

Rapport d'analyse de l'informatisation de l'Open Parc de Tennis de Lyon

Modèle: CPOA

Rapport: Rapport de CPOA

Auteur: Swann Benziane, Khalissa Rhoulam, Clément Darne,
Arthaud Morin

Version: 1.0.1

Date: 18/12/2021



Table des matières

Table des matières

I	INTRODUCTION.....	3
I.1	DESCRIPTION	3
I.1.1	Le module WEB.....	3
I.1.2	Le module JAVA.....	3
II	DESCRIPTION COMPLETE DU MODELE.....	4
II.1	DIAGRAMME DE PACKAGES.....	4
II.2	PACKAGE HEBERGEMENT	4
II.2.1	Liste des diagrammes et éléments d'analyse	4
II.2.2	Diagramme Cas Utilisation – Hébergement.....	5
II.2.3	Diagramme Classe – Hébergement.....	6
II.2.4	Scénario – Demande Hébergement	7
II.2.5	Diagramme Séquence – Demande Hébergement	8
II.2.6	Maquette – Demande Hébergement	10
II.3	PACKAGE PLANNING DES MATCHS	15
II.3.1	Liste des diagrammes et éléments d'analyse	15
II.3.2	Diagramme de Cas d'Utilisation du Planning.....	16
II.3.3	Diagramme de Classe du Planning.....	17
II.3.4	Diagramme d'Activité – Choisir Arbitre.....	18
II.3.5	Diagramme d'Activité – Générer Planning Jour (jour, listeMatch).....	19
II.3.6	Diagramme d'Activité – Modifier Match	20
II.3.7	Diagramme de Séquence Déplacer Match.....	21
II.3.8	Diagramme de Séquence – Réserver Court Annexe	22
II.3.9	Diagramme d'Activité – Générer Tournoi	23
II.3.10	Maquette – Réserver Court Annexe	26
III	CONCEPTION DU MODELE.....	29
III.1	DIAGRAMMES DE CLASSES DE CONCEPTION	29
III.1.1	Diagramme de Classe – Planification.....	29
III.1.2	Diagramme de Classe – Hébergement.....	32
III.2	CLASSES JAVA GENEREES	33
III.2.1	Package Planification – Arbitre	33
III.2.2	Package Hébergement – TypeChambre	37
III.3	SCRIPTS SQL GENERES	42
III.3.1	Package Planning – Court	42
III.3.2	Package Hébergement – Demande	42

I Introduction

I.1 Description

Le projet consiste à développer l'application OPEN PARC de Lyon, une application qui doit permettre d'encadrer et faciliter l'organisation du tournoi. Elle se compose de deux modules principaux :

- Un module WEB pour la gestion de l'hébergement.
- Un module JAVA pour le planning des matchs.

I.1.1 Le module WEB

Le module WEB est dédié au responsable de l'hébergement du tournoi, aux gérants des hébergements (hôtels) et aux VIPs. Il doit permettre :

- Aux gérants des hébergements d'entrer des informations relatives à leurs établissements (nombre de chambres, lits, services, etc..) et de les mettre à jour.
- De faciliter la demande d'hébergement aux VIPs en leur offrant un catalogue de chambres disponibles en fonction de critères sélectionnés par leurs soins.
- De faciliter au responsable de l'hébergement du tournoi la gestion et le transfert des demandes d'hébergement vers les gérants.

I.1.2 Le module JAVA

Le module JAVA est dédié aux joueurs et à l'organisateur du tournoi.

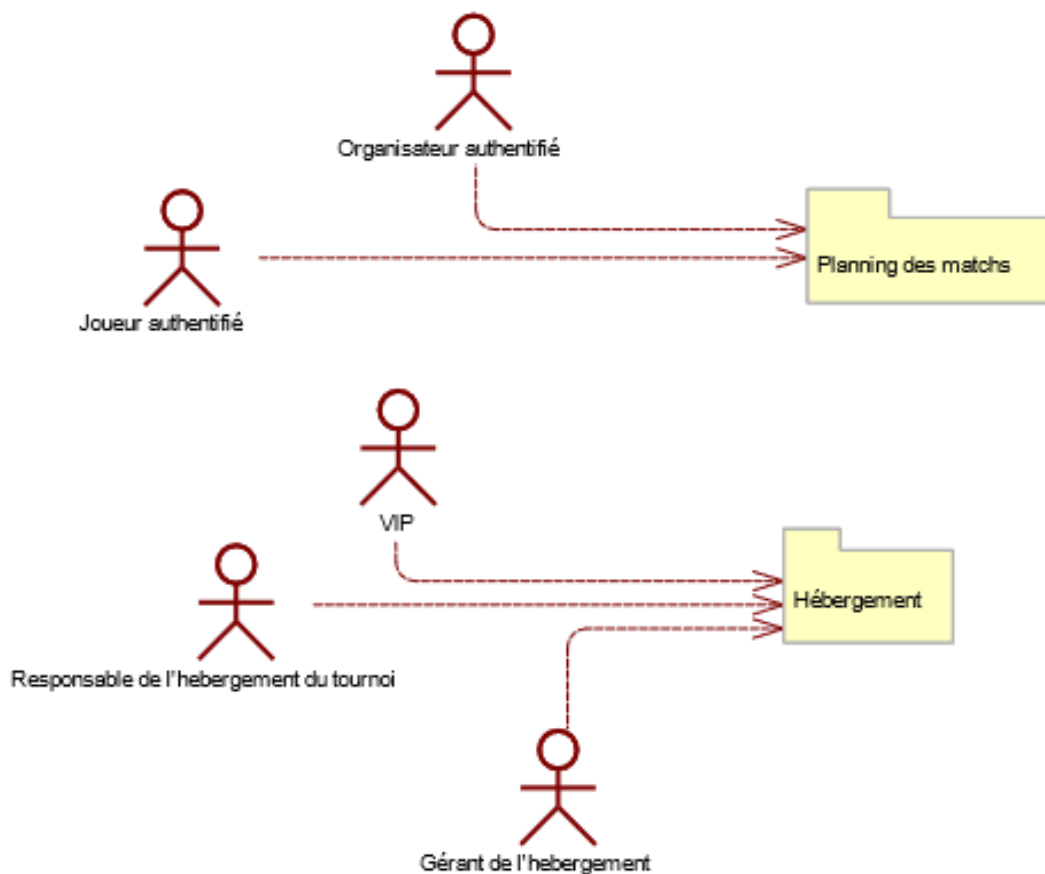
Une première application doit permettre à l'organisateur de :

- Générer un planning des matchs (arbre de tournoi et emploi du temps). La gestion des affrontements est automatique, l'organisateur devra seulement indiquer le joueur ou l'équipe gagnant/e.
- De modifier les joueurs qui s'affrontent pour le premier tour de chaque tournoi.
- De modifier les arbitres de sorte qu'il n'aient pas la même nationalité que les joueurs qui s'affrontent.
- De modifier les équipes de ramasseurs de balles.
- De déplacer un match (changer de court et/ou de créneau).

Une seconde application doit permettre aux joueurs de réserver un court annexe libre pour un entraînement personnel.

