istic Informatique Électronique	**Cyril KONE** largaton-ange-cyri.kone@etudiant.univ-rennes1.fr **Salma BOUCHRA** salma.bouchra@etudiant.univ-rennes1.fr

DIAGRAMME DE CAS D'UTILISATION

Le diagramme de cas d'utilisation capture le comportement d'un système, d'un sous-système, d'une classe ou d'un composant tel qu'un utilisateur extérieur le voit. Il scinde la fonctionnalité du système en unités cohérentes, les cas d'utilisation, ayant un sens pour les acteurs. Autrement dit, le diagramme de cas d'utilisation représente la structure des grandes fonctionnalités nécessaires aux utilisateurs du système.

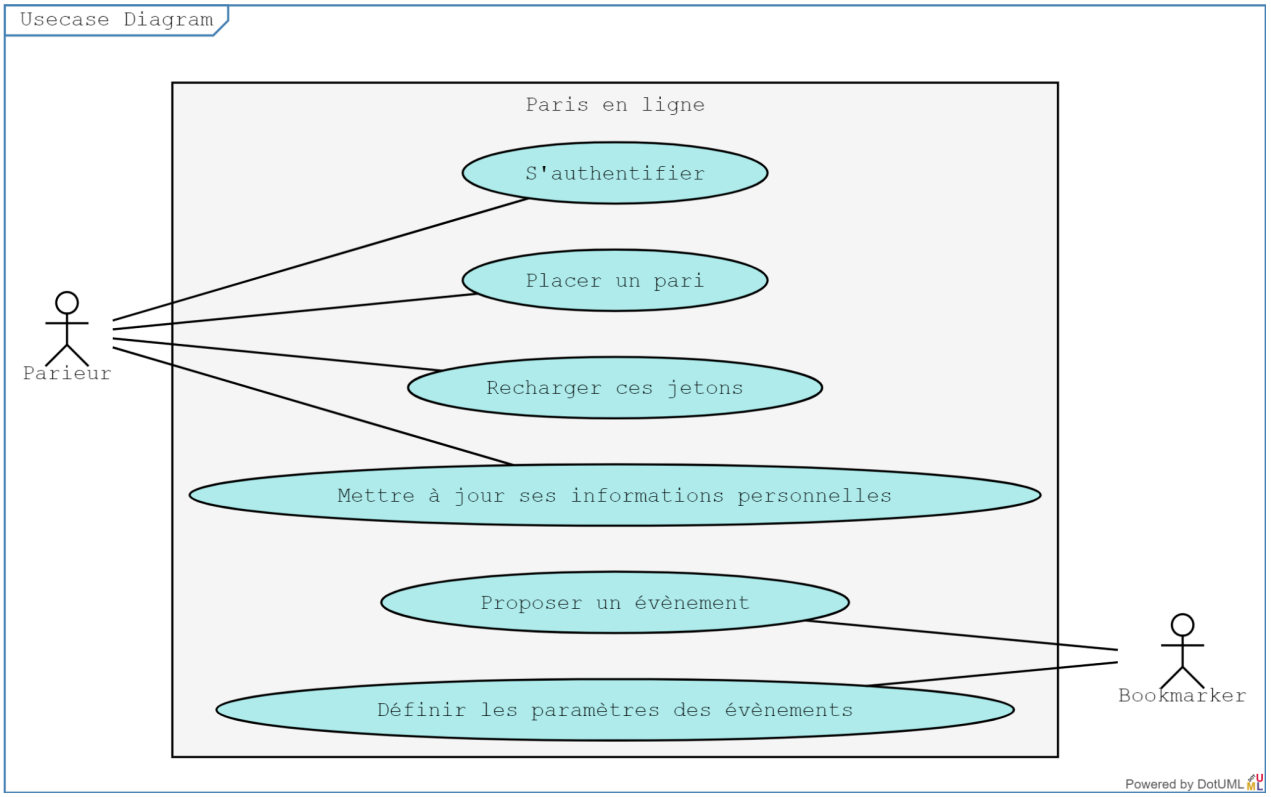


DIAGRAMME DE CLASSES

Le diagramme de classes montre la structure interne du système. Il permet de fournir une représentation abstraite des objets du système qui vont interagir ensemble pour réaliser les cas d'utilisation. Il faut noter qu'un même objet peut très bien intervenir dans la réalisation de plusieurs cas d'utilisation. Le diagramme de classe constitue l'un des pivots de la modélisation avec UML.

Les principaux éléments du diagramme de classes sont les classes, l'association, la composition, la dépendance, l'agrégation, la généralisation.

Classe : Les classes sont les modules de base de la programmation orientée objet. Une classe est représentée en utilisant un rectangle divisé en trois sections. La section supérieure est le nom de la classe. La section centrale définit les propriétés de la classe. La section du bas énumère les méthodes de la classe.

Association : Une association est une relation entre deux ou plusieurs classes. Elle est modélisée par une ligne reliant les classes. Cette ligne peut être qualifiée avec le type de relation, et peut également comporter des règles de multiplicité (par exemple un à un, un à plusieurs, plusieurs à plusieurs) pour la relation.

Composition : Si une classe ne peut pas exister par elle-même, mais doit être un membre d'une autre classe, alors elle possède une relation de composition avec la classe contenante. Une relation de composition est indiquée par une ligne avec un losange plein.

Dépendance : Quand une classe utilise une autre classe, par exemple comme membre ou comme paramètre d'une de ces fonctions, elle "dépend" ainsi de cette classe. Une relation de dépendance est représentée par une flèche pointillée.

Généralisation : Une relation de généralisation est l'équivalent d'une relation d'héritage en terme orienté objet. Une relation de généralisation est indiquée par une flèche creuse se dirigeant vers la classe "parent ".

