminer]**

**5. Diagramme d'États**

État : Article en fonction du stock

* État Initial : Article en stock
* État : Stock mis à jour
* État : Article en rupture de stock
* État Final : Article marqué comme indisponible

**[Article en stock] --> [Stock mis à jour]**

**[Stock mis à jour] --> [Article en rupture de stock]**

**[Article en rupture de stock] --> [Article marqué comme indisponible]**

**6. Diagramme de Communication**

Scénario : Mise à jour du stock et gestion de la disponibilité

* Employé demande à mettre à jour le stock d'un article.
* Système met à jour le stock et vérifie la disponibilité.
* Système met à jour le menu si nécessaire.

**Employé -> Système : mettreAJourStock()**

**Système -> Système : vérifierStock()**

**Système -> Menu : mettreAJourDisponibilité()**

**7. Diagramme de Composants**

**Composants :**

* Interface Utilisateur (permet aux employés de mettre à jour le stock)
* Service de Gestion du Stock (gère la logique de mise à jour du stock et la disponibilité des articles)
* Menu (affiche la disponibilité des articles)

Relations :

* Interface Utilisateur communique avec le Service de Gestion du Stock.
* Service de Gestion du Stock met à jour le Menu en fonction des informations sur la disponibilité des articles.

**+---------------------+**

**| Interface Utilisateur |**

**+---------------------+**

**|**

**v**

**+-----------------------------+**

**| Service de Gestion du Stock |**

**+-----------------------------+**

**|**

**v**

**+------------------+**

**| Menu |**

**+------------------+**

**8. Diagramme de Déploiement**

**Nœuds :**

* Serveur d'Application (héberge le Service de Gestion du Stock et l'Interface Utilisateur)
* Base de Données (stocke les informations sur les articles et leur stock)

Relations :

* Le Serveur d'Application se connecte à la Base de Données pour accéder et mettre à jour les informations sur le stock des articles.

**+---------------------+**

**| Serveur d'Application |**

**+---------------------+**

**|**

**v**

**+---------------------+**

**| Base de Données |**

**+---------------------+**

1. **Création et gestion de catégories :**

**1. Diagramme de Cas d'Utilisation**

Acteurs :

* Manager

Cas d'utilisation :

* Créer une catégorie
* Modifier une catégorie
* Supprimer une catégorie
* Associer un article à une catégorie
* Vérifier l'unicité d'une catégorie pour un article

Relations :

**[Manager] --> (Créer une catégorie)**

**[Manager] --> (Modifier une catégorie)**

**[Manager] --> (Supprimer une catégorie)**

**[Manager] --> (Associer un article à une catégorie)**

**[Système] --> (Vérifier l'unicité d'une catégorie pour un article)**

**2. Diagramme de Classes**

Classes :

* Catégorie
  + Attributs : id, nom
  + Méthodes : ajouterCatégorie(), modifierCatégorie(), supprimerCatégorie()
* Article
  + Attributs : id, nom, description, prix, catégorie
  + Méthodes : associerCatégorie(), vérifierCatégorieUnique()
* Manager (hérite de Utilisateur)
  + Méthodes : ajouterCatégorie(), modifierCatégorie(), supprimerCatégorie()

**Diagramme de Classes :**

**+------------------+**

**| Catégorie |**

**+------------------+**

**| - id |**

**| - nom |**

**+------------------+**

**| + ajouterCatégorie() |**

**| + modifierCatégorie() |**

**| + supprimerCatégorie() |**

**+------------------+**

**+------------------+**

**| Article |**

**+------------------+**

**| - id |**

**| - nom |**

**| - description |**

**| - prix |**

**| - catégorie |**

**+------------------+**

**| + associerCatégorie() |**

**| + vérifierCatégorieUnique() |**

**+------------------+**

**+------------------+**

**| Manager |**

**+------------------+**

**| + ajouterCatégorie() |**

**| + modifierCatégorie() |**

**| + supprimerCatégorie() |**

**+------------------+**

**3. Diagramme de Séquence**

Scénario : Création, gestion et association de catégories

1. Manager crée/modifie/supprime une catégorie.
2. Système enregistre les modifications dans la base de données.
3. Manager associe un article à une catégorie.
4. Système vérifie si l'article peut être associé à la catégorie sans conflit.

**Manager -> Système : créer/modifier/supprimerCatégorie()**

**Système -> Base de Données : enregistrerModificationCatégorie()**

**Manager -> Système : associerArticleACatégorie()**

**Système -> Système : vérifierCatégorieUnique()**

**Système -> Base de Données : mettreAJourAssociationArticleCatégorie()**

**4. Diagramme d'Activités**

Activité : Gestion des catégories et association des articles

1. Début
2. Créer une catégorie
3. Modifier une catégorie
4. Supprimer une catégorie
5. Associer un article à une catégorie
   * Vérifier l'unicité de la catégorie pour l'article
6. Terminer

**[Début]**

**|**

**[Créer une catégorie]**

**|**

**[Modifier une catégorie]**

**|**

**[Supprimer une catégorie]**

**|**

**[Associer un article à une catégorie]**

**|**

**[Vérifier l'unicité de la catégorie pour l'article]**

**|**

**[Terminer]**

**5. Diagramme d'États**

État : Catégorie et Association d'Articles

* État Initial : Catégorie non créée
* État : Catégorie créée
* État : Catégorie modifiée
* État : Catégorie supprimée
* État : Article associé à une catégorie
* État Final : Article correctement associé à une catégorie unique

**[Catégorie non créée] --> [Catégorie créée]**

**[Catégorie créée] --> [Catégorie modifiée]**

**[Catégorie modifiée] --> [Catégorie supprimée]**

**[Catégorie créée] --> [Article associé à une catégorie]**

**[Article associé à une catégorie] --> [Article correctement associé à une catégorie unique]**

**6. Diagramme de Communication**

Scénario : Gestion des catégories et association des articles

* Manager demande à créer, modifier ou supprimer une catégorie.
* Système met à jour les informations dans la base de données.
* Manager associe un article à une catégorie.
* Système vérifie l'unicité et met à jour l'association.

**Manager -> Système : créer/modifier/supprimerCatégorie()**

**Système -> Base de Données : enregistrerModificationCatégorie()**

**Manager -> Système : associerArticleACatégorie()**

**Système -> Système : vérifierCatégorieUnique()**

**Système -> Base de Données : mettreAJourAssociationArticleCatégorie()**

**7. Diagramme de Composants**

**Composants :**

* Interface Utilisateur (pour la gestion des catégories et l'association des articles)
* Service de Gestion des Catégories (gère la logique de création, modification, suppression et association des catégories)
* Base de Données (stocke les informations sur les catégories et les associations avec les articles)

Relations :

* Interface Utilisateur communique avec le Service de Gestion des Catégories.
* Service de Gestion des Catégories accède à la Base de Données pour les opérations de gestion des catégories et des articles.

**+---------------------+**

**| Interface Utilisateur |**

**+---------------------+**

**|**

**v**

**+-----------------------------+**

**| Service de Gestion des Catégories |**

**+-----------------------------+**

**|**

**v**

**+------------------+**

**| Base de Données |**

**+------------------+**

**8. Diagramme de Déploiement**

**Nœuds :**

* Serveur d'Application (héberge le Service de Gestion des Catégories et l'Interface Utilisateur)
* Base de Données (stocke les informations sur les catégories et les articles)

Relations :

* Le Serveur d'Application se connecte à la Base de Données pour accéder et mettre à jour les informations sur les catégories et les articles.

**+---------------------+**

**| Serveur d'Application |**

**+---------------------+**

**|**

**v**

**+---------------------+**

**| Base de Données |**

**+---------------------+**

1. **Trie et filtrage de articles :**

**1. Diagramme de Cas d'Utilisation**

Acteurs :

* Utilisateur

Cas d'utilisation :

* Trier les articles par catégorie
* Trier les articles par popularité
* Trier les articles par date d'ajout
* Trier les articles par prix
* Filtrer les articles dynamiquement
* Naviguer à travers les catégories

Relations :

**[Utilisateur] --> (Trier les articles par catégorie)**

**[Utilisateur] --> (Trier les articles par popularité)**

**[Utilisateur] --> (Trier les articles par date d'ajout)**

**[Utilisateur] --> (Trier les articles par prix)**

**[Utilisateur] --> (Filtrer les articles dynamiquement)**

**[Utilisateur] --> (Naviguer à travers les catégories)**

**2. Diagramme de Classes**

Classes :

* Article
  + Attributs : id, nom, description, prix, catégorie, popularité, dateAjout
  + Méthodes : trierParCatégorie(), trierParPopularité(), trierParDateAjout(), trierParPrix()
* Filtre
  + Attributs : critère
  + Méthodes : appliquerFiltre()
* Utilisateur
  + Méthodes : utiliserFiltre(), utiliserTri()

**Diagramme de Classes :**

**+------------------+**

**| Article |**

**+------------------+**

**| - id |**

**| - nom |**

**| - description |**

**| - prix |**

**| - catégorie |**

**| - popularité |**

**| - dateAjout |**

**+------------------+**

**| + trierParCatégorie() |**

**| + trierParPopularité() |**

**| + trierParDateAjout() |**

**| + trierParPrix() |**

**+------------------+**

**+------------------+**

**| Filtre |**

**+------------------+**

**| - critère |**

**+------------------+**

**| + appliquerFiltre() |**

**+------------------+**

**+------------------+**

**| Utilisateur |**

**+------------------+**

**| + utiliserFiltre() |**

**| + utiliserTri() |**

**+------------------+**

**3. Diagramme de Séquence**

Scénario : Tri et Filtrage des articles

1. Utilisateur choisit un critère de tri ou de filtrage.
2. Système applique le tri ou le filtre sur la liste des articles.
3. Système met à jour la vue des articles en fonction du tri ou du filtre.
4. Utilisateur navigue à travers les différentes catégories si nécessaire.