II Description complète du modèle

II.1 Diagramme de packages

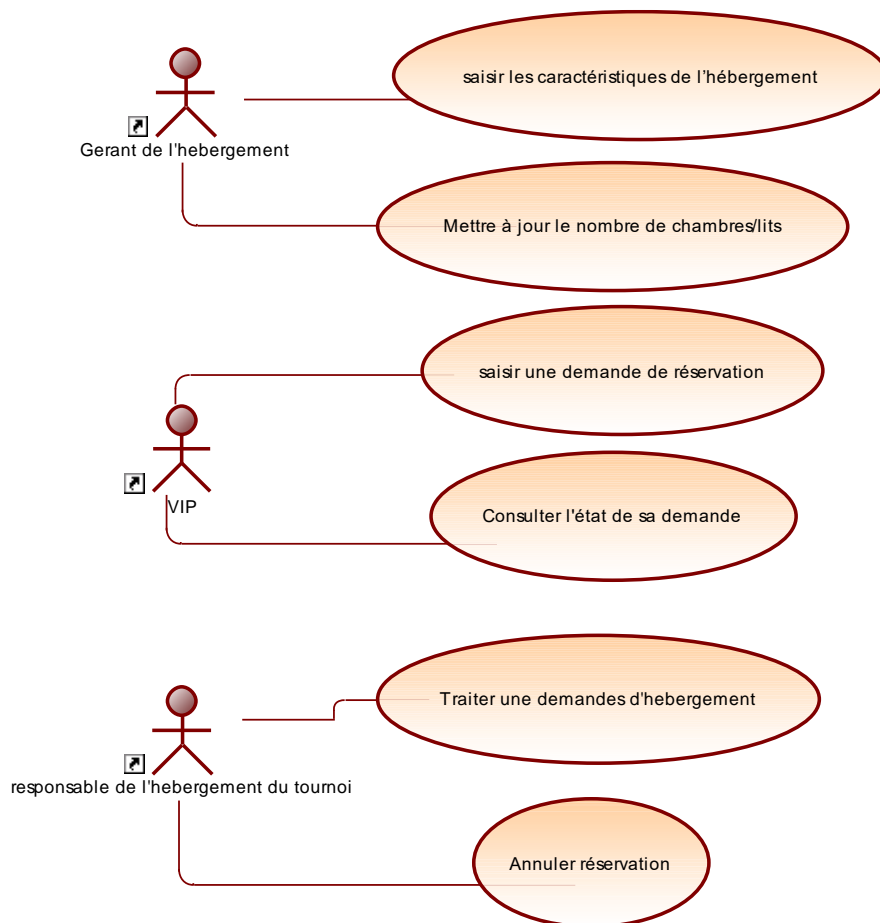


II.2 Package Hébergement

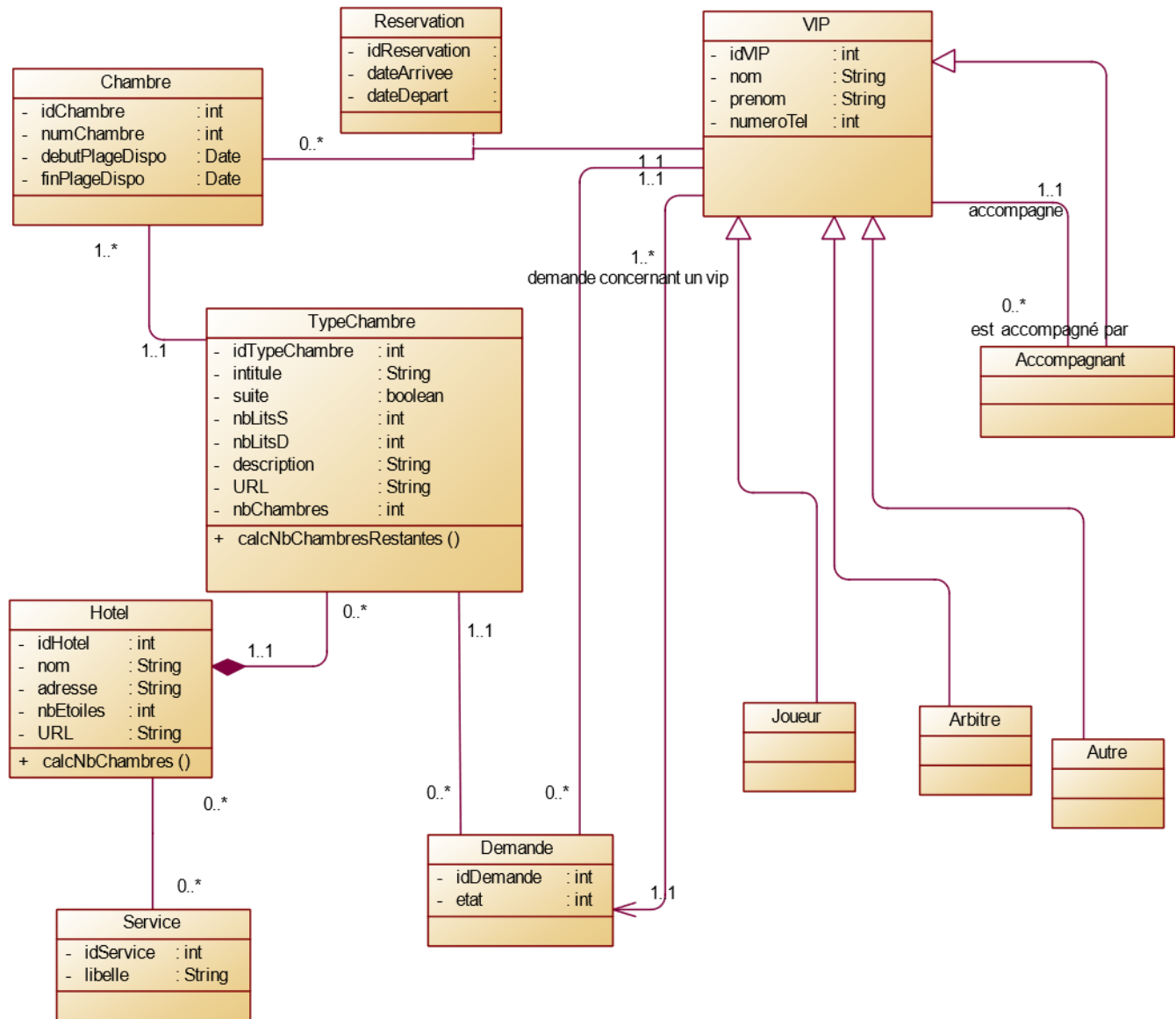
II.2.1 Liste des diagrammes et éléments d'analyse

1. Cas Utilisation – Hébergement
2. Diagramme Classe – Hébergement
3. Scénario – Demande Hébergement
4. Diagramme Séquence – Demande Hébergement
5. Maquette – Demande Hébergement