Ci-dessous, le diagramme de classe de notre système

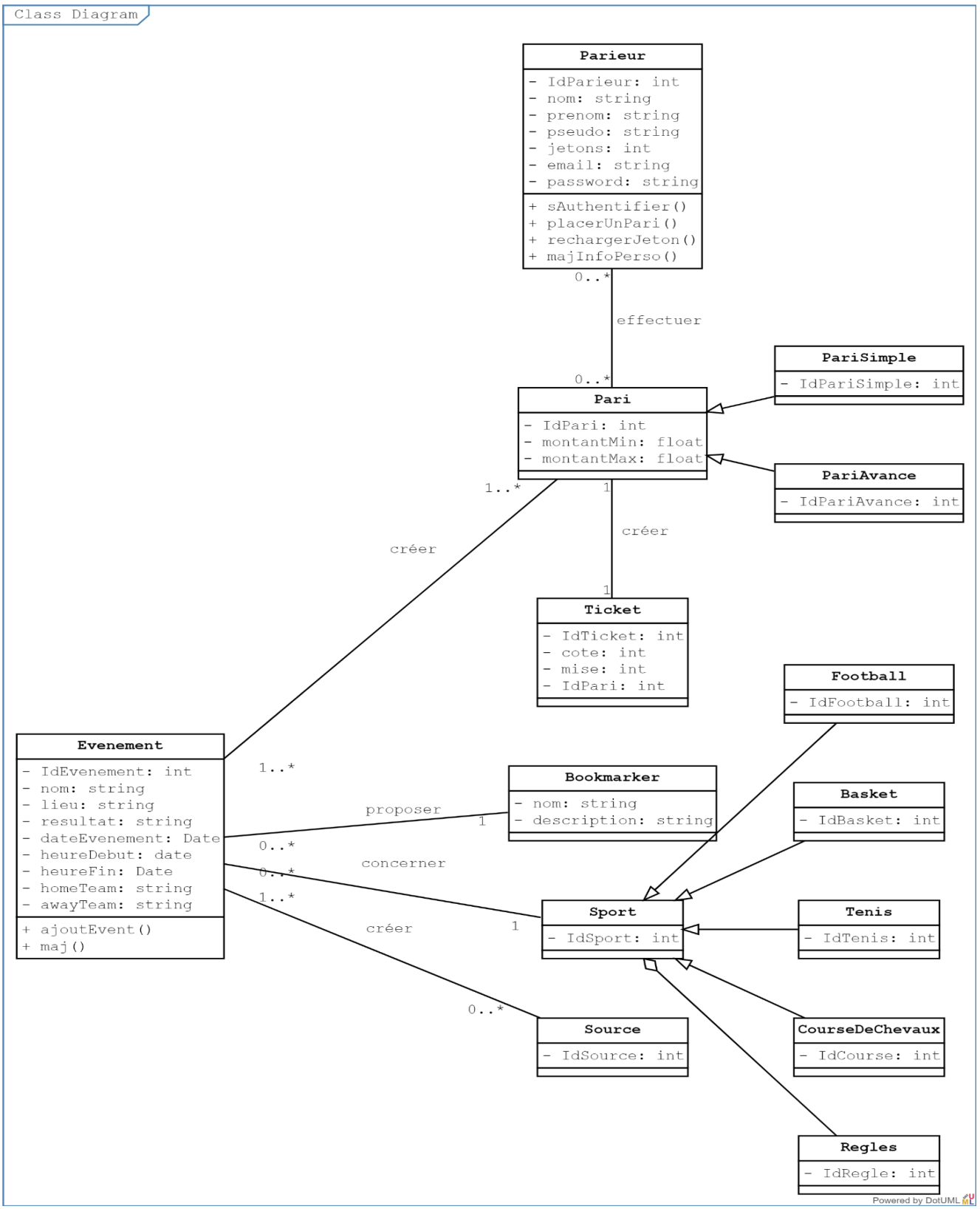
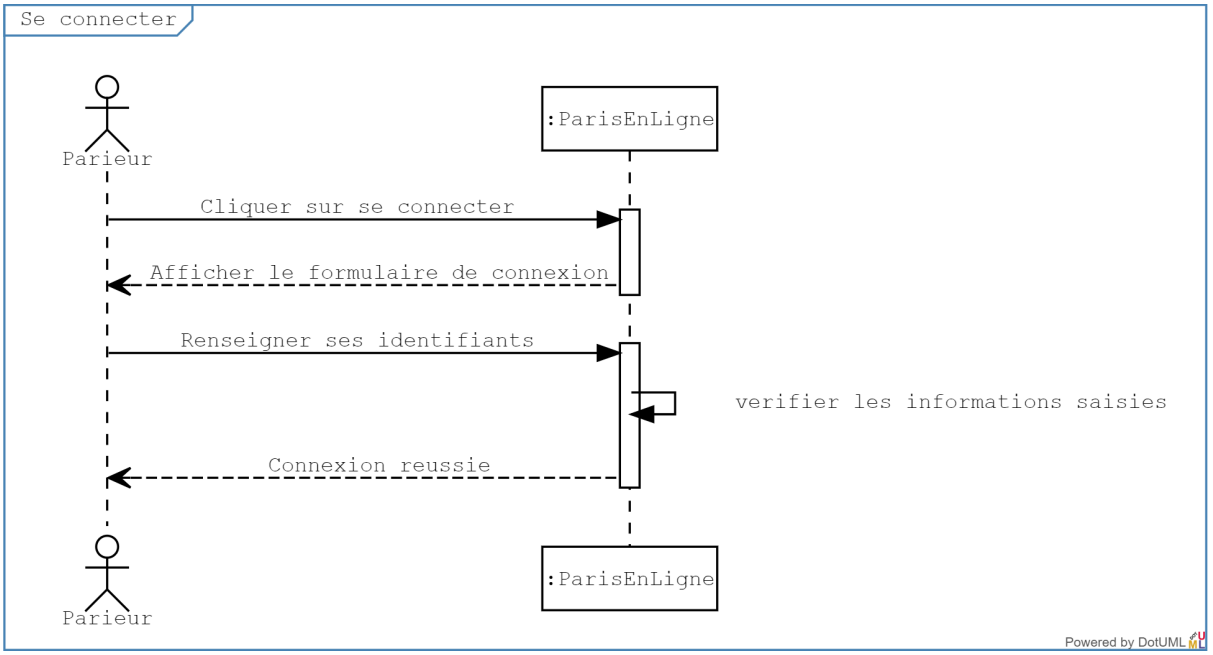

