**BMO**

* **Diagramme des cas d’utilisation**

UseCaseDiagram [frame=true framecolor=steelblue label="Usecase Diagram"] {

actor a as "Bookmakeur"

actor a2 as "Tiers"

actor a1 as "Parieur"

system s as "Gestion des paris sportifs" [fillcolor=whitesmoke]{

usecase u as "Gérer les paris\net évènements"

usecase u2 as "Consulter les \névènements"

usecase u7 as "Consulter les résultats"

usecase u4 as "Placer pari"

usecase u5 as "Se connecter"

usecase u6 as "Faire mise à jour ou \najouter des données"

usecase u8 as "Gérer compte\n utilisateur"

}

a -- u

a -- u5

a2 -- u6

a2 -- u5

a1 -- u2

a1 -- u4

a1 -- u5

a1 -- u7

a1 -- u8

}

* **Diagramme de classe**

**Règles de gestion :**

ClassDiagram [frame=true framecolor=steelblue label="Class Diagram"] {

abstract class Parieur {

private id : int

private email : string

private nom: string

private prenom: string

private password : string

private jetons: long

private capitalInitial: int

private mise: long

public login(email: string, password:string)

public parier(montant: int, evenement: String, typePari: int, resultatPari: String): boolean

public validerMontant(montant: long)

public consulterSolde()

public rechargerCompte(montant: long)

}

class TypeSport {

private id: int

private nom : string

private description : string

}

class Football {

private id : int

private regle: string

}

class CourseCheveau {

private id : int

private regle: string

}

class Tennis {

private id : int

private regle: string

}

class BasketBall {

private id : int

private regle: string

}

class Pari {

private id : int

private Idparieur: int

private Idevenement: int

private IdtypePari: int

private montant: long

private cotes: string

private resultatPari: String

}

class EvenementSportif {

private id : int

private dateDebut: date

private dateFin: date

private cotes: string

private description: string

}

class Simple {

private id : int

}

class Avance {

private id : int

}

class Bookmaker {

private id : int

private email : string

private nom: string

private prenom: string

private password : string

public proposerEvenement(): void

public definirParametres(): void

public miseAJourCote(evenement, typePari, resultatPari)

}

class Tiers {

private id : int

public ajouterDonnees()

private email : string

private nom: string

private prenom: string

private password : string

public mettreAJourDonnees()

}

Football -g-> TypeSport

BasketBall -g-> TypeSport

CourseCheveau -g-> TypeSport

Tennis -g-> TypeSport

Avance -g-> Pari

Simple -g-> Pari

Pari "0..\*" -- "1..1" Parieur : "Placer"

Pari "1..\*" -- "1..1" TypeSport : "Concerne"

Bookmaker "1..1" -- "0..\*" EvenementSportif : "Gère"

Tiers "1..1" -- "0..\*" EvenementSportif : "Mettre à jour"

EvenementSportif "0..\*" -- "1..\*" Pari : "Concerne"

}

1. Un parieur peut placé sur plusieurs paries
2. Un parie est fait par plusieurs parieurs
3. Un parie concerne un évènement sportif
4. Un parie concerne un et un seul type de sport
5. Un évènement sportif comporte plusieurs paries
6. D’après les spécifications du sujet, on a 4 types de sports d’où la création les 4 classes spécifiques à la classe TypeSport
7. Un pari à deux spécificités : on fait soit un parie simple soit un parie avancer
8. Le bookmaker et le tiers ajoute et font des mises à jour sur les évènements sportifs

* **Diagrammes de séquence**

Se connecter

* **Placer Parie**

**Scénario nominal**

SequenceDiagram [frame=true framecolor=steelblue label="Sequence Diagram"] {

actor Parieur

lifeline ":Système de Pari" as sys

lifeline BDD

Parieur --> sys "connexion"

sys -r-> Parieur "envoies la page de connexion"

Parieur --> sys "login(Julienne,a1z5)"

activate BDD

sys --> BDD " vérification des informations de connexion"

BDD --> sys "connexion validée"

deactivate BDD

sys -r-> Parieur "affiche l'interface pour faire un pari"

Parieur --> sys " sélectionne un événement sportif"

Parieur --> sys "choisi un type de pari"

Parieur --> sys "choisi le ou les résultats du pari"

