Une image contenant texte, diagramme, Plan, Dessin technique

Le contenu généré par l’IA peut être incorrect.

Une image contenant texte, diagramme, Plan, Dessin technique

Le contenu généré par l’IA peut être incorrect.

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

Le contenu généré par l’IA peut être incorrect.

"Cas d’utilisation sécurisés"

Une image contenant texte, diagramme, ligne, Parallèle

Le contenu généré par l’IA peut être incorrect.

"Cas d’utilisation publics"

Une image contenant texte, diagramme, ligne, cercle

Le contenu généré par l’IA peut être incorrect.

**2. Diagramme de flux (Activity Diagram)**

**2.1 Flux “Effectuer un trade (achat/vente)”**

**2.1.1 Description**

Ce diagramme illustre le processus complet lorsqu’un utilisateur réalise un trade :

1. **Début**
2. L’utilisateur choisit “Acheter” ou “Vendre”
3. Sélection de la cryptomonnaie + saisie de la quantité
4. **Vérifier solde disponible**
   * Si fonds insuffisants (pour achat) ou holdings insuffisants (pour vente) → affichage d’un message d’erreur → **Fin**
   * Sinon → suite
5. Saisie du prix unitaire
6. **Calculer frais**
7. Confirmation utilisateur
   * Si refus → affichage “Trade annulé” → **Fin**
   * Si validation → suite
8. **Mettre à jour/inserer un holding**
   * Si achat : mise à jour ou création de la ligne Wallet\_Holding (quantité + average\_price)
   * Si vente : diminution de la quantité, suppression si quantité = 0
9. Enregistrement du Trade
10. Affichage du récapitulatif
11. **Fin**

**2.1.2 Code PlantUML**

plantuml

CopierModifier

@startuml

|Utilisateur|

start

:Choisir "Acheter" ou "Vendre";

:Sélectionner crypto\_symbol\net saisir quantité;

if (Type = Achat ?) then (oui)

:Vérifier solde disponible;

if (Fonds suffisants ?) then (non)

:Afficher "Fonds insuffisants";

stop

endif

else (vente)

:Vérifier holdings disponibles;

if (Quantité insuffisante ?) then (non)

:Afficher "Quantité insuffisante";

stop

endif

endif

:Demander prix unitaire;

:Calculer frais;

if (Utilisateur confirme ?) then (non)

:Afficher "Trade annulé";

stop

endif

:Mettre à jour/inserer Wallet\_Holding;

" - si achat : incrémenter quantity,\n recalculer average\_price\n - si vente : décrémenter quantity,\n supprimer si = 0" as note1

note right of Mettre\_à\_jour/inserer: note1

:Créer enregistrement Trade;

:Afficher récapitulatif du trade;

stop

@enduml

**3. Diagramme d’entités (Class Diagram)**

**3.1 Description**

On part du schéma physique (Relations) :

* **User → Wallet : One-to-One**
  + En pratique, 1 utilisateur → N portefeuilles (vous avez noté 1..1–0..1 en Merise, mais l’implémentation est 1 utilisateur = 1 portefeuille obligatoire)
* **Wallet → Wallet\_Holding : One-to-Many**
* **Wallet\_Holding → Trade : One-to-Many**
* **User → Preference : One-to-Many**
* **User ↔ Learn : Many-to-Many** via **User\_Learn** (PK composite : user\_id + learn\_id, attributs is\_completed, completed\_at)
* **Price : entité autonome** (sans FK en V1)

**3.2 Attributs (tirés du dictionnaire de données & SQL)**

* **User**
  + user\_id : INT (PK)
  + username : VARCHAR
  + email : VARCHAR
  + password\_hash : VARCHAR
  + created\_at : DATETIME
  + last\_login : DATETIME
* **Wallet**
  + wallet\_id : INT (PK)
  + user\_id : INT (FK → User.user\_id, NOT NULL)
  + created\_at : DATETIME
  + initial\_balance : DECIMAL(15,2)
* **Wallet\_Holding**
  + holding\_id : INT (PK)
  + wallet\_id : INT (FK → Wallet.wallet\_id, NOT NULL)
  + crypto\_symbol : VARCHAR
  + quantity : DECIMAL(18,8)
  + average\_price : DECIMAL(15,2)
* **Trade**
  + trade\_id : INT (PK)
  + holding\_id : INT (FK → Wallet\_Holding.holding\_id, NOT NULL)
  + crypto\_symbol : VARCHAR
  + type : ENUM('buy','sell')
  + amount : DECIMAL(18,8)
  + price : DECIMAL(15,2)
  + fee : DECIMAL(15,2)
  + timestamp : DATETIME
* **Preference**
  + preference\_id : INT (PK)
  + user\_id : INT (FK → User.user\_id, NOT NULL)
  + pref\_key : VARCHAR
  + pref\_value : VARCHAR
* **Learn**
  + learn\_id : INT (PK)
  + title : VARCHAR
  + content : TEXT
  + order\_index : INT
* **User\_Learn** (association M–N)
  + user\_id : INT (PK partiel, FK → User.user\_id)
  + learn\_id : INT (PK partiel, FK → Learn.learn\_id)
  + is\_completed : BOOLEAN
  + completed\_at : DATETIME
* **Price**
  + price\_id : INT (PK)
  + crypto\_symbol : VARCHAR
  + value : DECIMAL(15,2)
  + recorded\_at : DATETIME