Class Diagram
Parieur
- IdParieur: int
- nom: string
- prenom: string
- pseudo: string
- jetons: int
- email: string
- password: string
+ sAuthentifier()
+ placerUnPari()
+ rechargerJeton()
+ majInfoPerso()
0..*
effectuer
0..*
PariSimple
- IdPariSimple: int
Pari
- IdPari: int
- montantMin: float
- montantMax: float
PariAvance
- IdPariAvance: int
1..*
1
créer
créer
1
Ticket
- IdTicket: int
- cote: int
- mise: int
- IdPari: int
Football
- IdFootball: int
Evenement
- IdEvenement: int
- nom: string
- lieu: string
- resultat: string
- dateEvenement: Date
- heureDebut: date
- heureFin: Date
- homeTeam: string
- awayTeam: string
+ ajoutEvent()
+ maj()
1..*
Bookmarker
- nom: string
- description: string
proposer
1
0..*
Basket
- IdBasket: int
concerner
0..*
1..*
créer
1
Sport
- IdSport: int
Tenis
- IdTenis: int
0..*
Source
- IdSource: int
CourseDeChevaux
- IdCourse: int
Regles
- IdRegle: int
Powered by DotUML

DIAGRAMME DE SÉQUENCES

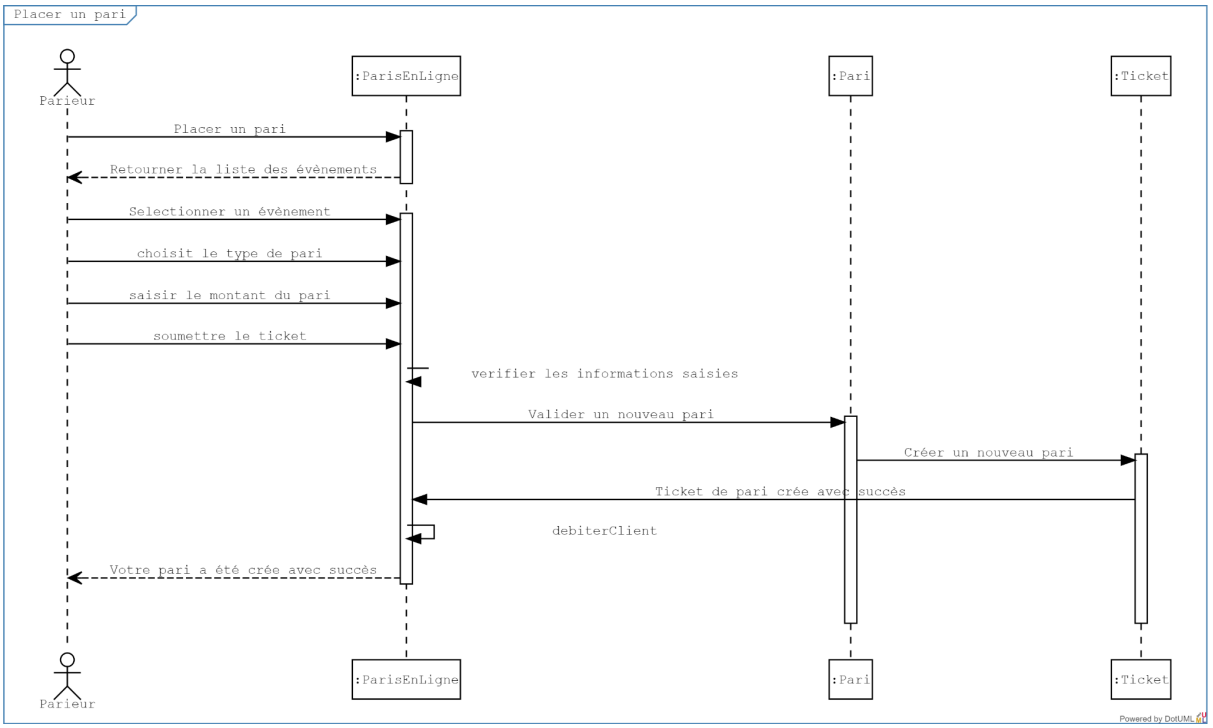
Le diagramme de séquence présente les messages échangés entre les lignes de vie dans un ordre chronologique. Un diagramme de séquence sert aussi à illustrer un cas d'utilisation en représentant les événements et les messages échangés entre le système et un acteur. Les diagrammes suivants présentent le cas nominal.

Cas 1: S'authentifier

Cette opération permet à l'utilisateur de vérifier son identité et de se connecter