**Utilisateur -> Système : choisirCritèreDeTri/Filtrage()**

**Système -> Système : appliquerTri/Filtre()**

**Système -> Vue : mettreAJourArticles()**

**Utilisateur -> Système : naviguerCatégories()**

**4. Diagramme d'Activités**

Activité : Tri et Filtrage des articles

1. Début
2. Choisir un critère de tri ou de filtrage
3. Appliquer le tri ou le filtre
4. Mettre à jour la liste des articles affichés
5. Naviguer à travers les catégories si nécessaire
6. Terminer

**[Début]**

**|**

**[Choisir un critère de tri ou de filtrage]**

**|**

**[Appliquer le tri ou le filtre]**

**|**

**[Mettre à jour la liste des articles affichés]**

**|**

**[Naviguer à travers les catégories si nécessaire]**

**|**

**[Terminer]**

**5. Diagramme d'États**

État : Gestion du tri et du filtrage des articles

* État Initial : Liste des articles non triée/filtrée
* État : Trié par catégorie
* État : Trié par popularité
* État : Trié par date d'ajout
* État : Trié par prix
* État : Filtré dynamiquement
* État Final : Liste des articles triée et filtrée

**[Liste des articles non triée/filtrée] --> [Trié par catégorie]**

**[Liste des articles non triée/filtrée] --> [Trié par popularité]**

**[Liste des articles non triée/filtrée] --> [Trié par date d'ajout]**

**[Liste des articles non triée/filtrée] --> [Trié par prix]**

**[Liste des articles non triée/filtrée] --> [Filtré dynamiquement]**

**[Trié par catégorie] --> [Liste des articles triée et filtrée]**

**[Trié par popularité] --> [Liste des articles triée et filtrée]**

**[Trié par date d'ajout] --> [Liste des articles triée et filtrée]**

**[Trié par prix] --> [Liste des articles triée et filtrée]**

**[Filtré dynamiquement] --> [Liste des articles triée et filtrée]**

**6. Diagramme de Communication**

Scénario : Application du tri et du filtrage

* Utilisateur choisit un critère de tri ou de filtrage.
* Système applique le critère et met à jour la vue des articles.
* Utilisateur navigue à travers les catégories si nécessaire.

**Utilisateur -> Système : choisirCritèreDeTri/Filtrage()**

**Système -> Vue : mettreAJourArticles()**

**Utilisateur -> Système : naviguerCatégories()**

**7. Diagramme de Composants**

**Composants :**

* Interface Utilisateur (permet aux utilisateurs de sélectionner les critères de tri et de filtrage)
* Service de Tri et Filtrage (gère la logique de tri et de filtrage des articles)
* Vue des Articles (affiche les articles triés et filtrés)

Relations :

* Interface Utilisateur communique avec le Service de Tri et Filtrage.
* Service de Tri et Filtrage met à jour la Vue des Articles.

**+---------------------+**

**| Interface Utilisateur |**

**+---------------------+**

**|**

**v**

**+-----------------------------+**

**| Service de Tri et Filtrage |**

**+-----------------------------+**

**|**

**v**

**+------------------+**

**| Vue des Articles |**

**+------------------+**

**8. Diagramme de Déploiement**

**Nœuds :**

* Serveur d'Application (héberge le Service de Tri et Filtrage et l'Interface Utilisateur)
* Base de Données (stocke les informations sur les articles)

Relations :

* Le Serveur d'Application se connecte à la Base de Données pour récupérer et mettre à jour les informations sur les articles en fonction du tri et du filtrage.

**+---------------------+**

**| Serveur d'Application |**

**+---------------------+**

**|**

**v**

**+---------------------+**

**| Base de Données |**

**+---------------------+**

1. **Rapport de vente par catégories :**

**1**. Diagramme de Cas d'Utilisation

Acteurs :

* Manager

Cas d'utilisation :

* Générer un rapport financier des ventes par catégorie
* Voir les articles les plus vendus par catégorie
* Voir les articles les moins vendus par catégorie
* Mettre à jour les quantités de stock en temps réel

Relations :

**[Manager] --> (Générer un rapport financier des ventes par catégorie)**

**[Manager] --> (Voir les articles les plus vendus par catégorie)**

**[Manager] --> (Voir les articles les moins vendus par catégorie)**

**[Système] --> (Mettre à jour les quantités de stock en temps réel)**

**2. Diagramme de Classes**

Classes :

* RapportVentes
  + Attributs : id, date, catégorie, ventesTotales
  + Méthodes : générerRapport(), afficherArticlesLesPlusVendus(), afficherArticlesLesMoinsVendus()
* Article
  + Attributs : id, nom, catégorie, quantiteVendue
  + Méthodes : mettreAJourQuantité()
* Manager (hérite de Utilisateur)
  + Méthodes : générerRapportVentes()
* Stock
  + Attributs : id, article, quantité
  + Méthodes : mettreAJourQuantitéEnTempsRéel()

**Diagramme de Classes :**

**+--------------------+**

**| RapportVentes |**

**+--------------------+**

**| - id |**

**| - date |**

**| - catégorie |**

**| - ventesTotales |**

**+--------------------+**

**| + générerRapport() |**

**| + afficherArticlesLesPlusVendus() |**

**| + afficherArticlesLesMoinsVendus() |**

**+--------------------+**

**+--------------------+**

**| Article |**

**+--------------------+**

**| - id |**

**| - nom |**

**| - catégorie |**

**| - quantiteVendue |**

**+--------------------+**

**| + mettreAJourQuantité() |**

**+--------------------+**

**+--------------------+**

**| Manager |**

**+--------------------+**

**| + générerRapportVentes() |**

**+--------------------+**

**+--------------------+**

**| Stock |**

**+--------------------+**

**| - id |**

**| - article |**

**| - quantité |**

**+--------------------+**

**| + mettreAJourQuantitéEnTempsRéel() |**

**+--------------------+**

**3. Diagramme de Séquence**

Scénario : Génération de rapports de vente

1. Manager demande un rapport financier des ventes par catégorie.
2. Système génère le rapport en calculant les ventes totales par catégorie.
3. Système identifie les articles les plus et les moins vendus.
4. Système met à jour les quantités de stock en temps réel.
5. Système fournit le rapport au Manager.

**Manager -> Système : demanderRapportVentesParCatégorie()**

**Système -> Système : calculerVentesTotalesParCatégorie()**

**Système -> Système : identifierArticlesLesPlusVendus()**

**Système -> Système : identifierArticlesLesMoinsVendus()**

**Système -> Stock : mettreAJourQuantitéEnTempsRéel()**

**Système -> Manager : fournirRapport()**

**4. Diagramme d'Activités**

Activité : Génération des rapports de vente et mise à jour du stock

1. Début
2. Demander un rapport financier des ventes par catégorie
3. Calculer les ventes totales par catégorie
4. Identifier les articles les plus et les moins vendus
5. Mettre à jour les quantités de stock en temps réel
6. Générer et fournir le rapport au Manager
7. Terminer

**[Début]**

**|**

**[Demander un rapport financier des ventes par catégorie]**

**|**

**[Calculer les ventes totales par catégorie]**

**|**

**[Identifier les articles les plus et les moins vendus]**

**|**

**[Mettre à jour les quantités de stock en temps réel]**

**|**

**[Générer et fournir le rapport au Manager]**

**|**

**[Terminer]**

**5. Diagramme d'États**

État : Rapport de Vente et Stock

* État Initial : Rapport non généré
* État : Rapport financier des ventes par catégorie généré
* État : Articles les plus vendus identifiés
* État : Articles les moins vendus identifiés
* État : Quantités de stock mises à jour en temps réel
* État Final : Rapport fourni au Manager avec les informations mises à jour

**[Rapport non généré] --> [Rapport financier des ventes par catégorie généré]**

**[Rapport financier des ventes par catégorie généré] --> [Articles les plus vendus identifiés]**

**[Articles les plus vendus identifiés] --> [Articles les moins vendus identifiés]**

**[Articles les moins vendus identifiés] --> [Quantités de stock mises à jour en temps réel]**

**[Quantités de stock mises à jour en temps réel] --> [Rapport fourni au Manager]**

**6. Diagramme de Communication**

**Scénario : Génération des rapports de vente et mise à jour du stock**

* **Manager demande un rapport financier.**
* **Système génère le rapport financier.**
* **Système identifie les articles les plus et les moins vendus.**
* **Système met à jour le stock en temps réel.**
* **Système fournit le rapport au Manager.**

vbnet

Copier le code

**Manager -> Système : demanderRapportVentesParCatégorie()**

**Système -> Système : calculerVentesTotales()**

**Système -> Système : identifierArticlesLesPlusVendus()**

**Système -> Système : identifierArticlesLesMoinsVendus()**

**Système -> Stock : mettreAJourQuantitéEnTempsRéel()**

**Système -> Manager : fournirRapport()**

**7. Diagramme de Composants**

Composants :

* Interface Utilisateur (permet au Manager de demander des rapports)
* Service de Génération de Rapports (gère la logique de calcul des ventes, identification des articles les plus/moins vendus, et mise à jour du stock)
* Base de Données (stocke les informations sur les ventes, les articles et les stocks)

Relations :

* Interface Utilisateur communique avec le Service de Génération de Rapports.
* Service de Génération de Rapports accède à la Base de Données pour générer des rapports et mettre à jour les stocks.

**+---------------------+**

**| Interface Utilisateur |**

**+---------------------+**

**|**

**v**

**+-----------------------------+**

**| Service de Génération de Rapports |**

**+-----------------------------+**

**|**

**v**

**+------------------+**

**| Base de Données |**

**+------------------+**

**8. Diagramme de Déploiement**

**Nœuds :**

* Serveur d'Application (héberge le Service de Génération de Rapports et l'Interface Utilisateur)
* Base de Données (stocke les informations sur les ventes, les articles et les stocks)

Relations :

* Le Serveur d'Application se connecte à la Base de Données pour récupérer et mettre à jour les informations nécessaires aux rapports.

**+---------------------+**

**| Serveur d'Application |**

**+---------------------+**

**|**

**v**

**+---------------------+**

**| Base de Données |**

**+---------------------+**

1. **Gestion Comptabilité :**
2. **Facturation :**

**1. Diagramme de Cas d'Utilisation**

**Acteurs :**

* **Système de Paiement**
* **Utilisateur (Client)**

**Cas d'utilisation :**

* **Générer une facture après confirmation du paiement**
* **Inclure les détails des articles commandés, les quantités, les prix et le total dans la facture**

**Relations :**

**scss**

**Copier le code**

**[Système de Paiement] --> (Générer une facture après confirmation du paiement)**

**[Utilisateur] --> (Recevoir la facture)**

**[Système] --> (Inclure les détails des articles, les quantités, les prix et le total dans la facture)**

**2. Diagramme de Classes**

**Classes :**

* **Commande**
  + **Attributs : id, date, montantTotal**
  + **Méthodes : confirmerPaiement(), générerFacture()**
* **Article**
  + **Attributs : id, nom, prix**
  + **Méthodes : obtenirDétails()**
* **Facture**
  + **Attributs : id, commande, détailsArticles, quantités, prix, total**
  + **Méthodes : créerFacture()**
* **Système de Paiement**
  + **Méthodes : confirmerPaiement()**