II.2.2 Diagramme Cas Utilisation – Hébergement



II.2.3 Diagramme Classe – Hébergement

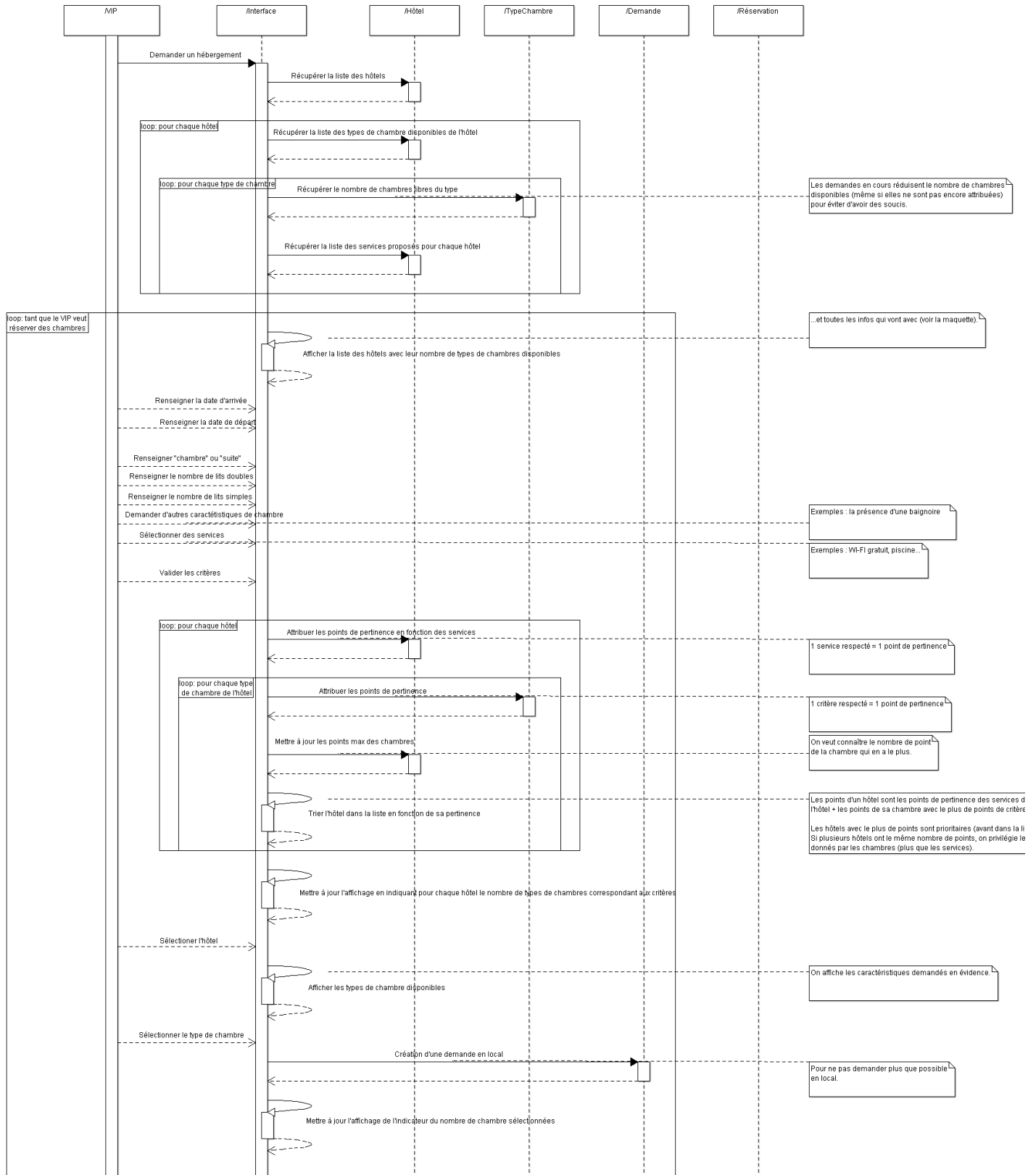


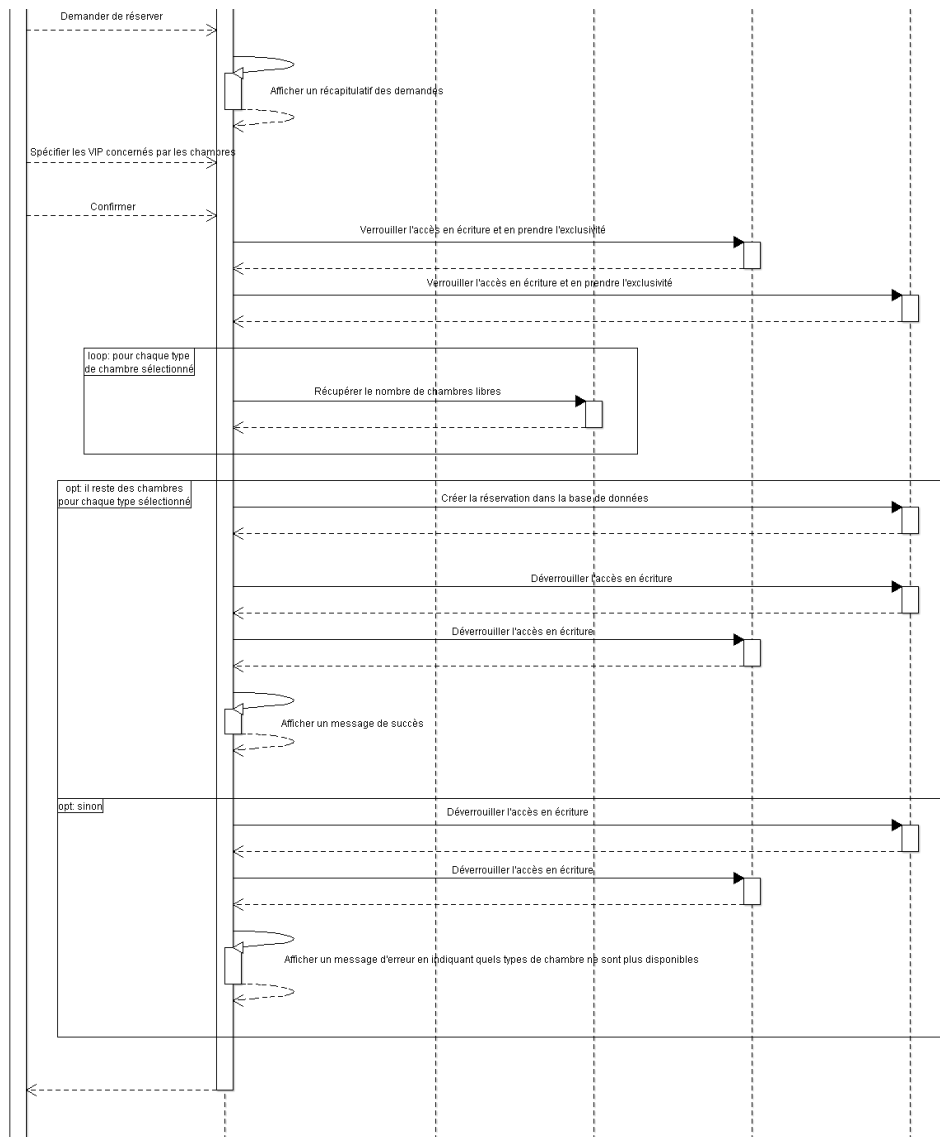
II.2.4 Scénario – Demande Hébergement

Un joueur fait une demande d'hébergement pour lui et sa famille (sa femme et deux enfants)

1. Joueur clique sur demande d'hébergement.
2. Joueur redirigé vers un nouvel onglet avec la liste des hôtels et des champs de critères.
3. Joueur coche le champ « chambre ».
4. Joueur rentre 1 dans le champ « nombre de lits doubles ».
5. Joueur rentre 2 dans le champs « nombre de lits simples ».
6. Joueur renseigne la date d'arrivée.
7. Joueur renseigne la date de départ.
8. Joueur parcourt les services et sélectionne « baignoire » et « petit-déjeuner intégré ».
9. Joueur clique sur « valider les critères ».
10. Joueur redirigé vers un onglet avec une liste d'hôtel répondant à sa demande.
11. Joueur clique sur un hôtel intéressant.
12. Joueur redirigé vers un nouvel onglet avec une liste de chambres répondant aux critères dans l'hôtel sélectionné.
13. Joueur sélectionne une chambre (« sélectionner ») et clique sur « réserver ».
14. Joueur redirigé vers un onglet récapitulant sa réservation.
15. Joueur saisit le nom des personnes l'accompagnant.
16. Joueur clique sur « valider ».
17. Fin de la réservation.