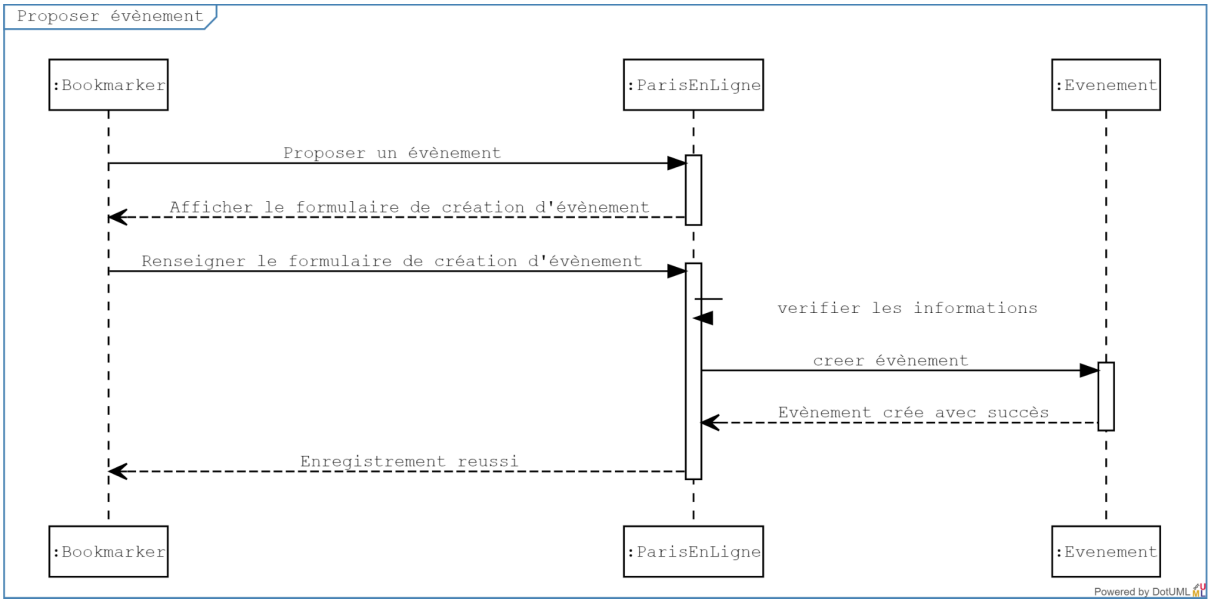


Cas 2: Placer un pari



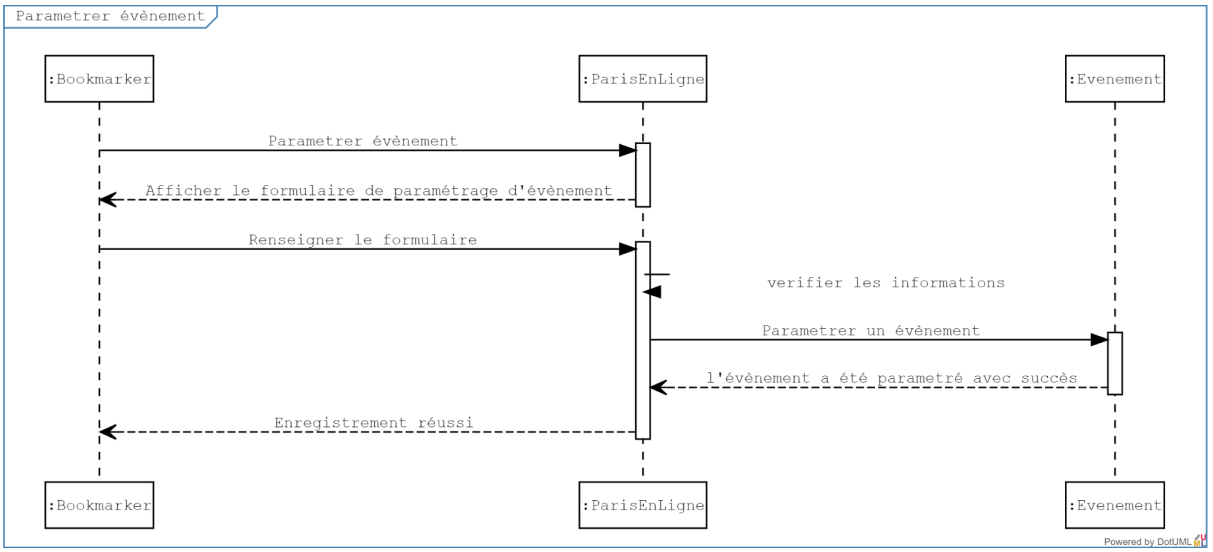
Cas 3: Proposer un événement

Cette opération permet au bookmaker de proposer un évènement



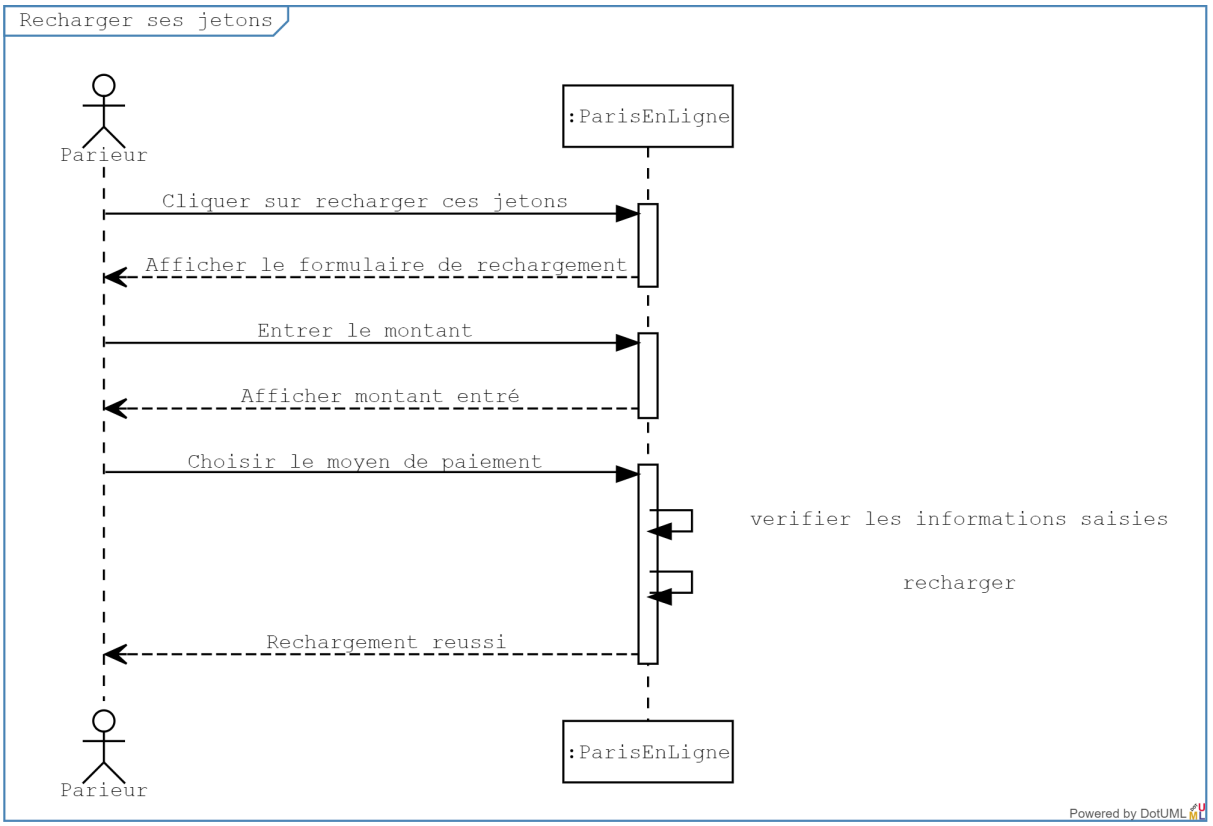
Cas 4: Paramétrer un événement

Cette opération permet au bookmaker de paramétrer un évènement