Parieur --> sys "choisi le montant du pari"

sys -r-> Parieur "envoies la page pour la vérification des informations"

Parieur --> sys "Confirmation des informations"

sys -r-> Parieur "Parie validé et envoies de la page pour faire un pari"

}

**Scénario alternatif**

1. Premier

SequenceDiagram [frame=true framecolor=steelblue label="Sequence Diagram"] {

actor Parieur

lifeline ":Système de Pari" as sys

lifeline BDD

Parieur --> sys "Demande la page pour placer un parie"

fragment alt "[Si le parieur n'est paas connecté]" {

sys -r-> Parieur "renvoies la page de connexion"

Parieur --> sys "login(Julienne,a1555)"

activate BDD

sys --> BDD " vérification des informations de connexion"

BDD --> sys "connexion validée"

sys -r-> Parieur "affiche l'interface pour faire un pari"

deactivate BDD

case "[Si le parieur était connecté]"

sys -r-> Parieur "affiche l'interface pour faire un pari"

deactivate BDD

}

Parieur --> sys " sélectionne un événement sportif"

Parieur --> sys "choisi un type de pari"

Parieur --> sys "choisi le ou les résultats du pari"

Parieur --> sys "choisi le montant du pari"

sys -r-> Parieur "envoies la page pour la vérification des informations"

fragment alt "[Si le parieur ne confirme pas les informations]" {

Parieur --> sys "Annule le Parie"

sys -r-> Parieur "Rien n'est enregistrer et envoies la page de parie"

case "[Si le parieur confirme le parie]"

Parieur --> sys "Confirmation des informations"

activate BDD

sys --> BDD "enregistre les informations dans la base de données"

BDD -r-> sys "confirmation d'enregistrement"

sys -r-> Parieur "Notification du Parie validé et envoies de la page pour faire un pari"

}

}

1. Deuxieme

SequenceDiagram [frame=true framecolor=steelblue label="Sequence Diagram"] {

actor Parieur

lifeline ":Système de Pari" as sys

lifeline BDD

Parieur --> sys "Demande la page pour placer un parie"

fragment alt "[Si le parieur n'est paas connecté]" {

sys -r-> Parieur "renvoies la page de connexion"

Parieur --> sys "login(Julienne,a1555)"

activate BDD

sys --> BDD " vérification des informations de connexion"

BDD --> sys "connexion validée"

sys -r-> Parieur "affiche l'interface pour faire un pari"

deactivate BDD

case "[Si le parieur était connecté]"

sys -r-> Parieur "affiche l'interface pour faire un pari"

deactivate BDD

}

Parieur --> sys " sélectionne un événement sportif"

Parieur --> sys "choisi un type de pari"

Parieur --> sys "choisi le ou les résultats du pari"

Parieur --> sys "choisi le montant du pari"

sys -r-> Parieur "envoies la page pour la vérification des informations"

fragment alt "[Si le parieur ne confirme pas les informations]" {

Parieur --> sys "Annule le Parie"

sys -r-> Parieur "Rien n'est enregistrer et envoies la page de parie"

case "[Si le parieur confirme le parie]"

Parieur --> sys "Confirmation des informations"

activate BDD

sys --> BDD "envoies les informations dans la base de données"

fragment alt "[Une erreur d'enregistrement]" {

BDD --> sys "N'enregistre rien et envoies une erreur"

sys -r-> Parieur "notification d'erreur, envoies la page de parie et demande au parieur de reprendre"

case "[Succès d'enregistrement]"

BDD -r-> BDD "enregistre les informations"

BDD -r-> sys "confirmation d'enregistrement"

sys -r-> Parieur "Notification du Parie validé et envoies de la page pour faire un pari"

}

sys -r-> Parieur "Notification du Parie validé et envoies de la page pour faire un pari"

}

}

3. En cas de perte de Parie

SequenceDiagram [frame=true framecolor=steelblue label="Sequence Diagram"] {

actor Parieur

lifeline ":Système" as sys

lifeline Parie

Parieur --> sys "Demande de connexion"

activate Parieur

activate sys

sys -r-> Parieur "renvoies la page de connexion"

Parieur --> sys "connecter(Jule,12X203)"

sys -r-> sys "vérification des informations de connexion"

sys -r-> Parieur "interface de parie"

Parieur --> sys " sélectionne l'événement sportif"