II.2.5 Diagramme Séquence – Demande Hébergement





II.2.6 Maquette – Demande Hébergement

Browser

http://openlyon.fr/logement

Réservation de logement

Date d'arrivée : 10 /12 / 2021

Date de départ : 19 /12 / 2021

Type de chambre : Suite

Nombre de lits doubles : 1

Nombre de lits simples : 2

Services :

- ☒ Piscine
- ☐ Petit-déjeuner inclus
- ☐ Parking inclus
- ☒ Animaux acceptés

Chercher

Browser

http://openlyon.fr/logement

Réservation de logement

Q nom d'hôtel

Date d'arrivée : 10 / 12 / 2021

Date de départ : 19 / 12 / 2021

Type de chambre : Suite

Nombre de lits doubles : 1

Nombre de lits simples : 2

Services :

- ☒ Piscine
- ☐ Petit-déjeuner inclus
- ☐ Parking inclus
- ☒ Animaux acceptés
- ☐ Salle de sport
- ☐ Wifi gratuit

Campanile Lyon Centre

Adresse

Note

Piscine Animaux acceptés Bar Restaurant

Nombre d'étoiles

Réserver

Nemea Appart Hotel

Adresse

Note

Piscine Animaux acceptés Parking disponible

Nombre d'étoiles

Réserver

Hôtel de paris

Adresse

Note

Piscine Animaux acceptés

Nombre d'étoiles

Réserver

Browser

http://openlyon.fr/logement

Réservation de logement

Campanile Lyon Centre

nombre d'étoiles

Choisir les chambres

IMG

Services :

Piscine

Animaux acceptés

Bar

Restaurant

adresse

Description :

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute inure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum

IMG

IMG

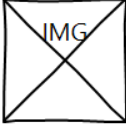
IMG

Browser

http://openlyon.fr/logement

Réservation de logement

Choisissez vos chambres



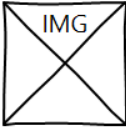
taille de la chambre : 17m2

taille du lit : 160*200

Nombre : 0 ▼

Sélectionner

chambre double, 1 lit double



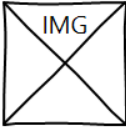
taille de la chambre : 13m2

taille des lits : 80*190

Nombre : 1 ▼

Sélectionner

chambre double, 2 lit simples



taille de la chambre : 40m2

taille du lit : 180*200

douche avec baignoire

Nombre : 1 ▼

Sélectionner

Suite double, 1 lit double

Réserver

Browser

http://openlyon.fr/logement

Réservation de logement

Récapitulatif de votre réservation

Hôtel Campanile Lyon Centre

IMG

taille de la chambre : 13m2

taille des lits : 80*190

Nombre : 1 ▼

chambre double, 2 lit simples

Sélectionner

Nom des personnes dans cette chambre :

À Remplir

IMG

taille de la chambre : 40m2

taille du lit : 180*200

Nombre : 1 ▼

douche avec baignoire

Sélectionner

Suite double, 1 lit double

Nom des personnes dans cette chambre :