Cas 5:Recharger ses jetons

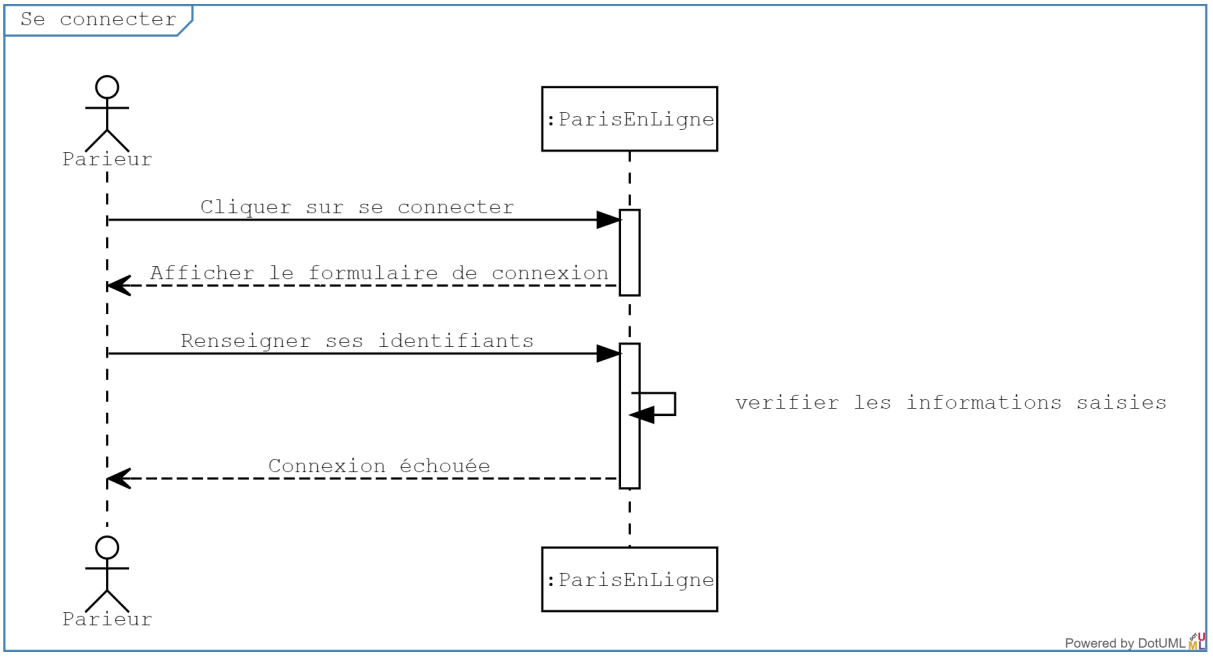
Cette opération permet au parieur de recharger ses jetons.



Les diagrammes suivants présentent les cas exceptionnels.

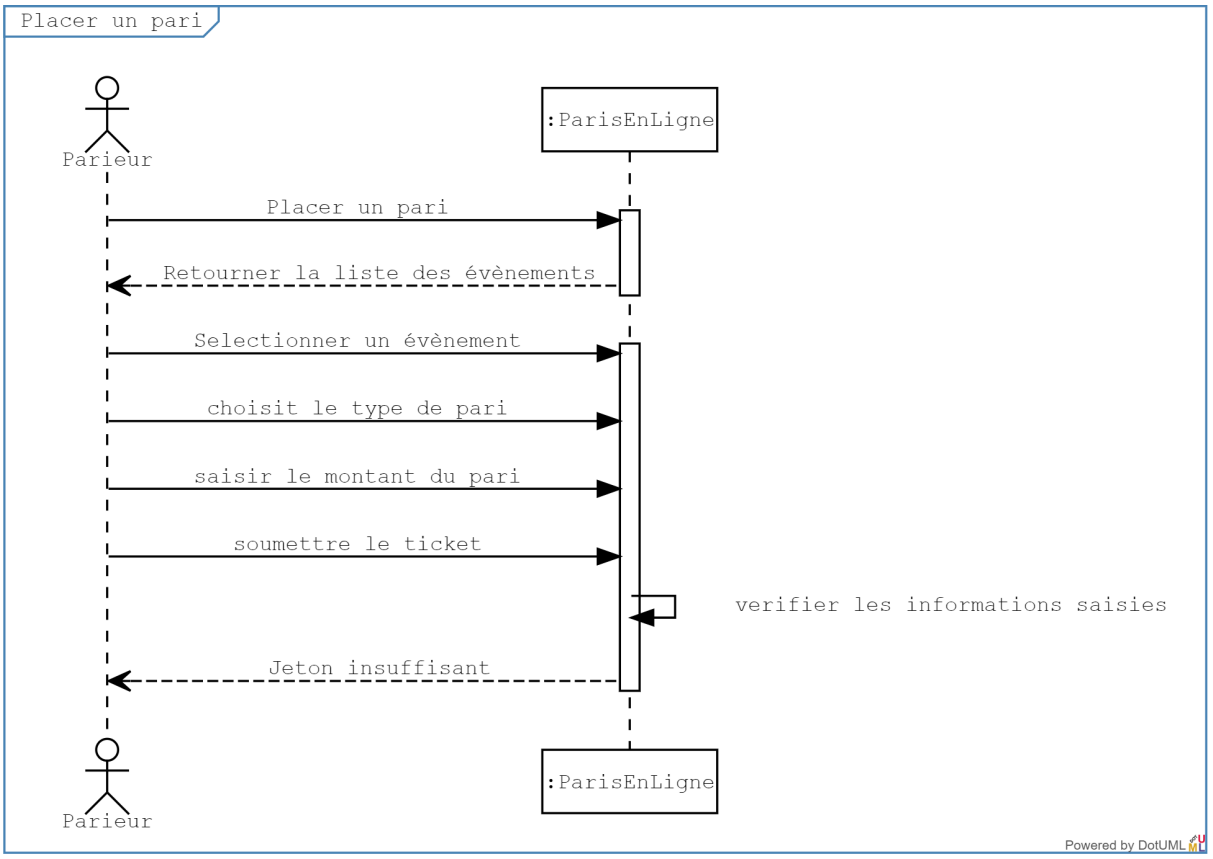
Cas 1: S’authentifier

Cette opération permet à l’utilisateur de vérifier son identité et de se connecter: échec