**3.3 Code PlantUML**

plantuml

CopierModifier

@startuml

' Définition des classes avec attributs

class User {

- user\_id : INT

- username : VARCHAR

- email : VARCHAR

- password\_hash : VARCHAR

- created\_at : DATETIME

- last\_login : DATETIME

}

class Wallet {

- wallet\_id : INT

- user\_id : INT

- created\_at : DATETIME

- initial\_balance : DECIMAL(15,2)

}

class Wallet\_Holding {

- holding\_id : INT

- wallet\_id : INT

- crypto\_symbol : VARCHAR

- quantity : DECIMAL(18,8)

- average\_price : DECIMAL(15,2)

}

class Trade {

- trade\_id : INT

- holding\_id : INT

- crypto\_symbol : VARCHAR

- type : ENUM { buy, sell }

- amount : DECIMAL(18,8)

- price : DECIMAL(15,2)

- fee : DECIMAL(15,2)

- timestamp : DATETIME

}

class Preference {

- preference\_id : INT

- user\_id : INT

- pref\_key : VARCHAR

- pref\_value : VARCHAR

}

class Learn {

- learn\_id : INT

- title : VARCHAR

- content : TEXT

- order\_index : INT

}

class User\_Learn {

- user\_id : INT

- learn\_id : INT

- is\_completed : BOOLEAN

- completed\_at : DATETIME

}

class Price {

- price\_id : INT

- crypto\_symbol : VARCHAR

- value : DECIMAL(15,2)

- recorded\_at : DATETIME

}

' Associations avec cardinalités

User "1" -- "1" Wallet : possède >

Wallet "1" -- "0..\*" Wallet\_Holding : contient >

Wallet\_Holding "1" -- "0..\*" Trade : génère >

User "1" -- "0..\*" Preference : a >

User "0..\*" -- "0..\*" Learn : suit >

User\_Learn \*-- "1" User :

User\_Learn \*-- "1" Learn :

' Note : Price est autonome, pas de lien direct

@enduml

**Légende des multiplicités (UML)** :

* User "1" -- "1" Wallet signifie qu’un User est lié à exactement un Wallet (One-to-One).
* Wallet "1" -- "0..\*" Wallet\_Holding signifie qu’un Wallet peut avoir zéro à plusieurs Wallet\_Holding (One-to-Many).
* Wallet\_Holding "1" -- "0..\*" Trade signifie qu’un Wallet\_Holding peut avoir zéro à plusieurs Trade.
* User "1" -- "0..\*" Preference montre qu’un User peut avoir zéro à plusieurs Preference.
* La relation User "0..\*" -- "0..\*" Learn est matérialisée par la classe d’association User\_Learn (Many-to-Many).

Diagramme activité plantuml

@startuml

start

:Choisir "Acheter" ou "Vendre";

:Sélectionner crypto\_symbol et saisir quantité;

if (Type = "Achat" ?) then (oui)

:Vérifier solde disponible;

if (Fonds suffisants ?) then (non)

:Afficher "Fonds insuffisants";

stop

endif

else (vente)

:Vérifier holdings disponibles;

if (Quantité insuffisante ?) then (oui)

:Afficher "Quantité insuffisante";

stop

endif

endif

:Demander prix unitaire;

:Calculer frais;

if (Utilisateur confirme ?) then (non)

:Afficher "Trade annulé";

stop

endif

:Mettre à jour/inserer Wallet\_Holding;

note right

- si achat : incrémenter quantity, recalculer average\_price

- si vente : décrémenter quantity, supprimer si = 0

end note

:Créer enregistrement Trade;

:Afficher récapitulatif du trade;

stop

@enduml

Une image contenant capture d’écran

Le contenu généré par l’IA peut être incorrect.