Parieur --> sys "choisi un type de pari"

Parieur --> sys "choisi le ou les résultats du parie"

Parieur --> sys "choisi le montant 5000 du parie"

sys -r-> Parieur "envoies la page pour confirmer le parie"

Parieur --> sys " Parie confirmé"

activate Parie

sys --> Parie "Création du parie"

Parie -r-> sys "pari perdu"

activate Parie

sys -r-> Parieur "perte(15000)"

deactivate Parie

deactivate sys

deactivate Parie

}

* Gérer les paries et les évènements

**Scénario nominal**

SequenceDiagram [frame=true framecolor=steelblue label="Sequence Diagram"] {

actor bookmaker

lifeline ":Système" as sys

lifeline BDD

bookmaker --> sys "Se connecte"

sys -r-> bookmaker "envoies la page de connexion"

bookmaker --> sys "envoies les données de connexion"

activate BDD

sys --> BDD "vérification des informations de connexion"

BDD -r-> sys "connexion réussi"

sys -r-> bookmaker "formulaire de mise à jour des données"

bookmaker --> sys "choisir un évènement sportif"

bookmaker --> sys "choisir un type de parie"

bookmaker --> sys "limiter(100000)"

bookmaker --> sys "plafonner (20000)"

sys -r-> bookmaker "vérification des informations saisies"

sys --> BDD "Envoies les informations pour mettre à jour la base de données"

BDD --> BDD "mettre à jour des données"

BDD -r-> sys "Mise à jour réussis"

sys --> bookmaker "Notification de réussite de mise à jour et envoies de la page initiale"

deactivate BDD

}

**Scénario alternatif**

SequenceDiagram [frame=true framecolor=steelblue label="Sequence Diagram"] {

actor bookmaker

lifeline ":Système" as sys

lifeline BDD

bookmaker --> sys "fais une demande de nouvelle connexion"

activate sys

sys -r-> bookmaker "envoies la page de connexion"

bookmaker --> sys "login(Admin,Admin2024)"

activate BDD

sys --> BDD "vérification des données de connexion"

fragment alt "[Données incorrectes]" {

BDD -r-> sys "information erronées"

sys -r-> bookmaker "Notification d'une erreurs de saisie"

bookmaker --> sys "login(Admin,boook0024)"

sys --> BDD "vérification des informations de connexion"

case "[no permission]"

BDD --> sys "vérification des informations de connexion"

sys -r-> bookmaker "interface de gestion des paris"

}

sys -r-> bookmaker "Notification des données erronées et envoies de la page de connexion"

bookmaker --> sys "login(Admin,boook0024)"

sys -r-> bookmaker "vérification des informations de connexion"

sys -r-> bookmaker "interface de gestion des paris"

bookmaker --> sys "ajout des évènements "

bookmaker --> sys "paramétrer(4,2.5)"

sys --> BDD "Update des donner dans la base"

deactivate BDD

}

Faire mise à jour des données

* **Scénario nominal**

SequenceDiagram [frame=true framecolor=steelblue label="Sequence Diagram"] {

actor tiers

lifeline ":Système" as sys

lifeline BDD

tiers --> sys "fais une demande de nouvelle connexion"

activate sys

sys -r-> tiers "envoies la page de connexion"

tiers --> sys "login(user,Admin2024)"

activate BDD

sys --> BDD "vérification des données de connexion"

BDD --> sys "Connexion réussie"

sys -r-> tiers "interface de mise à jour des évènements sportif"

tiers --> sys "modifierResultat(1,2)"

sys --> BDD "Update des donner dans la base"

BDD --> BDD "enregistrement"

BDD --> sys "Enregistrement réussis"

sys -r-> tiers "Envoies de message de succès de mise à jour et la page de départ"

deactivate BDD

}

* **Scénario alternatif**

SequenceDiagram [frame=true framecolor=steelblue label="Sequence Diagram"] {

actor tiers

lifeline ":Système" as sys

lifeline BDD

tiers --> sys "fais une demande de nouvelle connexion"

activate sys

sys -r-> tiers "envoies la page de connexion"

tiers --> sys "login(user,Admin2024)"

activate BDD

sys --> BDD "vérification des données de connexion"

fragment alt "[Données incorrectes]" {

BDD -r-> sys "information erronées"

sys -r-> tiers "Notification d'une erreurs de saisie"

tiers --> sys "login(User,boook0024)"

sys --> BDD "vérification des informations de connexion"

case "[Connexion réussie]"