**Diagramme de Classes :**

**lua**

**Copier le code**

**+--------------------+**

**| Commande |**

**+--------------------+**

**| - id |**

**| - date |**

**| - montantTotal |**

**+--------------------+**

**| + confirmerPaiement() |**

**| + générerFacture() |**

**+--------------------+**

**+--------------------+**

**| Article |**

**+--------------------+**

**| - id |**

**| - nom |**

**| - prix |**

**+--------------------+**

**| + obtenirDétails() |**

**+--------------------+**

**+--------------------+**

**| Facture |**

**+--------------------+**

**| - id |**

**| - commande |**

**| - détailsArticles |**

**| - quantités |**

**| - prix |**

**| - total |**

**+--------------------+**

**| + créerFacture() |**

**+--------------------+**

**+--------------------+**

**| Système de Paiement |**

**+--------------------+**

**| + confirmerPaiement() |**

**+--------------------+**

**3. Diagramme de Séquence**

**Scénario : Génération de la facture après confirmation du paiement**

1. **Système de Paiement confirme le paiement.**
2. **Commande reçoit la confirmation du paiement.**
3. **Commande appelle la méthode générerFacture().**
4. **Facture est créée avec les détails des articles, les quantités, les prix et le total.**
5. **Facture est envoyée à l'Utilisateur.**

**css**

**Copier le code**

**Système de Paiement -> Commande : confirmerPaiement()**

**Commande -> Commande : générerFacture()**

**Commande -> Facture : créerFacture()**

**Facture -> Utilisateur : envoyerFacture()**

**4. Diagramme d'Activités**

**Activité : Génération de la facture**

1. **Début**
2. **Confirmation du paiement par le Système de Paiement**
3. **Commande reçoit la confirmation**
4. **Commande génère la facture**
5. **Création de la facture avec détails des articles, quantités, prix, et total**
6. **Envoi de la facture à l'Utilisateur**
7. **Terminer**

**csharp**

**Copier le code**

**[Début]**

**|**

**[Confirmation du paiement par le Système de Paiement]**

**|**

**[Commande reçoit la confirmation]**

**|**

**[Commande génère la facture]**

**|**

**[Création de la facture avec détails des articles, quantités, prix, et total]**

**|**

**[Envoi de la facture à l'Utilisateur]**

**|**

**[Terminer]**

**5. Diagramme d'États**

**État : Facturation**

* **État Initial : Commande non facturée**
* **État : Paiement confirmé**
* **État : Facture générée**
* **État : Facture envoyée à l'Utilisateur**
* **État Final : Facturation terminée**

**css**

**Copier le code**

**[Commande non facturée] --> [Paiement confirmé]**

**[Paiement confirmé] --> [Facture générée]**

**[Facture générée] --> [Facture envoyée à l'Utilisateur]**

**[Facture envoyée à l'Utilisateur] --> [Facturation terminée]**

**6. Diagramme de Communication**

**Scénario : Génération de la facture après confirmation du paiement**

* **Système de Paiement confirme le paiement.**
* **Commande reçoit la confirmation et génère la facture.**
* **Facture est créée avec les détails nécessaires.**
* **Facture est envoyée à l'Utilisateur.**

**css**

**Copier le code**

**Système de Paiement -> Commande : confirmerPaiement()**

**Commande -> Facture : générerFacture()**

**Facture -> Utilisateur : envoyerFacture()**

**7. Diagramme de Composants**

**Composants :**

* **Interface Utilisateur (permet à l'Utilisateur de recevoir la facture)**
* **Service de Gestion des Commandes (gère la génération de la facture)**
* **Système de Paiement (confirme le paiement)**
* **Base de Données (stocke les informations des commandes, des articles et des factures)**

**Relations :**

* **Interface Utilisateur communique avec le Service de Gestion des Commandes.**
* **Service de Gestion des Commandes interagit avec le Système de Paiement et la Base de Données pour générer et envoyer la facture.**

**diff**

**Copier le code**

**+---------------------+**

**| Interface Utilisateur |**

**+---------------------+**

**|**

**v**

**+------------------------------+**

**| Service de Gestion des Commandes |**

**+------------------------------+**

**|**

**v**

**+---------------------+**

**| Système de Paiement |**

**+---------------------+**

**|**

**v**

**+------------------+**

**| Base de Données |**

**+------------------+**

**8. Diagramme de Déploiement**

**Nœuds :**

* **Serveur d'Application (héberge le Service de Gestion des Commandes et l'Interface Utilisateur)**
* **Base de Données (stocke les informations sur les commandes, les articles et les factures)**

**Relations :**

* **Le Serveur d'Application se connecte à la Base de Données pour accéder et mettre à jour les informations nécessaires à la génération des factures.**

**diff**

**Copier le code**

**+---------------------+**

**| Serveur d'Application |**

**+---------------------+**

**|**

**v**

**+---------------------+**

**| Base de Données |**

**+---------------------+**

1. **Transaction de paiement :**

**1. Diagramme de Cas d'Utilisation**

**Acteurs :**

* **Utilisateur (Client)**
* **Système de Paiement**

**Cas d'utilisation :**

* **Effectuer un paiement via Airtel Money**
* **Effectuer un paiement via Mobile Money**
* **Enregistrer une transaction de paiement**
* **Fournir un identifiant unique pour chaque transaction**
* **Enregistrer la date et le montant de la transaction**

**Relations :**

**scss**

**Copier le code**

**[Utilisateur] --> (Effectuer un paiement via Airtel Money)**

**[Utilisateur] --> (Effectuer un paiement via Mobile Money)**

**[Système de Paiement] --> (Enregistrer une transaction de paiement)**

**[Système de Paiement] --> (Fournir un identifiant unique pour chaque transaction)**

**[Système de Paiement] --> (Enregistrer la date et le montant de la transaction)**

**2. Diagramme de Classes**

**Classes :**

* **TransactionPaiement**
  + **Attributs : id, date, montant, méthodePaiement**
  + **Méthodes : enregistrerTransaction(), générerIdentifiantUnique()**
* **Utilisateur**
  + **Attributs : id, nom**
  + **Méthodes : effectuerPaiement()**
* **SystèmePaiement**
  + **Méthodes : traiterPaiementAirtel(), traiterPaiementMobile()**

**Diagramme de Classes :**

**lua**

**Copier le code**

**+----------------------+**

**| TransactionPaiement |**

**+----------------------+**

**| - id |**

**| - date |**

**| - montant |**

**| - méthodePaiement |**

**+----------------------+**

**| + enregistrerTransaction() |**

**| + générerIdentifiantUnique() |**

**+----------------------+**

**+----------------------+**

**| Utilisateur |**

**+----------------------+**

**| - id |**

**| - nom |**

**+----------------------+**

**| + effectuerPaiement() |**

**+----------------------+**

**+----------------------+**

**| SystèmePaiement |**

**+----------------------+**

**| + traiterPaiementAirtel() |**

**| + traiterPaiementMobile() |**

**+----------------------+**

**3. Diagramme de Séquence**

**Scénario : Traitement et enregistrement d'une transaction de paiement**

1. **Utilisateur choisit une méthode de paiement (Airtel Money ou Mobile Money).**
2. **Système de Paiement traite la demande de paiement.**
3. **Système de Paiement enregistre la transaction avec un identifiant unique, la date et le montant.**
4. **Système de Paiement confirme le paiement à l'Utilisateur.**

**css**

**Copier le code**

**Utilisateur -> SystèmePaiement : choisirMéthodePaiement()**

**SystèmePaiement -> SystèmePaiement : traiterPaiement()**

**SystèmePaiement -> TransactionPaiement : enregistrerTransaction()**

**SystèmePaiement -> TransactionPaiement : générerIdentifiantUnique()**

**TransactionPaiement -> Utilisateur : confirmerPaiement()**

**4. Diagramme d'Activités**

**Activité : Traitement et enregistrement des transactions de paiement**

1. **Début**
2. **Utilisateur choisit la méthode de paiement**
3. **Système de Paiement traite la demande**
4. **Enregistrement de la transaction avec identifiant unique, date et montant**
5. **Confirmation de la transaction à l'Utilisateur**
6. **Terminer**

**csharp**

**Copier le code**

**[Début]**

**|**

**[Utilisateur choisit la méthode de paiement]**

**|**

**[Système de Paiement traite la demande]**

**|**

**[Enregistrement de la transaction avec identifiant unique, date et montant]**

**|**

**[Confirmation de la transaction à l'Utilisateur]**

**|**

**[Terminer]**

**5. Diagramme d'États**

**État : Transaction de Paiement**

* **État Initial : Transaction non initiée**
* **État : Méthode de paiement sélectionnée**
* **État : Paiement traité**
* **État : Transaction enregistrée avec identifiant, date et montant**
* **État Final : Transaction confirmée**

**css**

**Copier le code**

**[Transaction non initiée] --> [Méthode de paiement sélectionnée]**

**[Méthode de paiement sélectionnée] --> [Paiement traité]**

**[Paiement traité] --> [Transaction enregistrée avec identifiant, date et montant]**

**[Transaction enregistrée avec identifiant, date et montant] --> [Transaction confirmée]**

**6. Diagramme de Communication**

**Scénario : Traitement des transactions de paiement**

* **Utilisateur choisit une méthode de paiement.**
* **Système de Paiement traite la demande et enregistre la transaction.**
* **TransactionPaiement est créée avec un identifiant, une date et un montant.**
* **Système de Paiement confirme la transaction à l'Utilisateur.**

**css**

**Copier le code**

**Utilisateur -> SystèmePaiement : choisirMéthodePaiement()**

**SystèmePaiement -> TransactionPaiement : enregistrerTransaction()**

**TransactionPaiement -> SystèmePaiement : générerIdentifiantUnique()**

**SystèmePaiement -> Utilisateur : confirmerPaiement()**

**7. Diagramme de Composants**

**Composants :**

* **Interface Utilisateur (permet à l'Utilisateur de choisir la méthode de paiement)**
* **Service de Traitement des Transactions (gère le traitement et l'enregistrement des paiements)**
* **Base de Données (stocke les informations sur les transactions)**

**Relations :**

* **Interface Utilisateur communique avec le Service de Traitement des Transactions.**
* **Service de Traitement des Transactions interagit avec la Base de Données pour enregistrer les transactions.**

**diff**

**Copier le code**

**+---------------------+**

**| Interface Utilisateur |**

**+---------------------+**

**|**

**v**

**+-----------------------------+**

**| Service de Traitement des Transactions |**

**+-----------------------------+**

**|**

**v**

**+------------------+**

**| Base de Données |**

**+------------------+**

**8. Diagramme de Déploiement**

**Nœuds :**

* **Serveur d'Application (héberge le Service de Traitement des Transactions et l'Interface Utilisateur)**
* **Base de Données (stocke les informations sur les transactions)**

**Relations :**

* **Le Serveur d'Application se connecte à la Base de Données pour enregistrer les transactions et accéder aux informations nécessaires.**