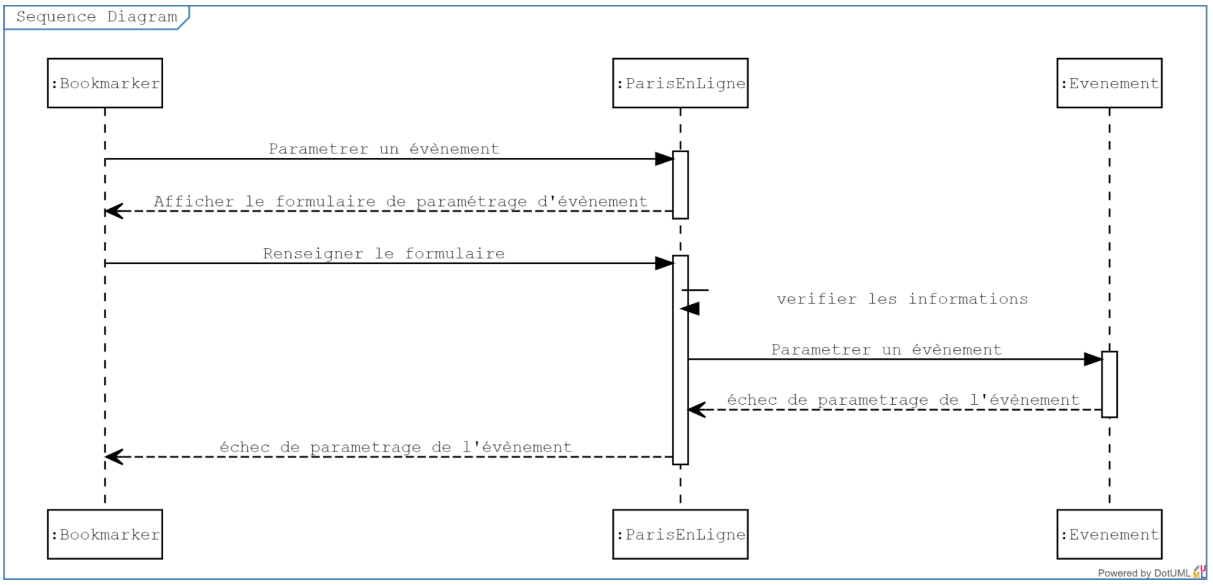
Cas 2: Placer un pari

Cette opération permet à l'utilisateur d'effectuer un pari



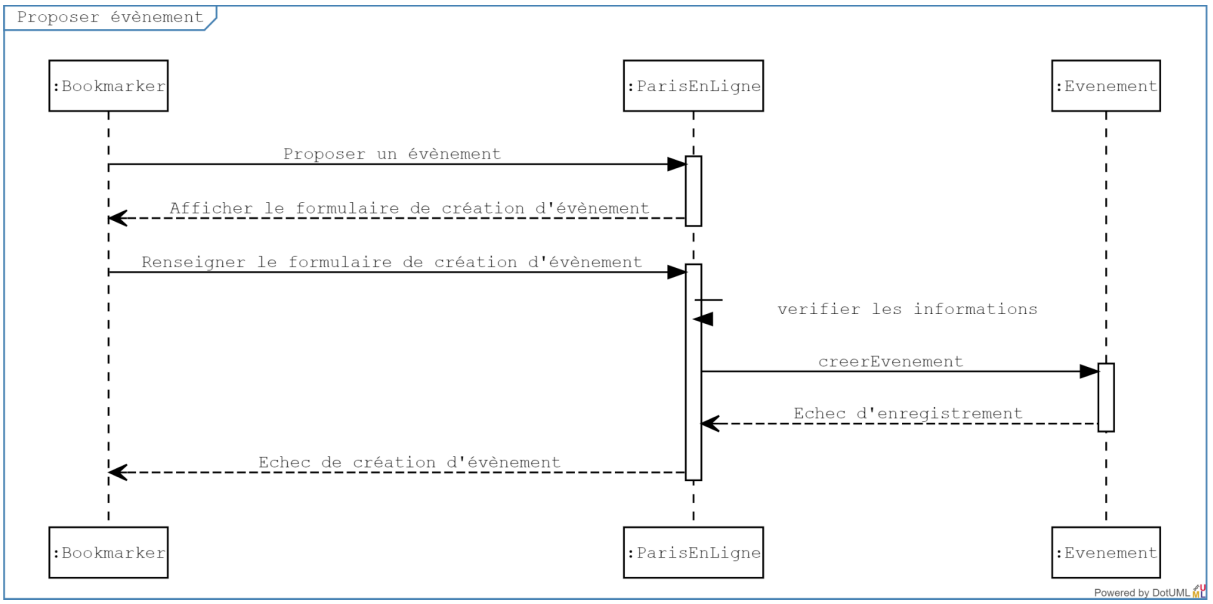
Cas 3: Paramétrer un évènement

Cette opération permet au bookmaker de paramétrer un évènement: échec



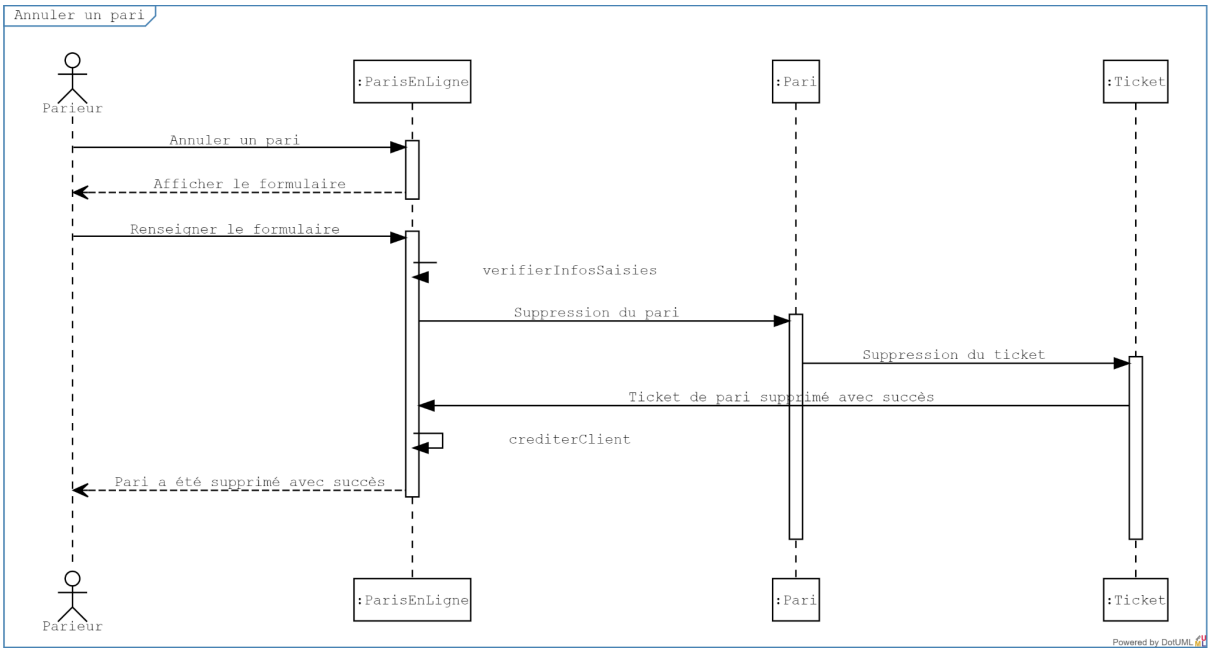
Cas 4:Proposer un évènement

Cette opération permet au bookmaker de proposer un évènement



Cas 5: Placer un pari: annulation

Cette opération permet au parieur d'annuler un pari .



Cas 6: Recharger ses jetons

Cette opération montre le cas d'échec pour le parieur de recharger ses jetons.