À Remplir

Valider la réservation

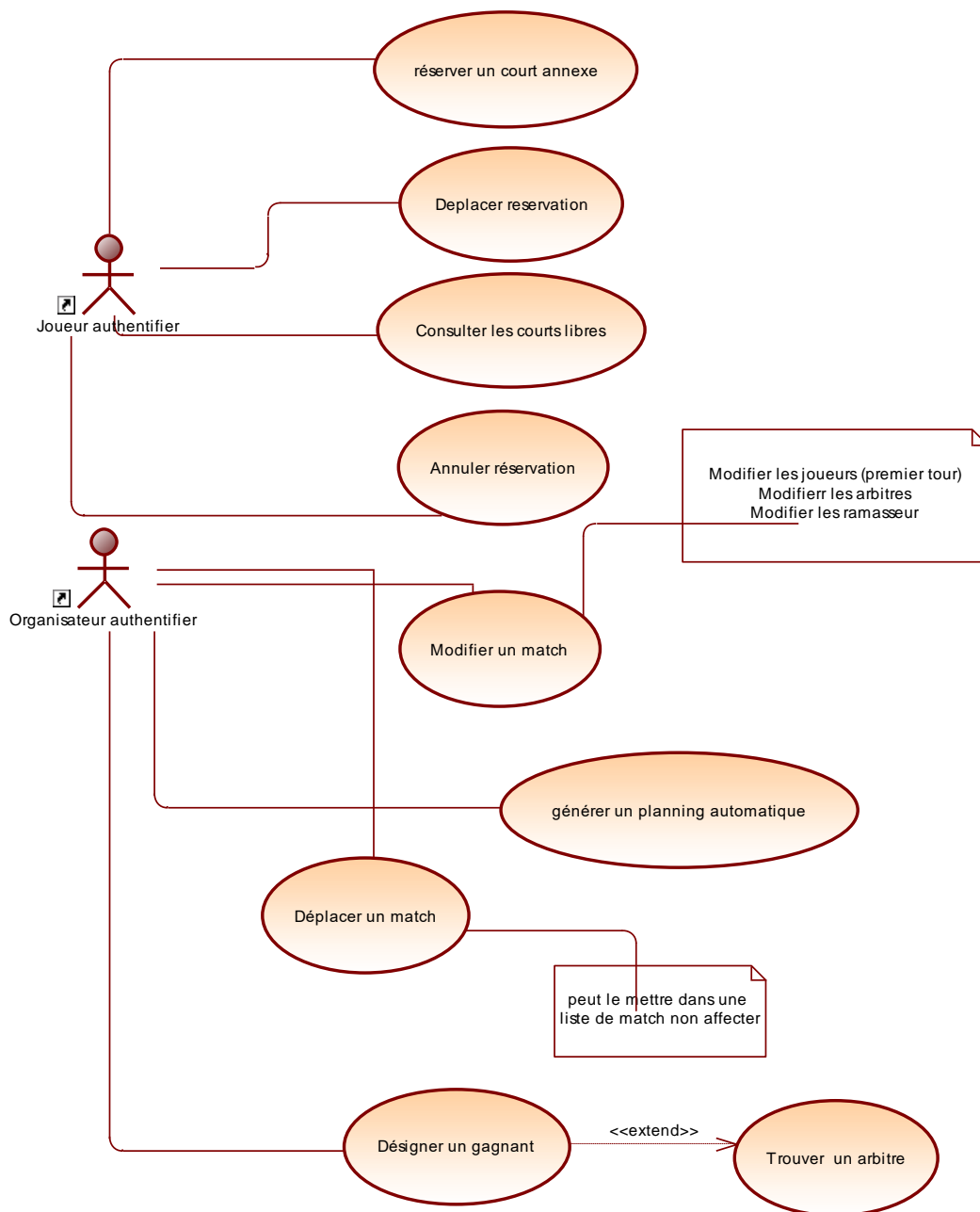
II.3 Package planning des matchs

II.3.1 Liste des diagrammes et éléments d'analyse

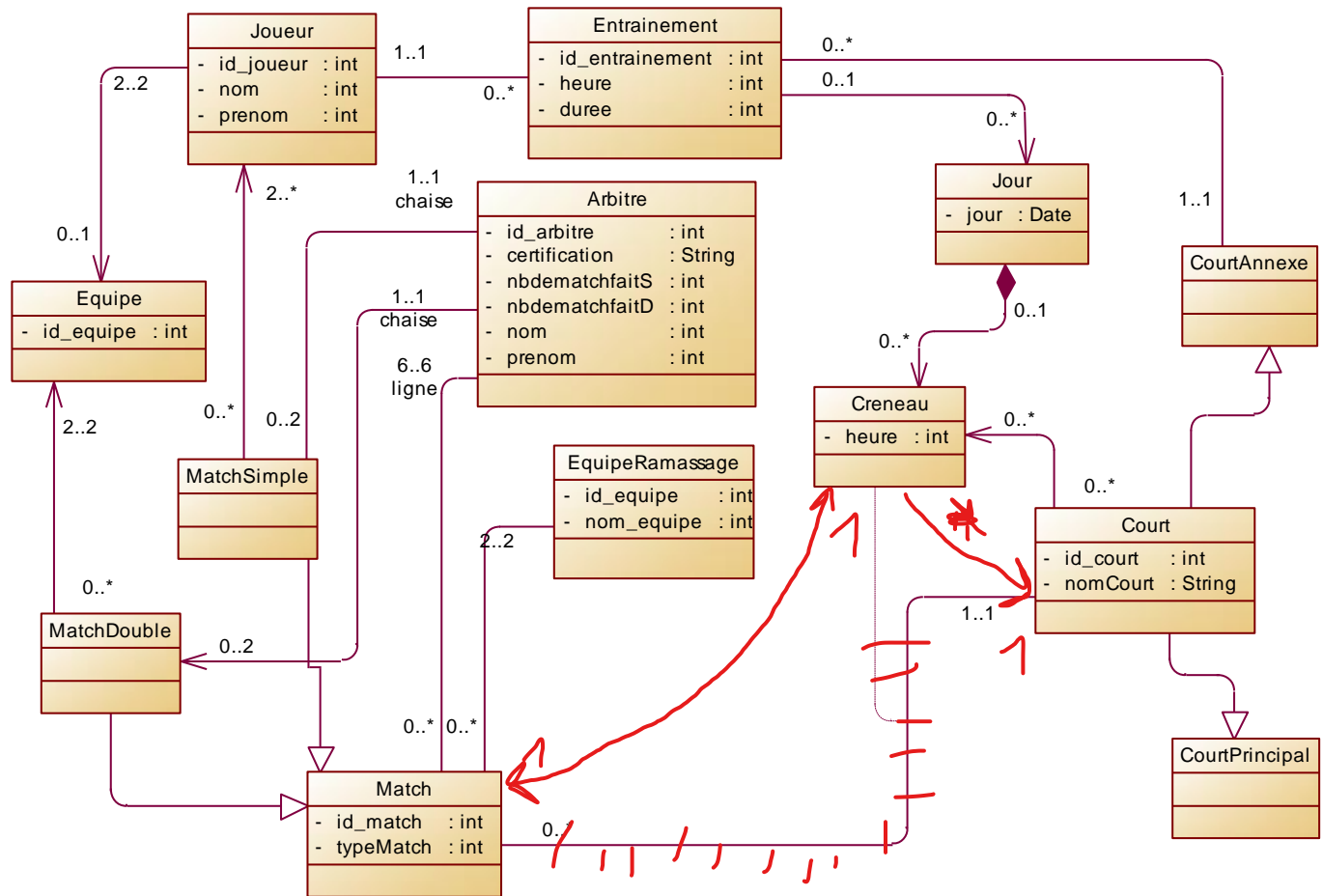
1. Diagramme de Cas d'Utilisation – Planning
2. Diagramme de Classes – Planning
3. Diagramme d'Activité – Choisir Arbitre
4. Diagramme d'Activité – Générer Planning Jour
5. Diagramme d'Activité – Modifier Match
6. Diagramme de Séquence – Déplacer Match
7. Diagramme de Séquence – Réserver Court Annexe
8. Diagramme d'Activité – Générer Tournoi
9. Maquette – Réserver Court Annexe

II.3.2 Diagramme de Cas d'Utilisation du Planning

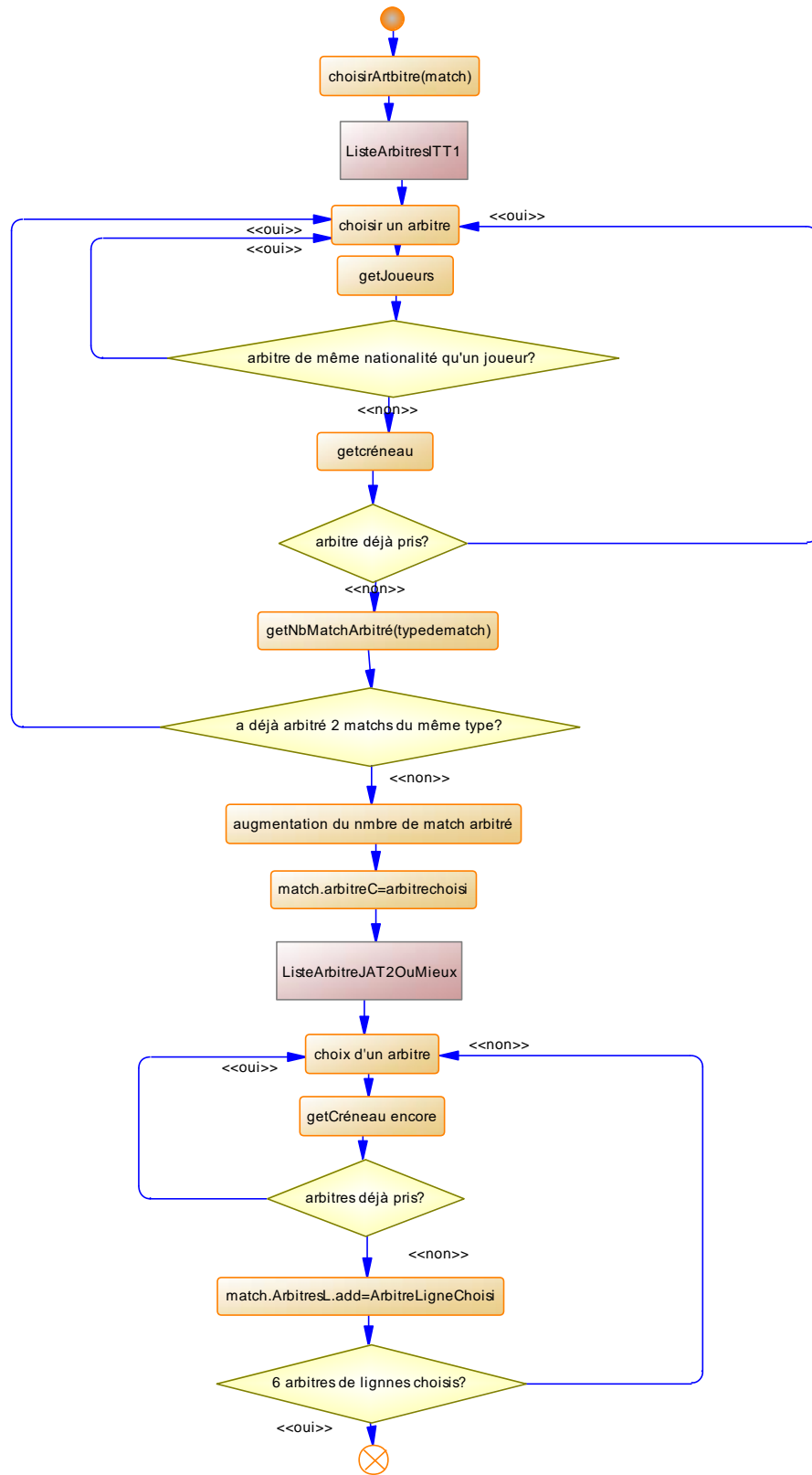
Modèle orienté objet
Modèle : CPOA
Package : planning des matchs
Diagramme : CasUtilisationPlanning
Auteur : p1808009 Date: 18/12/2021
Version:



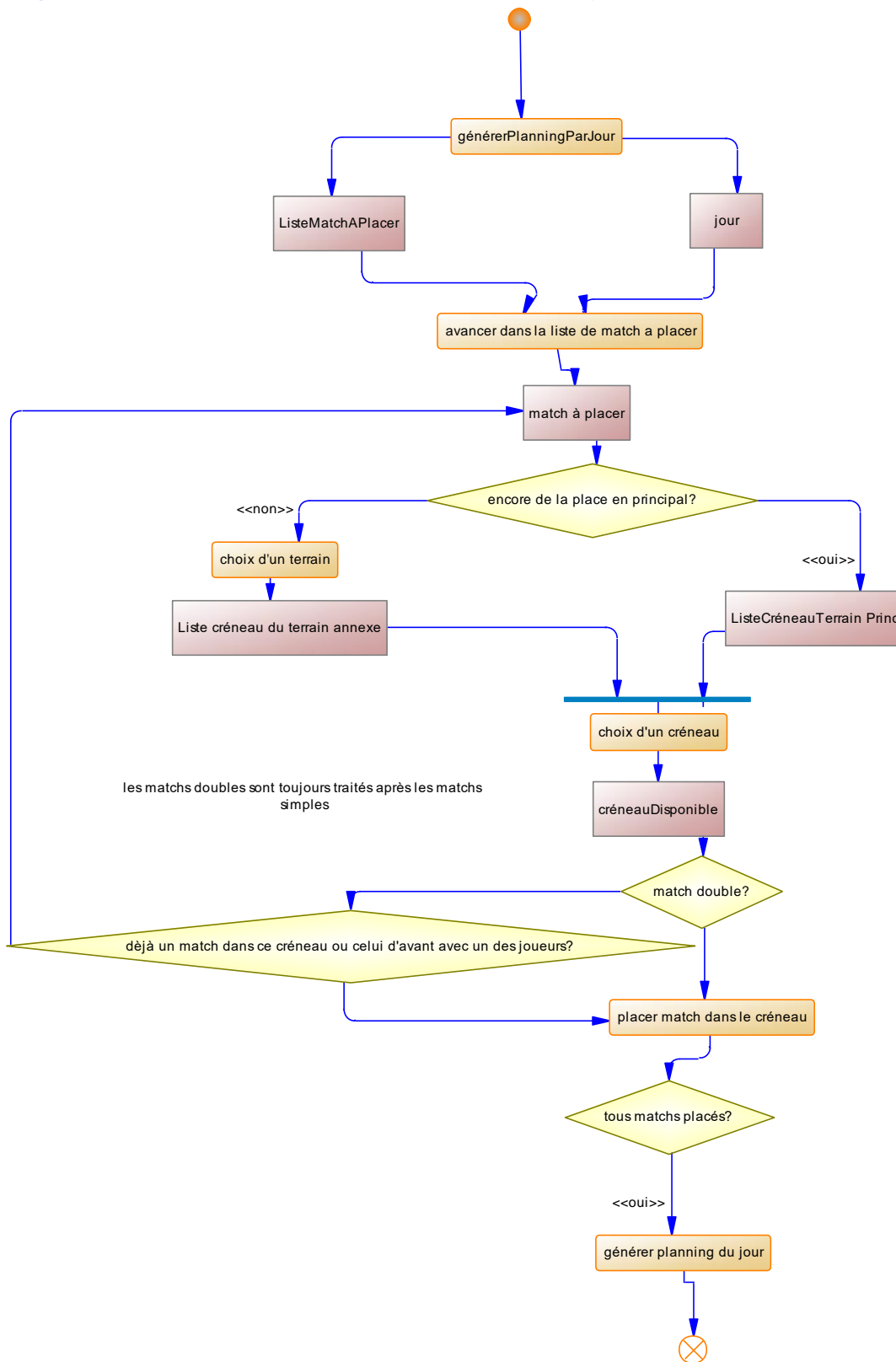
II.3.3 Diagramme de Classe du Planning



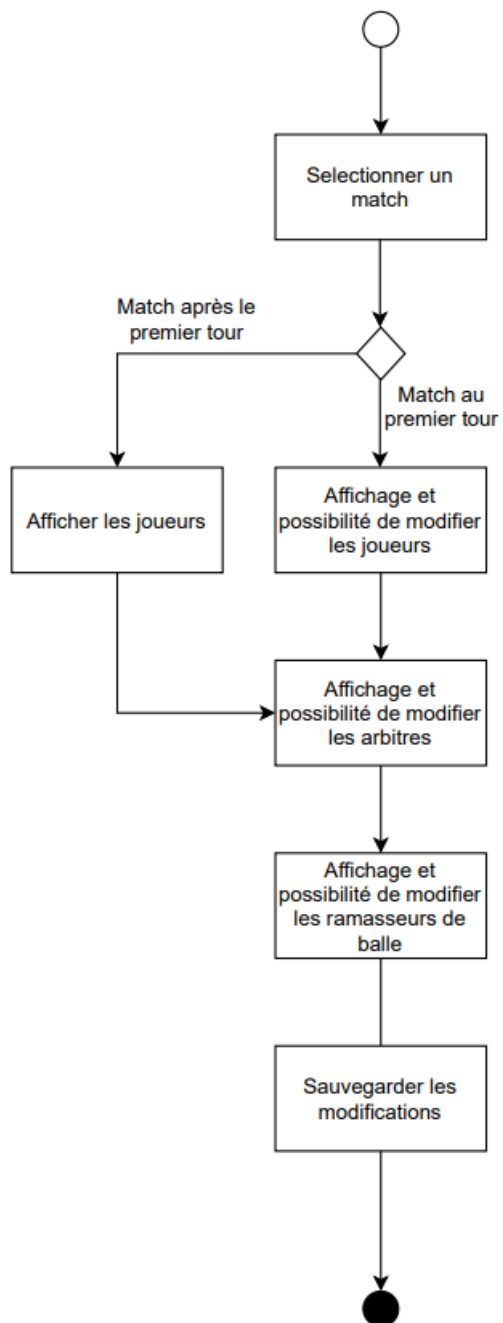
II.3.4 Diagramme d'Activité – Choisir Arbitre