**diff**

**Copier le code**

**+---------------------+**

**| Serveur d'Application |**

**+---------------------+**

**|**

**v**

**+---------------------+**

**| Base de Données |**

**+---------------------+**

1. **Rapport Financier :**

**1. Diagramme de Cas d'Utilisation**

**Acteurs :**

* **Manager**

**Cas d'utilisation :**

* **Générer des rapports financiers**
* **Suivre les ventes**
* **Suivre les revenus**
* **Suivre les dépenses**
* **Choisir la fréquence de génération des rapports (quotidienne, hebdomadaire, mensuelle, annuelle)**

**Relations :**

**scss**

**Copier le code**

**[Manager] --> (Générer des rapports financiers)**

**[Manager] --> (Suivre les ventes)**

**[Manager] --> (Suivre les revenus)**

**[Manager] --> (Suivre les dépenses)**

**[Manager] --> (Choisir la fréquence de génération des rapports)**

**2. Diagramme de Classes**

**Classes :**

* **RapportFinancier**
  + **Attributs : id, typeRapport, dateDébut, dateFin, ventes, revenus, dépenses**
  + **Méthodes : générerRapport(), définirFréquence()**
* **Manager**
  + **Attributs : id, nom**
  + **Méthodes : demanderRapport()**
* **SystèmeRapports**
  + **Méthodes : générerRapportQuotidien(), générerRapportHebdomadaire(), générerRapportMensuel(), générerRapportAnnuel()**

**Diagramme de Classes :**

**scss**

**Copier le code**

**+----------------------+**

**| RapportFinancier |**

**+----------------------+**

**| - id |**

**| - typeRapport |**

**| - dateDébut |**

**| - dateFin |**

**| - ventes |**

**| - revenus |**

**| - dépenses |**

**+----------------------+**

**| + générerRapport() |**

**| + définirFréquence() |**

**+----------------------+**

**+----------------------+**

**| Manager |**

**+----------------------+**

**| - id |**

**| - nom |**

**+----------------------+**

**| + demanderRapport() |**

**+----------------------+**

**+----------------------+**

**| SystèmeRapports |**

**+----------------------+**

**| + générerRapportQuotidien() |**

**| + générerRapportHebdomadaire() |**

**| + générerRapportMensuel() |**

**| + générerRapportAnnuel() |**

**+----------------------+**

**3. Diagramme de Séquence**

**Scénario : Génération d'un rapport financier**

1. **Manager demande un rapport financier.**
2. **Système de Rapports reçoit la demande et vérifie la fréquence de génération.**
3. **Système de Rapports génère le rapport en fonction de la fréquence choisie.**
4. **Système de Rapports compile les données (ventes, revenus, dépenses).**
5. **Système de Rapports fournit le rapport au Manager.**

**css**

**Copier le code**

**Manager -> SystèmeRapports : demanderRapport()**

**SystèmeRapports -> SystèmeRapports : vérifierFréquence()**

**SystèmeRapports -> SystèmeRapports : générerRapport()**

**SystèmeRapports -> SystèmeRapports : compilerDonnées()**

**SystèmeRapports -> Manager : fournirRapport()**

**4. Diagramme d'Activités**

**Activité : Génération des rapports financiers**

1. **Début**
2. **Manager demande un rapport financier**
3. **Vérification de la fréquence de génération**
4. **Génération du rapport (quotidienne, hebdomadaire, mensuelle, annuelle)**
5. **Compilation des données de ventes, revenus, dépenses**
6. **Fourniture du rapport au Manager**
7. **Terminer**

**csharp**

**Copier le code**

**[Début]**

**|**

**[Manager demande un rapport financier]**

**|**

**[Vérification de la fréquence de génération]**

**|**

**[Génération du rapport]**

**|**

**[Compilation des données de ventes, revenus, dépenses]**

**|**

**[Fourniture du rapport au Manager]**

**|**

**[Terminer]**

**5. Diagramme d'États**

**État : Rapport Financier**

* **État Initial : Rapport non généré**
* **État : Demande de rapport reçue**
* **État : Vérification de la fréquence de génération**
* **État : Rapport généré**
* **État : Rapport compilé avec ventes, revenus, et dépenses**
* **État Final : Rapport fourni au Manager**

**css**

**Copier le code**

**[Rapport non généré] --> [Demande de rapport reçue]**

**[Demande de rapport reçue] --> [Vérification de la fréquence de génération]**

**[Vérification de la fréquence de génération] --> [Rapport généré]**

**[Rapport généré] --> [Rapport compilé avec ventes, revenus, et dépenses]**

**[Rapport compilé avec ventes, revenus, et dépenses] --> [Rapport fourni au Manager]**

**6. Diagramme de Communication**

**Scénario : Génération d'un rapport financier**

* **Manager demande un rapport financier.**
* **Système de Rapports vérifie la fréquence et génère le rapport.**
* **Système de Rapports compile les données nécessaires.**
* **Système de Rapports fournit le rapport au Manager.**

**css**

**Copier le code**

**Manager -> SystèmeRapports : demanderRapport()**

**SystèmeRapports -> SystèmeRapports : vérifierFréquence()**

**SystèmeRapports -> SystèmeRapports : générerRapport()**

**SystèmeRapports -> SystèmeRapports : compilerDonnées()**

**SystèmeRapports -> Manager : fournirRapport()**

**7. Diagramme de Composants**

**Composants :**

* **Interface Utilisateur (permet au Manager de demander des rapports)**
* **Service de Génération des Rapports (gère la logique de génération des rapports financiers)**
* **Base de Données (stocke les informations sur les ventes, les revenus, et les dépenses)**

**Relations :**

* **Interface Utilisateur communique avec le Service de Génération des Rapports.**
* **Service de Génération des Rapports accède à la Base de Données pour obtenir les données nécessaires.**

**diff**

**Copier le code**

**+---------------------+**

**| Interface Utilisateur |**

**+---------------------+**

**|**

**v**

**+------------------------------+**

**| Service de Génération des Rapports |**

**+------------------------------+**

**|**

**v**

**+------------------+**

**| Base de Données |**

**+------------------+**

**8. Diagramme de Déploiement**

**Nœuds :**

* **Serveur d'Application (héberge le Service de Génération des Rapports et l'Interface Utilisateur)**
* **Base de Données (stocke les informations nécessaires à la génération des rapports financiers)**

**Relations :**

* **Le Serveur d'Application se connecte à la Base de Données pour récupérer les données nécessaires à la génération des rapports.**

**diff**

**Copier le code**

**+---------------------+**

**| Serveur d'Application |**

**+---------------------+**

**|**

**v**

**+---------------------+**

**| Base de Données |**

**+---------------------+**

1. **Géolocalisation :**
2. **Service de Livraison :**

**1. Diagramme de Cas d'Utilisation**

**Acteurs :**

* **Livreur**
* **Système de Géolocalisation**

**Cas d'utilisation :**

* **Accéder aux itinéraires de livraison via le service de géolocalisation**
* **Associer une livraison à une adresse de livraison valide**

**Relations :**

**scss**

**Copier le code**

**[Livreur] --> (Accéder aux itinéraires de livraison via le service de géolocalisation)**

**[Livreur] --> (Associer une livraison à une adresse de livraison valide)**

**[Système de Géolocalisation] --> (Fournir les itinéraires de livraison)**

**2. Diagramme de Classes**

**Classes :**

* **Livraison**
  + **Attributs : id, adresse, itinéraire, statut**
  + **Méthodes : assignerAdresse(), obtenirItinéraire()**
* **Adresse**
  + **Attributs : rue, ville, codePostal, pays**
  + **Méthodes : vérifierValidité()**
* **Itinéraire**
  + **Attributs : départ, destination, étapes**
  + **Méthodes : calculerItinéraire()**
* **SystèmeGéolocalisation**
  + **Méthodes : fournirItinéraire()**

**Diagramme de Classes :**

**diff**

**Copier le code**

**+------------------+**

**| Livraison |**

**+------------------+**

**| - id |**

**| - adresse |**

**| - itinéraire |**

**| - statut |**

**+------------------+**

**| + assignerAdresse() |**

**| + obtenirItinéraire() |**

**+------------------+**

**+------------------+**

**| Adresse |**

**+------------------+**

**| - rue |**

**| - ville |**

**| - codePostal |**

**| - pays |**

**+------------------+**

**| + vérifierValidité() |**

**+------------------+**

**+------------------+**

**| Itinéraire |**

**+------------------+**

**| - départ |**

**| - destination |**

**| - étapes |**

**+------------------+**

**| + calculerItinéraire() |**

**+------------------+**

**+------------------+**

**| SystèmeGéolocalisation |**

**+------------------+**

**| + fournirItinéraire() |**

**+------------------+**

**3. Diagramme de Séquence**

**Scénario : Accès aux itinéraires de livraison**

1. **Livreur demande l'itinéraire de livraison.**
2. **Système de Géolocalisation reçoit la demande et calcule l'itinéraire.**
3. **Système de Géolocalisation renvoie l'itinéraire au Livreur.**
4. **Livreur utilise l'itinéraire pour effectuer la livraison.**

**css**

**Copier le code**

**Livreur -> SystèmeGéolocalisation : demanderItinéraire()**

**SystèmeGéolocalisation -> SystèmeGéolocalisation : calculerItinéraire()**

**SystèmeGéolocalisation -> Livreur : fournirItinéraire()**

**Livreur -> Livreur : utiliserItinéraire()**

**4. Diagramme d'Activités**

**Activité : Accès aux itinéraires de livraison**

1. **Début**
2. **Livreur demande l'itinéraire pour une livraison**
3. **Système de Géolocalisation reçoit la demande**
4. **Calcul de l'itinéraire basé sur l'adresse de livraison**
5. **Fourniture de l'itinéraire au Livreur**
6. **Utilisation de l'itinéraire pour la livraison**
7. **Terminer**

**csharp**

**Copier le code**

**[Début]**

**|**

**[Livreur demande l'itinéraire pour une livraison]**

**|**

**[Système de Géolocalisation reçoit la demande]**

**|**

**[Calcul de l'itinéraire basé sur l'adresse de livraison]**

**|**

**[Fourniture de l'itinéraire au Livreur]**

**|**

**[Utilisation de l'itinéraire pour la livraison]**

**|**

**[Terminer]**

**5. Diagramme d'États**

**État : Livraison**

* **État Initial : Livraison non assignée**
* **État : Adresse de livraison validée**
* **État : Itinéraire calculé**
* **État : Livraison en cours**
* **État Final : Livraison terminée**

**css**

**Copier le code**

**[Livraison non assignée] --> [Adresse de livraison validée]**

**[Adresse de livraison validée] --> [Itinéraire calculé]**

**[Itinéraire calculé] --> [Livraison en cours]**

**[Livraison en cours] --> [Livraison terminée]**

**6. Diagramme de Communication**

**Scénario : Processus de livraison**

* **Livreur demande un itinéraire de livraison.**
* **Système de Géolocalisation calcule et fournit l'itinéraire basé sur l'adresse de livraison.**
* **Livreur utilise l'itinéraire pour effectuer la livraison.**