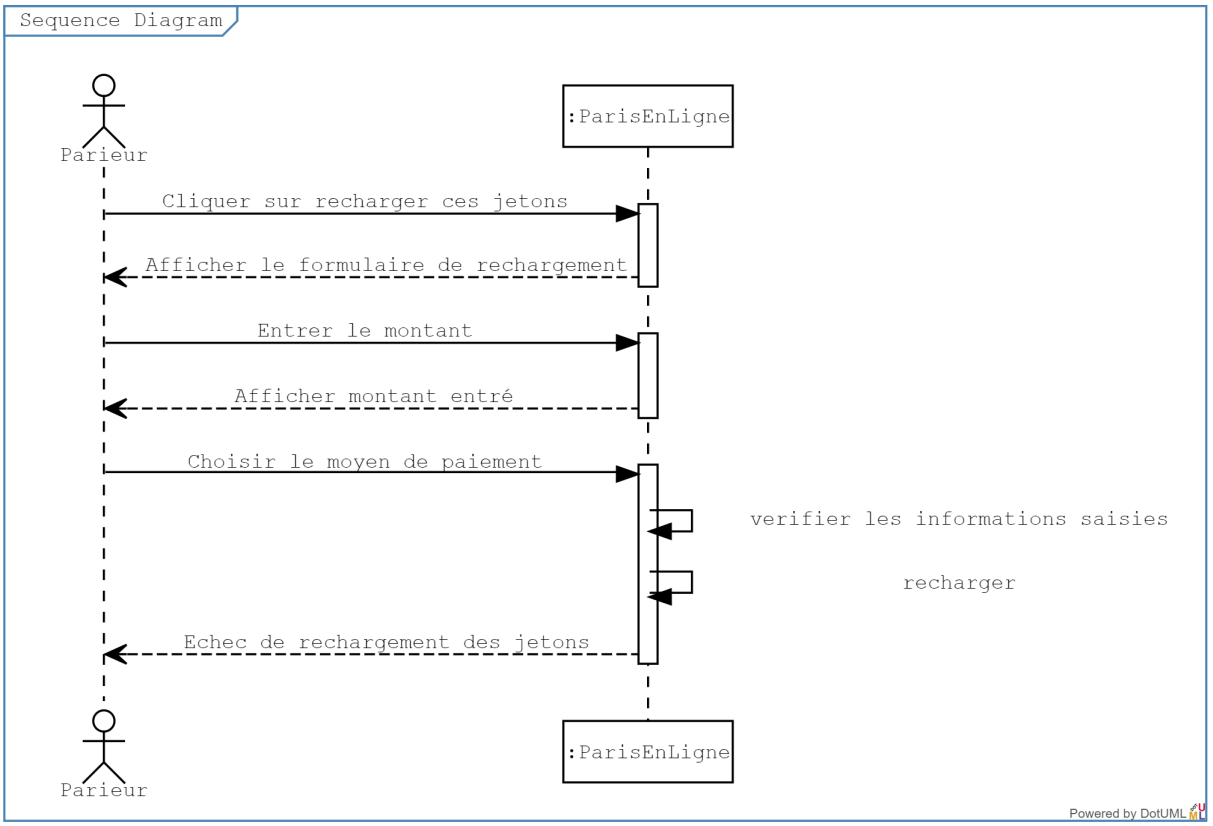


DIAGRAMME DES ÉTATS

Les diagrammes d'états-transitions d'UML décrivent le comportement interne d'un objet à l'aide d'un automate à états finis. Ils présentent les séquences possibles d'états et d'actions qu'une instance de classe peut traiter au cours de son cycle de vie en réaction à des événements discrets (de type signaux, invocations de méthode).

Ils spécifient habituellement le comportement d'une instance de classeur (classe ou composant), mais parfois aussi le comportement interne d'autres éléments tels que les cas d'utilisation, les sous-systèmes, les méthodes.