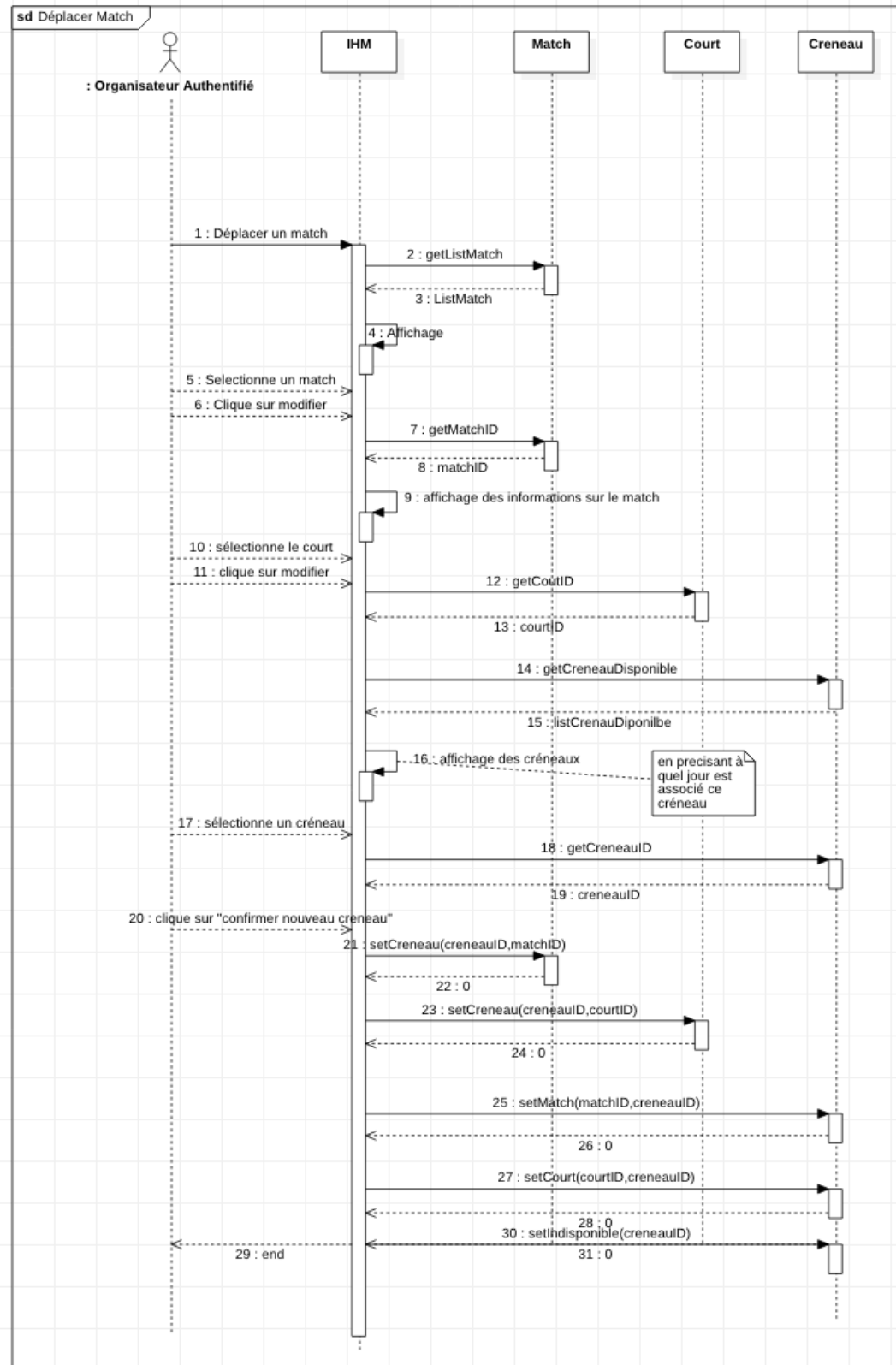
II.3.5 Diagramme d'Activité – Générer Planning Jour (jour, listeMatch)



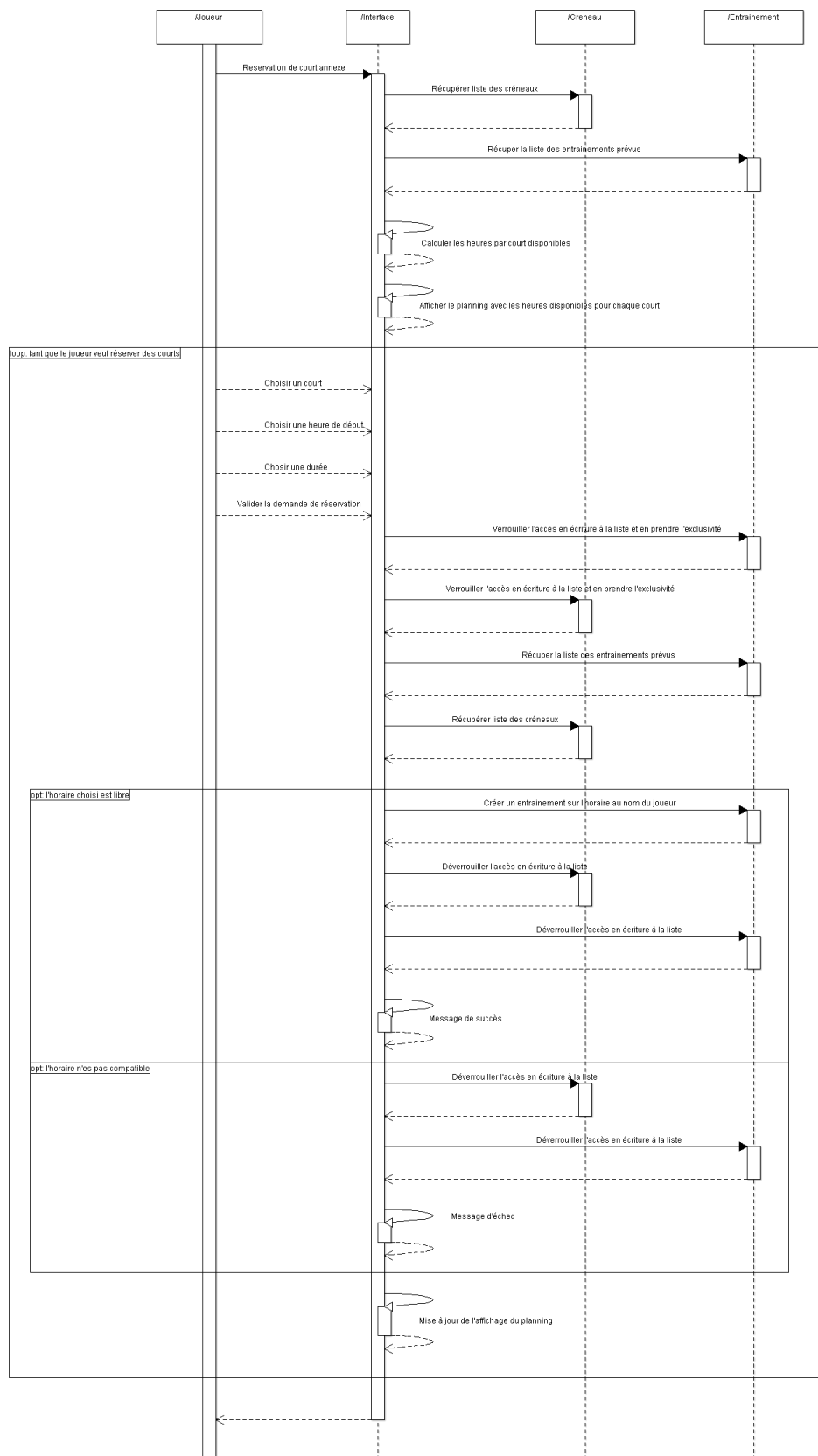
II.3.6 Diagramme d'Activité – Modifier Match