**css**

**Copier le code**

**Livreur -> SystèmeGéolocalisation : demanderItinéraire()**

**SystèmeGéolocalisation -> SystèmeGéolocalisation : calculerItinéraire()**

**SystèmeGéolocalisation -> Livreur : fournirItinéraire()**

**Livreur -> Livreur : utiliserItinéraire()**

**7. Diagramme de Composants**

**Composants :**

* **Interface Livreur (permet au livreur de demander et de voir les itinéraires)**
* **Service de Géolocalisation (calcule les itinéraires de livraison)**
* **Base de Données des Adresses (stocke les adresses de livraison)**

**Relations :**

* **Interface Livreur communique avec le Service de Géolocalisation.**
* **Service de Géolocalisation accède à la Base de Données des Adresses pour valider les adresses et calculer les itinéraires.**

**diff**

**Copier le code**

**+---------------------+**

**| Interface Livreur |**

**+---------------------+**

**|**

**v**

**+-----------------------------+**

**| Service de Géolocalisation |**

**+-----------------------------+**

**|**

**v**

**+-----------------------------+**

**| Base de Données des Adresses |**

**+-----------------------------+**

**8. Diagramme de Déploiement**

**Nœuds :**

* **Serveur d'Application (héberge le Service de Géolocalisation et l'Interface Livreur)**
* **Base de Données des Adresses (stocke les adresses nécessaires à la livraison)**

**Relations :**

* **Le Serveur d'Application se connecte à la Base de Données des Adresses pour récupérer les informations nécessaires à la géolocalisation et à la gestion des itinéraires.**

**diff**

**Copier le code**

**+---------------------+**

**| Serveur d'Application |**

**+---------------------+**

**|**

**v**

**+-----------------------------+**

**| Base de Données des Adresses |**

**+-----------------------------+**

1. **Suivi de Livraison :**

**1. Diagramme de Cas d'Utilisation**

**Acteurs :**

* **Client**
* **Livreur**
* **Système de Suivi des Livraisons**

**Cas d'utilisation :**

* **Suivre l'état de la livraison en temps réel**
* **Mettre à jour le statut de la livraison (début et fin)**

**Relations :**

**scss**

**Copier le code**

**[Client] --> (Suivre l'état de la livraison en temps réel)**

**[Livreur] --> (Mettre à jour le statut de la livraison)**

**[Système de Suivi des Livraisons] --> (Gérer les mises à jour de statut)**

**[Système de Suivi des Livraisons] --> (Afficher l'état de la livraison en temps réel)**

**2. Diagramme de Classes**

**Classes :**

* **Livraison**
  + **Attributs : id, statut, adresse, itinéraire**
  + **Méthodes : mettreÀJourStatut(), obtenirStatut()**
* **Client**
  + **Attributs : id, nom, email**
  + **Méthodes : suivreLivraison()**
* **Livreur**
  + **Attributs : id, nom**
  + **Méthodes : débuterLivraison(), terminerLivraison()**
* **SystèmeSuiviLivraison**
  + **Attributs : listeLivraisons**
  + **Méthodes : afficherStatutEnTempsRéel(), mettreÀJourStatutLivraison()**

**Diagramme de Classes :**

**diff**

**Copier le code**

**+------------------+**

**| Livraison |**

**+------------------+**

**| - id |**

**| - statut |**

**| - adresse |**

**| - itinéraire |**

**+------------------+**

**| + mettreÀJourStatut() |**

**| + obtenirStatut() |**

**+------------------+**

**+------------------+**

**| Client |**

**+------------------+**

**| - id |**

**| - nom |**

**| - email |**

**+------------------+**

**| + suivreLivraison() |**

**+------------------+**

**+------------------+**

**| Livreur |**

**+------------------+**

**| - id |**

**| - nom |**

**+------------------+**

**| + débuterLivraison() |**

**| + terminerLivraison() |**

**+------------------+**

**+-------------------------------+**

**| SystèmeSuiviLivraison |**

**+-------------------------------+**

**| - listeLivraisons |**

**+-------------------------------+**

**| + afficherStatutEnTempsRéel() |**

**| + mettreÀJourStatutLivraison() |**

**+-------------------------------+**

**3. Diagramme de Séquence**

**Scénario : Suivi des livraisons**

1. **Client demande le suivi de la livraison.**
2. **Système de Suivi des Livraisons vérifie l'état de la livraison.**
3. **Système de Suivi des Livraisons fournit l'état actuel de la livraison au Client.**
4. **Livreur commence la livraison et met à jour le statut.**
5. **Système de Suivi des Livraisons enregistre la mise à jour du statut.**
6. **Livreur termine la livraison et met à jour le statut.**
7. **Système de Suivi des Livraisons enregistre la mise à jour finale.**

**css**

**Copier le code**

**Client -> SystèmeSuiviLivraison : demanderSuivi()**

**SystèmeSuiviLivraison -> Client : fournirStatutActuel()**

**Livreur -> SystèmeSuiviLivraison : mettreÀJourStatutDébut()**

**SystèmeSuiviLivraison -> SystèmeSuiviLivraison : enregistrerStatutDébut()**

**Livreur -> SystèmeSuiviLivraison : mettreÀJourStatutFin()**

**SystèmeSuiviLivraison -> SystèmeSuiviLivraison : enregistrerStatutFin()**

**4. Diagramme d'Activités**

**Activité : Suivi et mise à jour des livraisons**

1. **Début**
2. **Client demande le suivi de la livraison**
3. **Système de Suivi des Livraisons récupère l'état actuel**
4. **Système de Suivi des Livraisons fournit l'état au Client**
5. **Livreur commence la livraison**
6. **Livreur met à jour le statut de début**
7. **Système de Suivi des Livraisons enregistre la mise à jour**
8. **Livreur termine la livraison**
9. **Livreur met à jour le statut de fin**
10. **Système de Suivi des Livraisons enregistre la mise à jour finale**
11. **Terminer**

**csharp**

**Copier le code**

**[Début]**

**|**

**[Client demande le suivi de la livraison]**

**|**

**[Système de Suivi des Livraisons récupère l'état actuel]**

**|**

**[Système de Suivi des Livraisons fournit l'état au Client]**

**|**

**[Livreur commence la livraison]**

**|**

**[Livreur met à jour le statut de début]**

**|**

**[Système de Suivi des Livraisons enregistre la mise à jour]**

**|**

**[Livreur termine la livraison]**

**|**

**[Livreur met à jour le statut de fin]**

**|**

**[Système de Suivi des Livraisons enregistre la mise à jour finale]**

**|**

**[Terminer]**

**5. Diagramme d'États**

**État : Livraison**

* **État Initial : Livraison non commencée**
* **État : Livraison en cours (statut mis à jour au début)**
* **État : Livraison terminée (statut mis à jour à la fin)**
* **État Final : Livraison complétée**

**css**

**Copier le code**

**[Livraison non commencée] --> [Livraison en cours]**

**[Livraison en cours] --> [Livraison terminée]**

**[Livraison terminée] --> [Livraison complétée]**

**6. Diagramme de Communication**

**Scénario : Suivi des livraisons**

* **Client demande des informations sur la livraison.**
* **Système de Suivi des Livraisons fournit l'état en temps réel.**
* **Livreur met à jour le statut de début et de fin de livraison.**
* **Système de Suivi des Livraisons enregistre et reflète ces mises à jour.**

**css**

**Copier le code**

**Client -> SystèmeSuiviLivraison : demanderSuivi()**

**SystèmeSuiviLivraison -> Client : fournirStatutActuel()**

**Livreur -> SystèmeSuiviLivraison : mettreÀJourStatutDébut()**

**SystèmeSuiviLivraison -> SystèmeSuiviLivraison : enregistrerStatutDébut()**

**Livreur -> SystèmeSuiviLivraison : mettreÀJourStatutFin()**

**SystèmeSuiviLivraison -> SystèmeSuiviLivraison : enregistrerStatutFin()**

**7. Diagramme de Composants**

**Composants :**

* **Interface Client (permet au client de suivre l'état de la livraison)**
* **Interface Livreur (permet au livreur de mettre à jour le statut de la livraison)**
* **Service de Suivi des Livraisons (gère la mise à jour et l'affichage des statuts)**
* **Base de Données des Livraisons (stocke les informations sur les livraisons et leurs statuts)**

**Relations :**

* **Interface Client et Interface Livreur communiquent avec le Service de Suivi des Livraisons.**
* **Service de Suivi des Livraisons accède à la Base de Données des Livraisons pour récupérer et mettre à jour les informations de livraison.**

**diff**

**Copier le code**

**+---------------------+**

**| Interface Client |**

**+---------------------+**

**|**

**v**

**+-------------------------------+**

**| Service de Suivi des Livraisons |**

**+-------------------------------+**

**|**

**v**

**+-----------------------------+**

**| Base de Données des Livraisons |**

**+-----------------------------+**

**+---------------------+**

**| Interface Livreur |**

**+---------------------+**

**|**

**v**

**+-------------------------------+**

**| Service de Suivi des Livraisons |**

**+-------------------------------+**

**8. Diagramme de Déploiement**

**Nœuds :**

* **Serveur d'Application (héberge le Service de Suivi des Livraisons et les Interfaces Client et Livreur)**
* **Base de Données des Livraisons (stocke les informations sur les livraisons et leurs statuts)**

**Relations :**

* **Le Serveur d'Application se connecte à la Base de Données des Livraisons pour récupérer et mettre à jour les informations nécessaires.**

**diff**

**Copier le code**

**+---------------------+**

**| Serveur d'Application |**

**+---------------------+**

**|**

**v**

**+-----------------------------+**

**| Base de Données des Livraisons |**

**+-----------------------------+**

1. **Messagerie et Notification :**
2. **Notification des Utilisateurs :**

**1. Diagramme de Cas d'Utilisation**

**Acteurs :**

* **Utilisateur (Client, Manager, Employé, Livreur)**
* **Système de Notification**

**Cas d'utilisation :**

* **Recevoir des notifications pour des événements importants**
* **Envoyer des notifications par e-mail**
* **Envoyer des notifications par SMS**
* **Envoyer des notifications via l'application**

**Relations :**

**lua**

**Copier le code**

**[Utilisateur] --> (Recevoir des notifications pour des événements importants)**

**[Système de Notification] --> (Envoyer des notifications par e-mail)**

**[Système de Notification] --> (Envoyer des notifications par SMS)**

**[Système de Notification] --> (Envoyer des notifications via l'application)**

**2. Diagramme de Classes**

**Classes :**

* **Utilisateur**
  + **Attributs : id, nom, email, numéroTéléphone, préférencesNotification**
  + **Méthodes : recevoirNotification()**
* **Notification**
  + **Attributs : id, type, message, date**
  + **Méthodes : créerNotification(), envoyer()**
* **SystèmeNotification**
  + **Attributs : listeNotifications**
  + **Méthodes : envoyerNotificationEmail(), envoyerNotificationSMS(), envoyerNotificationApp()**