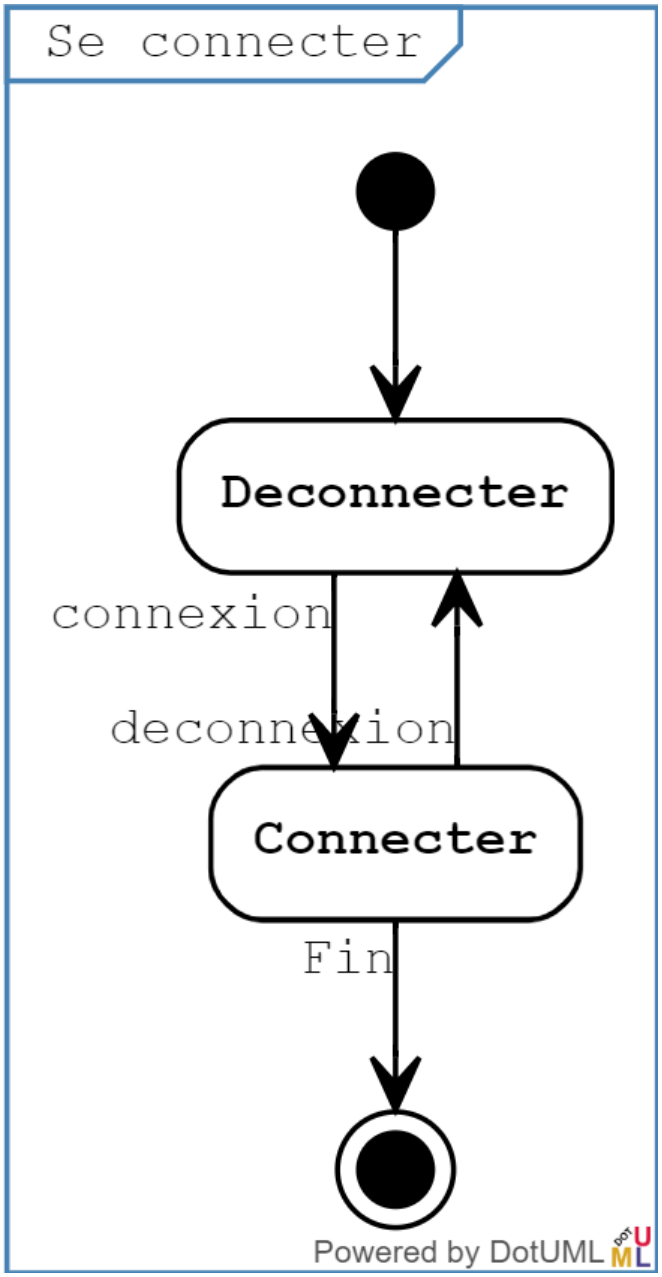
Le diagramme d'états-transitions est le seul diagramme, de la norme UML, à offrir une vision complète et non ambiguë de l'ensemble des comportements de l'élément auquel il est attaché. En effet, un diagramme d'interaction n'offre qu'une vue partielle correspondant à un scénario sans spécifier comment les différents scénarii interagissent entre eux.

La vision globale du système n'apparaît pas sur ce type de diagrammes puisqu'ils ne s'intéressent qu'à un seul élément du système indépendamment de son environnement.

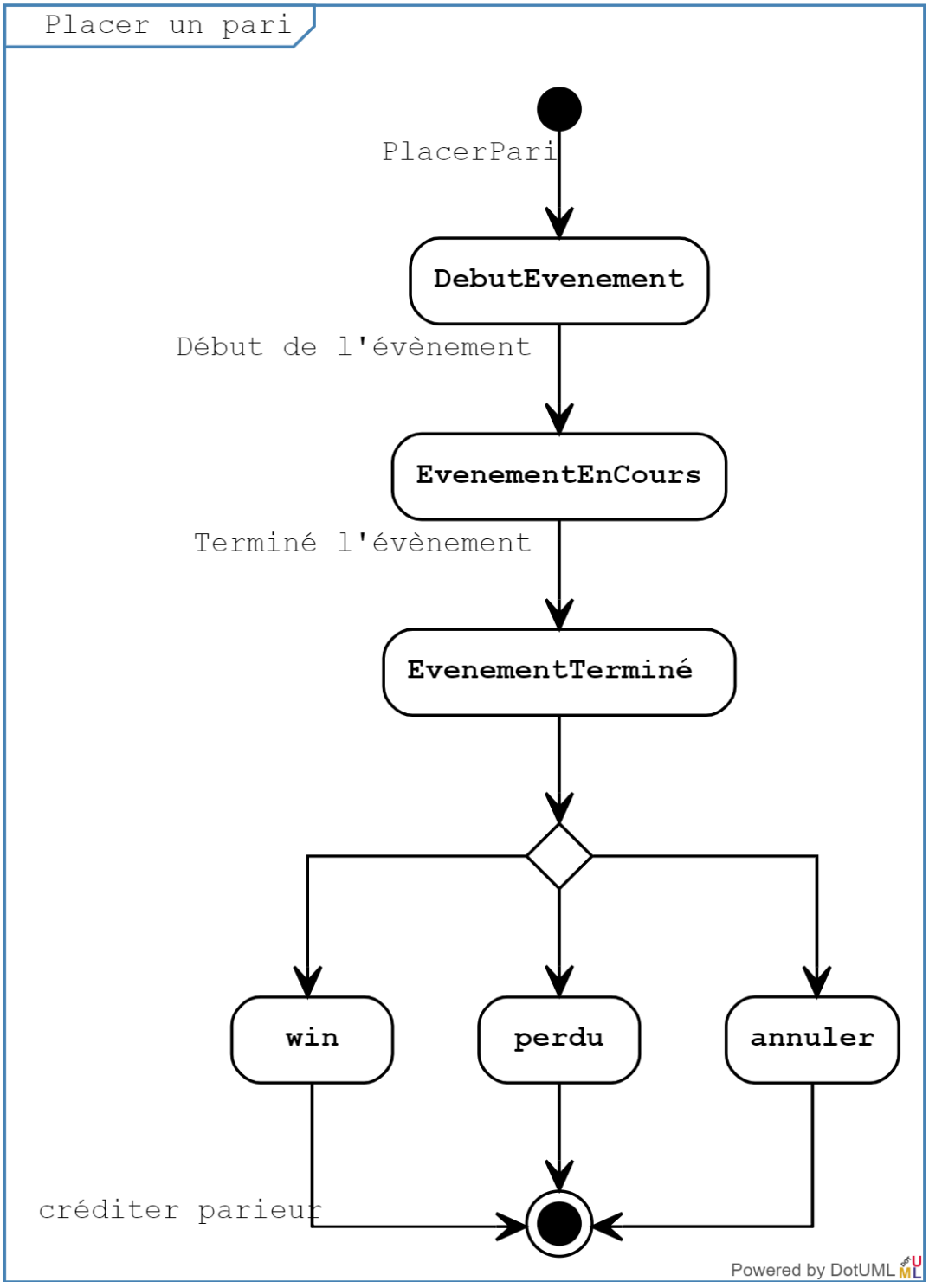
Concrètement, un diagramme d'états-transitions est un graphe qui représente un automate à états finis, c'est-à-dire une machine dont le comportement des sorties ne dépend pas seulement de l'état de ses entrées, mais aussi d'un historique des sollicitations passées.

Ci-dessous le diagramme d’état transition qui représente les réservations dans notre système

Cas: S’authentifier



Cas: Pari


Placer un pari
PlacerPari
DebutEvenement
Début de l'évènement
EvenementEnCours
Terminé l'évènement
EvenementTerminé
win
perdu
annuler
créditer parieur
Powered by DotUML

Cas: Evènement