BDD --> sys "vérification des informations de connexion"

sys -r-> tiers "interface de mise à jour des données"

}

sys -r-> tiers "interface de mise à jour des évènements sportif"

tiers --> sys "modifierResultat(1,2)"

sys --> BDD "Update des donner dans la base"

tiers --> sys "Envoies de message de succès de mise à jour"

deactivate BDD

}

Connexion

* **Scénario nominal**

SequenceDiagram [frame=true framecolor=steelblue label="Sequence Diagram"] {

actor Parieur

lifeline ":Système" as sys

Parieur --> sys "Demande la page d'inscription"

activate sys

sys -r-> Parieur "Envoies de la page d'inscription"

Parieur --> sys "inscription(Julienne, User,15X23)"

sys -r-> sys "vérifier informations"

sys -r-> Parieur "Inscription réussie et envoies la page d'acceuil"

deactivate sys

}

**Deuxième**

SequenceDiagram [frame=true framecolor=steelblue label="Sequence Diagram"] {

actor Parieur

lifeline ":Système" as sys

lifeline "Jule:Parieur" as Jule

Parieur --> sys "inscription(ZZZ, Jule,user1213)"

activate sys

sys -r-> sys "vérifier informations d'inscription"

sys -c-> Jule "create"

Jule -r-> sys "Création réussis"

activate Jule

sys -r-> Parieur "Notification de réussite d'inscription et crédit de 1000 jetons"

deactivate sys

}

**Consulter un evènement**

SequenceDiagram [frame=true framecolor=steelblue label="Sequence Diagram"] {

actor Parieur

lifeline "Système" as sys

Parieur --> sys "Demande la page des évènements sportifs"

activate sys

Parieur <-r- sys "Envoies la page des evènements sportifs"

}

**Diagramme d’état transition**

**Classe Parieur**

//Les différents états d’un parieur

StateDiagram [frame=true framecolor=steelblue label="State Diagram" splines=ortho] {

state Inscrit

state Connecter

state Déconnecter

initialState->Inscrit

Inscrit -> Connecter "Success"

Connecter -> Déconnecter "Connexion"

Connecter <- Déconnecter "Déconnexion"

Déconnecter -> finalstate

}

**Classe Parie**

StateDiagram {

state Créer

initialstate -> Créer

state Gérer

state Placer

state Perdu

state Gagner

choice c

choice c1

Créer -> c

c -> Gérer "[user = bookmaker]"

c -> Placer "[user = Parieur]"

Placer -> c1

c1 -> Gagner "[user.resultat = parie.resultat]"

c1 -> Perdu "[user.resultat != parie.resultat]"

Perdu -> finalstate

Gagner -> finalstate

}

**Type Parie**

StateDiagram {

state Actif

state Inactif

initialstate -> Actif

Actif -> Inactif "Existe"

Inactif -> Actif "Inexistant"

Inactif -> finalstate

Actif -> finalstate

}

**Evènement**

L'événement commence dans l'état "NonCommence", où il attend soit le début prévu, soit une éventuelle annulation.

Lorsque l'événement commence, il passe à l'état "EnCours", où il attend soit la fin normale de l'événement, soit une annulation.

Une fois l'événement terminé, il passe à l'état "Termine", où les résultats peuvent être annoncés.

Si l'événement est annulé à tout moment, il passe à l'état "Annule".

Ces états et transitions représentent les différents scénarios auxquels un événement sportif peut être confronté dans le système. Il peut évoluer à partir de l'état initial "NonCommence", passer par l'état "EnCours" et enfin atteindre l'état final "Termine" une fois que les résultats sont annoncés. Alternativement, il peut être annulé à tout moment.

StateDiagram {

state Commencer

state AttenteDebut

state Termine

state EnCours

state Annule

state EnAttente

initialstate -> AttenteDebut "En attente de début"

AttenteDebut -> Commencer "Début de l'événement"

Commencer -> EnCours "Déroulement de l'événement"

EnCours -> Termine "Fin normale de l'événement"

EnCours -> Annule "Annulation de l'événement"

Annule -> finalstate

Termine -> finalstate

}