**Diagramme de Classes :**

**lua**

**Copier le code**

**+------------------+**

**| Utilisateur |**

**+------------------+**

**| - id |**

**| - nom |**

**| - email |**

**| - numéroTéléphone|**

**| - préférencesNotification |**

**+------------------+**

**| + recevoirNotification() |**

**+------------------+**

**+------------------+**

**| Notification |**

**+------------------+**

**| - id |**

**| - type |**

**| - message |**

**| - date |**

**+------------------+**

**| + créerNotification() |**

**| + envoyer() |**

**+------------------+**

**+-------------------------------+**

**| SystèmeNotification |**

**+-------------------------------+**

**| - listeNotifications |**

**+-------------------------------+**

**| + envoyerNotificationEmail() |**

**| + envoyerNotificationSMS() |**

**| + envoyerNotificationApp() |**

**+-------------------------------+**

**3. Diagramme de Séquence**

**Scénario : Envoi de notifications**

1. **Système de Notification détecte un événement important (création de commande, mise à jour du statut de commande, livraison).**
2. **Système de Notification crée une Notification.**
3. **Système de Notification envoie la notification par e-mail, SMS, et/ou via l'application en fonction des préférences de l'Utilisateur.**
4. **Utilisateur reçoit la notification.**

**rust**

**Copier le code**

**SystèmeNotification -> Notification : créerNotification()**

**Notification -> SystèmeNotification : notification créée**

**SystèmeNotification -> SystèmeNotification : envoyerNotificationEmail()**

**SystèmeNotification -> Utilisateur : notificationEmail()**

**SystèmeNotification -> SystèmeNotification : envoyerNotificationSMS()**

**SystèmeNotification -> Utilisateur : notificationSMS()**

**SystèmeNotification -> SystèmeNotification : envoyerNotificationApp()**

**SystèmeNotification -> Utilisateur : notificationApp()**

**4. Diagramme d'Activités**

**Activité : Envoi de notifications**

1. **Début**
2. **Détecter un événement important (création de commande, mise à jour du statut, livraison)**
3. **Créer une notification**
4. **Vérifier les préférences de l'utilisateur**
5. **Envoyer la notification par e-mail si l'utilisateur a choisi cette option**
6. **Envoyer la notification par SMS si l'utilisateur a choisi cette option**
7. **Envoyer la notification via l'application si l'utilisateur a choisi cette option**
8. **Terminer**

**csharp**

**Copier le code**

**[Début]**

**|**

**[Détecter un événement important]**

**|**

**[Créer une notification]**

**|**

**[Vérifier les préférences de l'utilisateur]**

**|**

**[Envoyer la notification par e-mail]**

**|**

**[Envoyer la notification par SMS]**

**|**

**[Envoyer la notification via l'application]**

**|**

**[Terminer]**

**5. Diagramme d'États**

**État : Notification**

* **État Initial : Notification non créée**
* **État : Notification créée**
* **État : Notification envoyée par e-mail**
* **État : Notification envoyée par SMS**
* **État : Notification envoyée via l'application**
* **État Final : Notification reçue par l'utilisateur**

**css**

**Copier le code**

**[Notification non créée] --> [Notification créée]**

**[Notification créée] --> [Notification envoyée par e-mail]**

**[Notification créée] --> [Notification envoyée par SMS]**

**[Notification créée] --> [Notification envoyée via l'application]**

**[Notification envoyée par e-mail] --> [Notification reçue par l'utilisateur]**

**[Notification envoyée par SMS] --> [Notification reçue par l'utilisateur]**

**[Notification envoyée via l'application] --> [Notification reçue par l'utilisateur]**

**6. Diagramme de Communication**

**Scénario : Envoi de notifications**

* **Système de Notification crée une notification.**
* **Système de Notification envoie la notification par différents moyens (e-mail, SMS, application).**
* **Utilisateur reçoit la notification.**

**css**

**Copier le code**

**SystèmeNotification -> Notification : créerNotification()**

**SystèmeNotification -> Utilisateur : notificationEmail()**

**SystèmeNotification -> Utilisateur : notificationSMS()**

**SystèmeNotification -> Utilisateur : notificationApp()**

**7. Diagramme de Composants**

**Composants :**

* **Interface Utilisateur (affiche les notifications à l'utilisateur)**
* **Service de Notification (gère la création et l'envoi des notifications)**
* **Serveur de Messagerie (envoie les notifications par e-mail)**
* **Serveur SMS (envoie les notifications par SMS)**
* **Serveur d'Application (envoie les notifications via l'application)**

**Relations :**

* **Service de Notification communique avec les serveurs de messagerie, SMS, et d'application pour envoyer les notifications.**
* **Interface Utilisateur reçoit et affiche les notifications envoyées.**

**diff**

**Copier le code**

**+---------------------------+**

**| Interface Utilisateur |**

**+---------------------------+**

**|**

**v**

**+---------------------------+**

**| Service de Notification |**

**+---------------------------+**

**|**

**+----+-----+----+----+**

**| | | | |**

**v v v v v**

**+--------+ +--------+ +--------+**

**| Serveur | | Serveur | | Serveur |**

**| de | | SMS | | d'App |**

**| Messagerie | | | | |**

**+--------+ +--------+ +--------+**

**8. Diagramme de Déploiement**

**Nœuds :**

* **Serveur d'Application (héberge le Service de Notification et les Interfaces Utilisateur)**
* **Serveur de Messagerie (envoie des e-mails)**
* **Serveur SMS (envoie des SMS)**
* **Serveur d'Application Mobile (envoie des notifications via l'application)**

**Relations :**

* **Serveur d'Application se connecte aux serveurs de messagerie, SMS, et d'application mobile pour envoyer les notifications.**
* **Serveur d'Application est accessible par l'Interface Utilisateur pour afficher les notifications.**

**diff**

**Copier le code**

**+---------------------------+**

**| Serveur d'Application |**

**+---------------------------+**

**|**

**+-------+-------+-------+**

**| | | |**

**v v v v**

**+--------+ +--------+ +--------+**

**| Serveur | | Serveur | | Serveur |**

**| de | | SMS | | d'App |**

**| Messagerie | | | | |**

**+--------+ +--------+ +--------+**

1. **Messages Internes :**

**1. Diagramme de Cas d'Utilisation**

**Acteurs :**

* **Utilisateur (Client, Manager, Employé, Livreur)**
* **Système de Messagerie**

**Cas d'utilisation :**

* **Envoyer des messages internes**
* **Conserver les messages dans l'historique**
* **Consulter l'historique des messages**

**Relations :**

**scss**

**Copier le code**

**[Utilisateur] --> (Envoyer des messages internes)**

**[Utilisateur] --> (Consulter l'historique des messages)**

**[Système de Messagerie] --> (Conserver les messages dans l'historique)**

**2. Diagramme de Classes**

**Classes :**

* **Utilisateur**
  + **Attributs : id, nom, email, rôle**
  + **Méthodes : envoyerMessage(), consulterHistoriqueMessages()**
* **Message**
  + **Attributs : id, contenu, dateHeure, expéditeur, destinataire**
  + **Méthodes : créerMessage()**
* **HistoriqueMessages**
  + **Attributs : listeMessages**
  + **Méthodes : ajouterMessage(), obtenirMessages()**
* **SystèmeMessagerie**
  + **Attributs : historiqueMessages**
  + **Méthodes : envoyerMessage(), conserverMessage()**

**Diagramme de Classes :**

**lua**

**Copier le code**

**+------------------+**

**| Utilisateur |**

**+------------------+**

**| - id |**

**| - nom |**

**| - email |**

**| - rôle |**

**+------------------+**

**| + envoyerMessage() |**

**| + consulterHistoriqueMessages() |**

**+------------------+**

**+------------------+**

**| Message |**

**+------------------+**

**| - id |**

**| - contenu |**

**| - dateHeure |**

**| - expéditeur |**

**| - destinataire |**

**+------------------+**

**| + créerMessage() |**

**+------------------+**

**+-----------------------------+**

**| HistoriqueMessages |**

**+-----------------------------+**

**| - listeMessages |**

**+-----------------------------+**

**| + ajouterMessage() |**

**| + obtenirMessages() |**

**+-----------------------------+**

**+-------------------------------+**

**| SystèmeMessagerie |**

**+-------------------------------+**

**| - historiqueMessages |**

**+-------------------------------+**

**| + envoyerMessage() |**

**| + conserverMessage() |**

**+-------------------------------+**

**3. Diagramme de Séquence**

**Scénario : Envoi et consultation de messages**

1. **Utilisateur compose un message.**
2. **Système de Messagerie crée un Message.**
3. **Système de Messagerie conserve le message dans l'HistoriqueMessages.**
4. **Utilisateur consulte l'historique des messages.**
5. **Système de Messagerie récupère les messages depuis l'HistoriqueMessages.**
6. **Système de Messagerie fournit les messages à l'Utilisateur.**

**css**

**Copier le code**

**Utilisateur -> SystèmeMessagerie : envoyerMessage()**

**SystèmeMessagerie -> Message : créerMessage()**

**SystèmeMessagerie -> HistoriqueMessages : ajouterMessage()**

**Utilisateur -> SystèmeMessagerie : consulterHistoriqueMessages()**

**SystèmeMessagerie -> HistoriqueMessages : obtenirMessages()**

**SystèmeMessagerie -> Utilisateur : fournirMessages()**

**4. Diagramme d'Activités**

**Activité : Envoi et gestion des messages**

1. **Début**
2. **Utilisateur compose un message**
3. **Système de Messagerie crée le message**
4. **Système de Messagerie conserve le message dans l'historique**
5. **Utilisateur consulte l'historique des messages**
6. **Système de Messagerie récupère les messages de l'historique**
7. **Système de Messagerie affiche les messages à l'utilisateur**
8. **Terminer**

**csharp**

**Copier le code**

**[Début]**

**|**

**[Utilisateur compose un message]**

**|**

**[Système de Messagerie crée le message]**

**|**

**[Système de Messagerie conserve le message dans l'historique]**

**|**

**[Utilisateur consulte l'historique des messages]**

**|**

**[Système de Messagerie récupère les messages de l'historique]**

**|**

**[Système de Messagerie affiche les messages à l'utilisateur]**

**|**

**[Terminer]**

**5. Diagramme d'États**

**État : Message**

* **État Initial : Message non créé**
* **État : Message créé**
* **État : Message conservé dans l'historique**
* **État Final : Message consulté par l'utilisateur**

**mathematica**

**Copier le code**

**[Message non créé] --> [Message créé]**

**[Message créé] --> [Message conservé dans l'historique]**

**[Message conservé dans l'historique] --> [Message consulté par l'utilisateur]**

**6. Diagramme de Communication**

**Scénario : Envoi et consultation de messages**

* **Utilisateur envoie un message via le Système de Messagerie.**
* **Système de Messagerie crée et conserve le message dans l'HistoriqueMessages.**
* **Utilisateur consulte l'historique des messages.**
* **Système de Messagerie fournit les messages de l'historique à l'Utilisateur.**

**css**

**Copier le code**

**Utilisateur -> SystèmeMessagerie : envoyerMessage()**

**SystèmeMessagerie -> HistoriqueMessages : ajouterMessage()**

**Utilisateur -> SystèmeMessagerie : consulterHistoriqueMessages()**

**SystèmeMessagerie -> HistoriqueMessages : obtenirMessages()**

**SystèmeMessagerie -> Utilisateur : fournirMessages()**

**7. Diagramme de Composants**

**Composants :**

* **Interface Utilisateur (permet d'envoyer des messages et de consulter l'historique)**
* **Service de Messagerie (gère la création, l'envoi, et la conservation des messages)**
* **Base de Données des Messages (stocke les messages)**

**Relations :**

* **Interface Utilisateur communique avec le Service de Messagerie pour envoyer des messages et consulter l'historique.**
* **Service de Messagerie accède à la Base de Données des Messages pour stocker et récupérer les messages.**

**diff**

**Copier le code**

**+-----------------------+**

**| Interface Utilisateur |**

**+-----------------------+**

**|**

**v**

**+--------------------------+**

**| Service de Messagerie |**

**+--------------------------+**

**|**

**v**

**+-----------------------------+**

**| Base de Données des Messages|**

**+-----------------------------+**

**8. Diagramme de Déploiement**

**Nœuds :**

* **Serveur d'Application (héberge le Service de Messagerie et les Interfaces Utilisateur)**
* **Base de Données des Messages (stocke les messages et leur historique)**

**Relations :**

* **Serveur d'Application se connecte à la Base de Données des Messages pour stocker et récupérer les messages.**

**diff**

**Copier le code**

**+-----------------------+**

**| Serveur d'Application |**

**+-----------------------+**

**|**

**v**

**+-----------------------------+**

**| Base de Données des Messages|**

**+-----------------------------+**

1. **Gestion pour les avis avec Emojis :**
2. **Création des avis :**

**1. Diagramme de Cas d'Utilisation**

**Acteurs :**

* **Utilisateur (Client, Manager, Employé, Livreur)**
* **Système d'Avis**

**Cas d'utilisation :**

* **Laisser un avis**
* **Vérifier si l'utilisateur est connecté**
* **Vérifier qu'un seul avis par commande est autorisé**
* **Inclure un emoji de satisfaction et un commentaire optionnel**

**Relations :**

**rust**

**Copier le code**

**[Utilisateur] --> (Laisser un avis)**

**[Système d'Avis] --> (Vérifier si l'utilisateur est connecté)**

**[Système d'Avis] --> (Vérifier qu'un seul avis par commande est autorisé)**

**[Système d'Avis] --> (Inclure un emoji de satisfaction et un commentaire optionnel)**

**2. Diagramme de Classes**

**Classes :**

* **Utilisateur**
  + **Attributs : id, nom, email, connecté**
  + **Méthodes : laisserAvis()**
* **Avis**
  + **Attributs : id, emojiSatisfaction, commentaire, utilisateur, commande**
  + **Méthodes : créerAvis()**
* **Commande**
  + **Attributs : id, statut**
  + **Méthodes : vérifierAvis()**
* **SystèmeAvis**
  + **Attributs : listeAvis**
  + **Méthodes : vérifierConnexion(), vérifierAvisUnique(), enregistrerAvis()**

**Diagramme de Classes :**

**sql**

**Copier le code**

**+------------------+**

**| Utilisateur |**

**+------------------+**

**| - id |**

**| - nom |**

**| - email |**

**| - connecté |**

**+------------------+**

**| + laisserAvis() |**

**+------------------+**

**+------------------+**

**| Avis |**

**+------------------+**

**| - id |**

**| - emojiSatisfaction |**

**| - commentaire |**

**| - utilisateur |**

**| - commande |**

**+------------------+**

**| + créerAvis() |**

**+------------------+**

**+------------------+**

**| Commande |**

**+------------------+**

**| - id |**

**| - statut |**

**+------------------+**

**| + vérifierAvis() |**

**+------------------+**

**+-------------------------+**

**| SystèmeAvis |**

**+-------------------------+**

**| - listeAvis |**

**+-------------------------+**

**| + vérifierConnexion() |**

**| + vérifierAvisUnique() |**

**| + enregistrerAvis() |**

**+-------------------------+**

**3. Diagramme de Séquence**

**Scénario : Création d'un avis**

1. **Utilisateur se connecte à l'application.**
2. **Système d'Avis vérifie la connexion de l'utilisateur.**
3. **Utilisateur laisse un avis pour une commande.**
4. **Système d'Avis vérifie que l'utilisateur n'a pas déjà laissé un avis pour cette commande.**
5. **Système d'Avis crée un Avis avec un emoji de satisfaction et un commentaire optionnel.**
6. **Système d'Avis enregistre l'avis.**

**css**

**Copier le code**

**Utilisateur -> SystèmeAvis : se connecter()**

**SystèmeAvis -> Utilisateur : vérifierConnexion()**

**Utilisateur -> SystèmeAvis : laisserAvis(commande, emojiSatisfaction, commentaire)**

**SystèmeAvis -> Commande : vérifierAvis()**

**SystèmeAvis -> Avis : créerAvis()**

**SystèmeAvis -> SystèmeAvis : enregistrerAvis()**

**4. Diagramme d'Activités**

**Activité : Création d'un avis**

1. **Début**
2. **Vérifier si l'utilisateur est connecté**
3. **Utilisateur choisit une commande pour laisser un avis**
4. **Vérifier qu'un avis n'a pas encore été laissé pour cette commande**
5. **Utilisateur fournit un emoji de satisfaction et un commentaire optionnel**
6. **Créer l'avis**
7. **Enregistrer l'avis**
8. **Terminer**

**csharp**

**Copier le code**

**[Début]**

**|**

**[Verifier si l'utilisateur est connecté]**

**|**

**[Utilisateur choisit une commande]**

**|**

**[Vérifier qu'un avis n'a pas encore été laissé pour cette commande]**

**|**

**[Utilisateur fournit un emoji de satisfaction et un commentaire optionnel]**

**|**

**[Créer l'avis]**

**|**

**[Enregistrer l'avis]**

**|**

**[Terminer]**

**5. Diagramme d'États**

**État : Avis**

* **État Initial : Avis non créé**
* **État : Avis créé avec emoji et commentaire**
* **État Final : Avis enregistré**

**css**

**Copier le code**

**[Avis non créé] --> [Avis créé avec emoji et commentaire]**

**[Avis créé avec emoji et commentaire] --> [Avis enregistré]**

**6. Diagramme de Communication**

**Scénario : Création d'un avis**

* **Utilisateur envoie une demande de création d'avis au Système d'Avis.**
* **Système d'Avis vérifie la connexion et la validité de l'avis.**
* **Système d'Avis crée et enregistre l'avis.**

**css**

**Copier le code**

**Utilisateur -> SystèmeAvis : laisserAvis(commande, emojiSatisfaction, commentaire)**

**SystèmeAvis -> Commande : vérifierAvis()**

**SystèmeAvis -> Avis : créerAvis()**

**SystèmeAvis -> SystèmeAvis : enregistrerAvis()**

**7. Diagramme de Composants**

**Composants :**

* **Interface Utilisateur (permet à l'utilisateur de laisser un avis)**
* **Service d'Avis (gère la création et l'enregistrement des avis)**
* **Base de Données des Avis (stocke les avis)**

**Relations :**

* **Interface Utilisateur communique avec le Service d'Avis pour créer et gérer les avis.**
* **Service d'Avis accède à la Base de Données des Avis pour stocker les avis.**

**diff**

**Copier le code**

**+-----------------------+**

**| Interface Utilisateur |**

**+-----------------------+**

**|**

**v**

**+--------------------+**

**| Service d'Avis |**

**+--------------------+**

**|**

**v**

**+------------------------------+**

**| Base de Données des Avis |**

**+------------------------------+**

**8. Diagramme de Déploiement**

**Nœuds :**

* **Serveur d'Application (héberge le Service d'Avis et l'Interface Utilisateur)**
* **Base de Données des Avis (stocke les avis et leurs informations)**

**Relations :**

* **Serveur d'Application se connecte à la Base de Données des Avis pour stocker et récupérer les avis.**

**diff**

**Copier le code**

**+-----------------------+**

**| Serveur d'Application |**

**+-----------------------+**

**|**

**v**

**+------------------------------+**

**| Base de Données des Avis |**

**+------------------------------+**

1. **Moderation d’avis :**

**1. Diagramme de Cas d'Utilisation**

**Acteurs :**

* **Utilisateur (Client, Manager, Employé, Livreur)**
* **Système de Modération des Avis**
* **Manager**

**Cas d'utilisation :**

* **Signaler un avis**
* **Modérer un avis**
* **Revoir les avis modérés**
* **Prendre une décision sur un avis (approbation ou suppression)**

**Relations :**

**scss**

**Copier le code**

**[Utilisateur] --> (Signaler un avis)**

**[Système de Modération des Avis] --> (Modérer un avis)**

**[Manager] --> (Revoir les avis modérés)**

**[Manager] --> (Prendre une décision sur un avis)**

**2. Diagramme de Classes**

**Classes :**

* **Utilisateur**
  + **Attributs : id, nom, email, rôle**
  + **Méthodes : signalerAvis()**
* **Avis**
  + **Attributs : id, contenu, utilisateur, commande, signalé, modéré**
  + **Méthodes : signaler(), modérer()**
* **Manager**
  + **Attributs : id, nom, email**
  + **Méthodes : revoirAvis(), déciderAvis()**
* **SystèmeModérationAvis**
  + **Attributs : listeAvisModérés**
  + **Méthodes : recevoirSignalement(), attribuerModération(), enregistrerDécision()**

**Diagramme de Classes :**

**lua**

**Copier le code**

**+------------------+**

**| Utilisateur |**

**+------------------+**

**| - id |**

**| - nom |**

**| - email |**

**| - rôle |**

**+------------------+**

**| + signalerAvis() |**

**+------------------+**

**+------------------+**

**| Avis |**

**+------------------+**

**| - id |**

**| - contenu |**

**| - utilisateur |**

**| - commande |**

**| - signalé |**

**| - modéré |**

**+------------------+**

**| + signaler() |**

**| + modérer() |**

**+------------------+**

**+------------------+**

**| Manager |**

**+------------------+**

**| - id |**

**| - nom |**

**| - email |**

**+------------------+**

**| + revoirAvis() |**

**| + déciderAvis() |**

**+------------------+**

**+------------------------------+**

**| SystèmeModérationAvis |**

**+------------------------------+**

**| - listeAvisModérés |**

**+------------------------------+**

**| + recevoirSignalement() |**

**| + attribuerModération() |**

**| + enregistrerDécision() |**

**+------------------------------+**

**3. Diagramme de Séquence**

**Scénario : Modération d'un avis**

1. **Utilisateur signale un avis.**
2. **Système de Modération des Avis reçoit le signalement.**
3. **Système de Modération des Avis attribue l'avis à un Manager pour révision.**
4. **Manager révise l'avis modéré.**
5. **Manager prend une décision (approbation ou suppression).**
6. **Système de Modération des Avis enregistre la décision.**

**css**

**Copier le code**

**Utilisateur -> SystèmeModérationAvis : signalerAvis(avis)**

**SystèmeModérationAvis -> Manager : attribuerModération(avis)**

**Manager -> SystèmeModérationAvis : revoirAvis()**

**Manager -> SystèmeModérationAvis : déciderAvis(décision)**

**SystèmeModérationAvis -> SystèmeModérationAvis : enregistrerDécision(décision)**

**4. Diagramme d'Activités**

**Activité : Modération d'un avis**

1. **Début**
2. **Utilisateur signale un avis**
3. **Système de Modération des Avis reçoit le signalement**
4. **Système de Modération des Avis attribue l'avis à un manager**
5. **Manager révise l'avis modéré**
6. **Manager prend une décision (approbation ou suppression)**
7. **Système de Modération des Avis enregistre la décision**
8. **Terminer**

**csharp**

**Copier le code**

**[Début]**

**|**

**[Utilisateur signale un avis]**

**|**

**[Système de Modération des Avis reçoit le signalement]**

**|**

**[Système de Modération des Avis attribue l'avis à un manager]**

**|**

**[Manager révise l'avis modéré]**

**|**

**[Manager prend une décision (approbation ou suppression)]**

**|**

**[Système de Modération des Avis enregistre la décision]**

**|**

**[Terminer]**

**5. Diagramme d'États**

**État : Avis modéré**

* **État Initial : Avis non signalé**
* **État : Avis signalé**
* **État : Avis attribué à un manager**
* **État : Avis révisé**
* **État Final : Avis approuvé ou supprimé**

**css**

**Copier le code**

**[Avis non signalé] --> [Avis signalé]**

**[Avis signalé] --> [Avis attribué à un manager]**

**[Avis attribué à un manager] --> [Avis révisé]**

**[Avis révisé] --> [Avis approuvé ou supprimé]**

**6. Diagramme de Communication**

**Scénario : Modération d'un avis**

* **Utilisateur signale un avis au Système de Modération des Avis.**
* **Système de Modération des Avis attribue l'avis à un Manager.**
* **Manager révise et prend une décision sur l'avis.**
* **Système de Modération des Avis enregistre la décision.**

**css**

**Copier le code**

**Utilisateur -> SystèmeModérationAvis : signalerAvis(avis)**

**SystèmeModérationAvis -> Manager : attribuerModération(avis)**

**Manager -> SystèmeModérationAvis : revoirAvis()**

**Manager -> SystèmeModérationAvis : déciderAvis(décision)**

**SystèmeModérationAvis -> SystèmeModérationAvis : enregistrerDécision(décision)**

**7. Diagramme de Composants**

**Composants :**

* **Interface Utilisateur (permet aux utilisateurs de signaler des avis)**
* **Service de Modération des Avis (gère la réception, l'attribution, et la décision sur les avis signalés)**
* **Base de Données des Avis (stocke les avis et les décisions de modération)**

**Relations :**

* **Interface Utilisateur communique avec le Service de Modération des Avis pour signaler les avis.**
* **Service de Modération des Avis accède à la Base de Données des Avis pour stocker et récupérer les avis modérés.**

**diff**

**Copier le code**

**+-----------------------+**

**| Interface Utilisateur |**

**+-----------------------+**

**|**

**v**

**+--------------------------+**

**| Service de Modération des Avis |**

**+--------------------------+**

**|**

**v**

**+------------------------------+**

**| Base de Données des Avis |**

**+------------------------------+**

**8. Diagramme de Déploiement**

**Nœuds :**

* **Serveur d'Application (héberge le Service de Modération des Avis et l'Interface Utilisateur)**
* **Base de Données des Avis (stocke les avis, les signalements et les décisions)**

**Relations :**

* **Serveur d'Application se connecte à la Base de Données des Avis pour stocker et récupérer les avis et les décisions de modération.**

**diff**

**Copier le code**

**+-----------------------+**

**| Serveur d'Application |**

**+-----------------------+**

**|**

**v**

**+------------------------------+**

**| Base de Données des Avis |**

**+------------------------------+**

1. **Consultation d’avis :**

**1. Diagramme de Cas d'Utilisation**

**Acteurs :**

* Utilisateur (Client, Manager, Employé, Livreur)
* Système de Consultation des Avis

**Cas d'utilisation :**

* Consulter les avis
* Trier les avis par date
* Trier les avis par emoji
* Trier les avis par pertinence

**Relations :**

scss

Copier le code

[Utilisateur] --> (Consulter les avis)

[Système de Consultation des Avis] --> (Trier les avis par date)

[Système de Consultation des Avis] --> (Trier les avis par emoji)

[Système de Consultation des Avis] --> (Trier les avis par pertinence)

**2. Diagramme de Classes**

**Classes :**

* **Utilisateur**
  + Attributs : id, nom, email
  + Méthodes : consulterAvis()
* **Avis**
  + Attributs : id, contenu, emojiSatisfaction, date, pertinence, statut
  + Méthodes : afficherAvis()
* **SystèmeConsultationAvis**
  + Attributs : listeAvisApprouvés
  + Méthodes : trierAvisParDate(), trierAvisParEmoji(), trierAvisParPertinence(), consulterAvis()

**Diagramme de Classes :**

lua

Copier le code

+------------------+

| Utilisateur |

+------------------+

| - id |

| - nom |

| - email |

+------------------+

| + consulterAvis()|

+------------------+

+------------------+

| Avis |

+------------------+

| - id |

| - contenu |

| - emojiSatisfaction |

| - date |

| - pertinence |

| - statut |

+------------------+

| + afficherAvis() |

+------------------+

+-------------------------------+

| SystèmeConsultationAvis |

+-------------------------------+

| - listeAvisApprouvés |

+-------------------------------+

| + trierAvisParDate() |

| + trierAvisParEmoji() |

| + trierAvisParPertinence() |

| + consulterAvis() |

+-------------------------------+

**3. Diagramme de Séquence**

**Scénario : Consultation des avis**

1. **Utilisateur** demande la consultation des avis.
2. **Système de Consultation des Avis** récupère les avis approuvés.
3. **Utilisateur** demande le tri des avis par date, emoji, ou pertinence.
4. **Système de Consultation des Avis** applique le tri demandé.
5. **Système de Consultation des Avis** affiche les avis triés à l'**Utilisateur**.

css

Copier le code

Utilisateur -> SystèmeConsultationAvis : consulterAvis()

SystèmeConsultationAvis -> Base de Données : récupérerAvisApprouvés()

Utilisateur -> SystèmeConsultationAvis : trierAvisParDate()

SystèmeConsultationAvis -> Utilisateur : afficherAvis(triés)

**4. Diagramme d'Activités**

**Activité : Consultation des avis**

1. **Début**
2. **Utilisateur consulte les avis**
3. **Système de Consultation des Avis récupère les avis approuvés**
4. **Utilisateur choisit le critère de tri (date, emoji, pertinence)**
5. **Système de Consultation des Avis trie les avis selon le critère choisi**
6. **Afficher les avis triés**
7. **Terminer**

csharp

Copier le code

[Début]

|

[Utilisateur consulte les avis]

|

[Système de Consultation des Avis récupère les avis approuvés]

|

[Utilisateur choisit le critère de tri]

|

[Système de Consultation des Avis trie les avis selon le critère]

|

[Afficher les avis triés]

|

[Terminer]

**5. Diagramme d'États**

**État : Avis consulté**

* **État Initial** : Avis non consulté
* **État** : Avis récupérés
* **État** : Avis triés par date
* **État** : Avis triés par emoji
* **État** : Avis triés par pertinence
* **État Final** : Avis affichés

css

Copier le code

[Avis non consulté] --> [Avis récupérés]

[Avis récupérés] --> [Avis triés par date]

[Avis triés par date] --> [Avis affichés]

[Avis récupérés] --> [Avis triés par emoji]

[Avis triés par emoji] --> [Avis affichés]

[Avis récupérés] --> [Avis triés par pertinence]

[Avis triés par pertinence] --> [Avis affichés]

**6. Diagramme de Communication**

**Scénario : Consultation des avis**

* **Utilisateur** demande à consulter les avis.
* **Système de Consultation des Avis** récupère les avis approuvés.
* **Utilisateur** choisit le critère de tri.
* **Système de Consultation des Avis** trie les avis et les affiche à l'**Utilisateur**.

css

Copier le code

Utilisateur -> SystèmeConsultationAvis : consulterAvis()

SystèmeConsultationAvis -> Base de Données : récupérerAvisApprouvés()

Utilisateur -> SystèmeConsultationAvis : trierAvisParDate()

SystèmeConsultationAvis -> Utilisateur : afficherAvis(triés)

**7. Diagramme de Composants**

**Composants :**

* **Interface Utilisateur** (permet aux utilisateurs de consulter et trier les avis)
* **Service de Consultation des Avis** (gère la récupération et le tri des avis)
* **Base de Données des Avis** (stocke les avis approuvés)

**Relations :**

* **Interface Utilisateur** communique avec le **Service de Consultation des Avis** pour afficher et trier les avis.
* **Service de Consultation des Avis** accède à la **Base de Données des Avis** pour récupérer les avis approuvés.

diff

Copier le code

+-----------------------+

| Interface Utilisateur |

+-----------------------+

|

v

+-------------------------------+

| Service de Consultation des Avis |

+-------------------------------+

|

v

+------------------------------+

| Base de Données des Avis |

+------------------------------+

**8. Diagramme de Déploiement**

**Nœuds :**

* **Serveur d'Application** (héberge le Service de Consultation des Avis et l'Interface Utilisateur)
* **Base de Données des Avis** (stocke les avis approuvés)

**Relations :**

* **Serveur d'Application** se connecte à la **Base de Données des Avis** pour récupérer les avis approuvés.

diff

Copier le code

+-----------------------+

| Serveur d'Application |

+-----------------------+

|

v

+------------------------------+

| Base de Données des Avis |

+------------------------------+