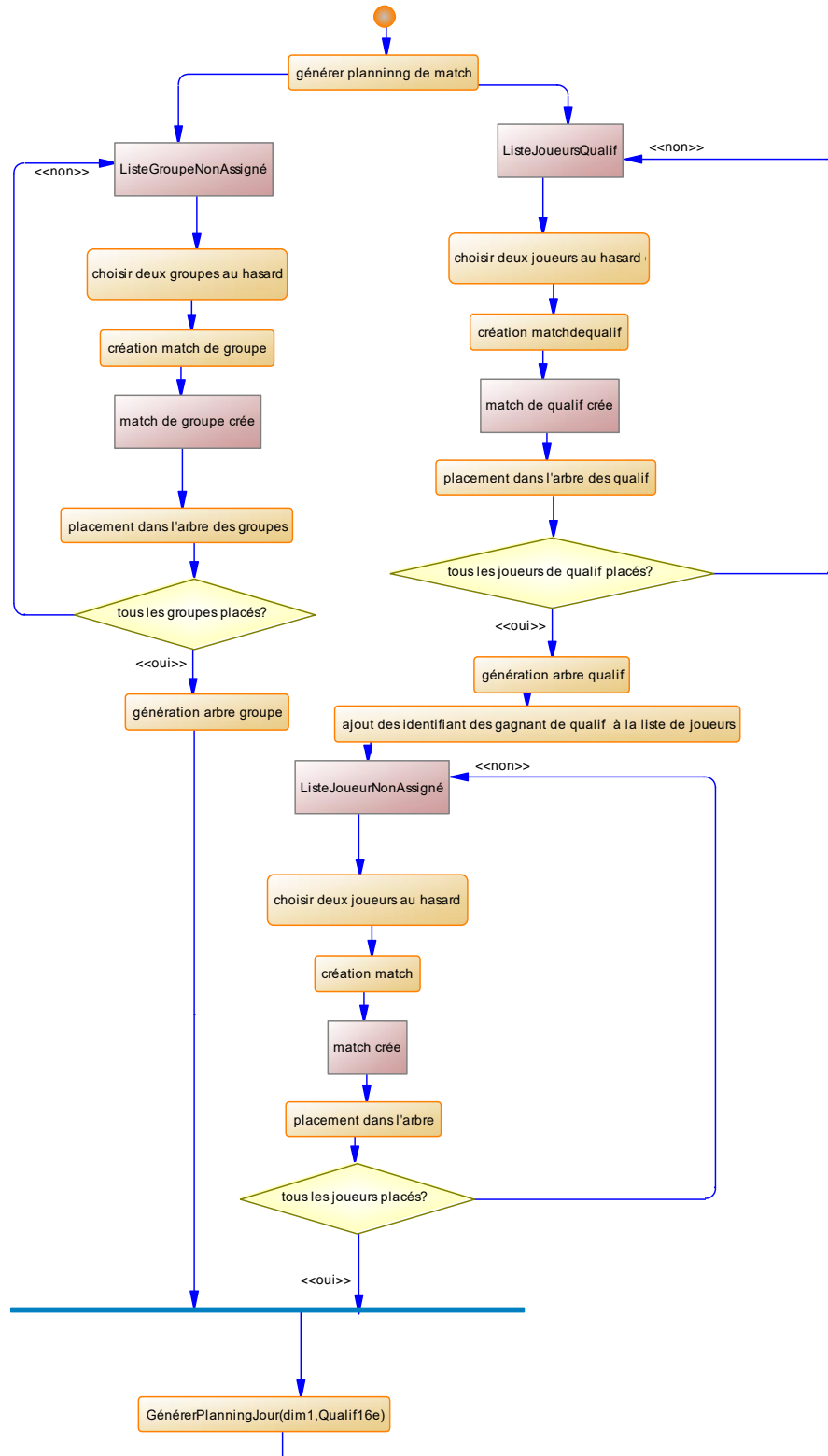
II.3.7 Diagramme de Séquence Déplacer Match

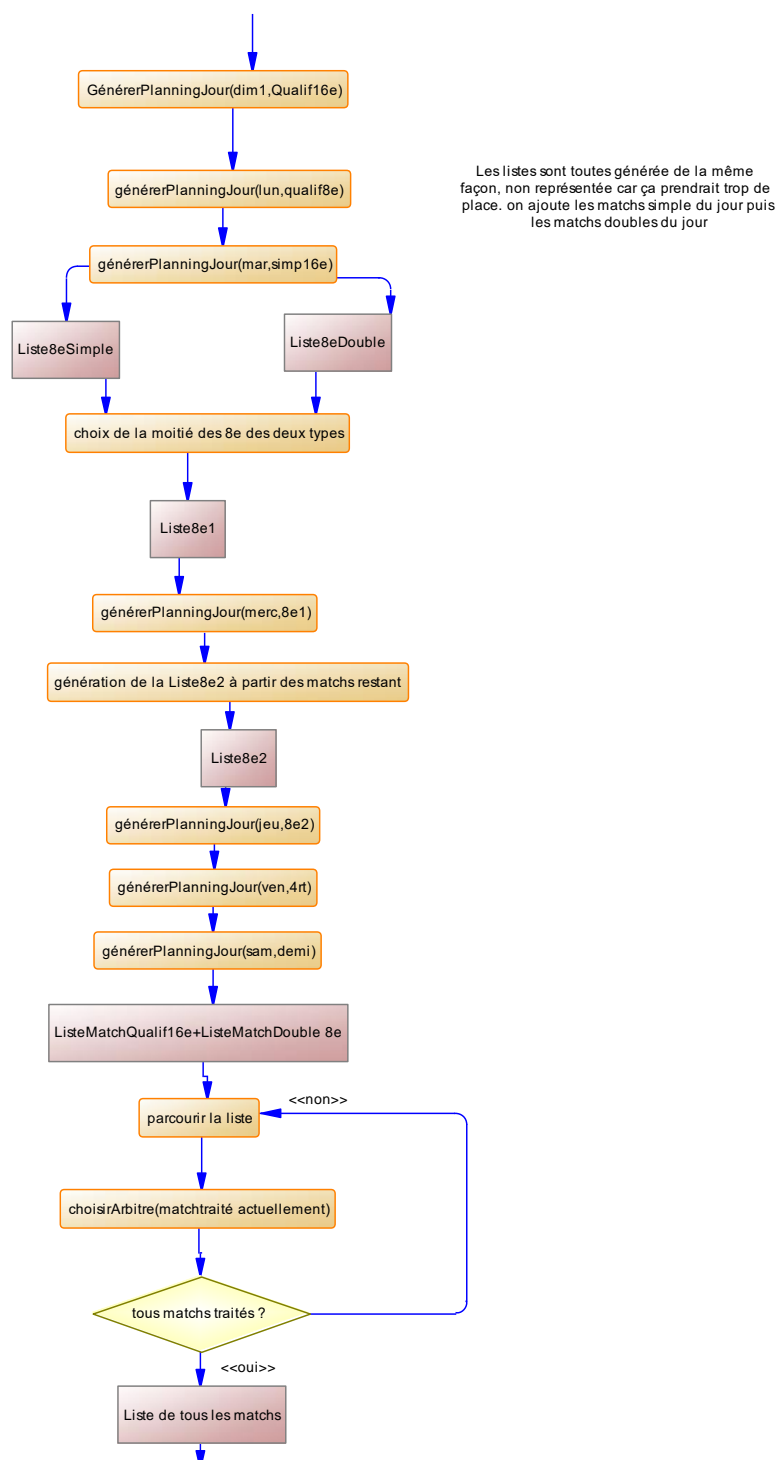


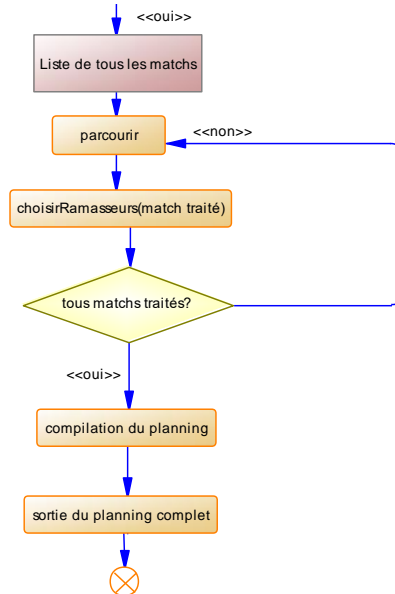
II.3.8 Diagramme de Séquence – Réserver Court Annexe



II.3.9 Diagramme d'Activité – Générer Tournoi







II.3.10 Maquette – Réserver Court Annexe

Réservation d'entraînement

Date

12 / 12 / 2021

Court

Tous

Durée

1h

30

Réservation d'entraînement

Date

12/12/2021

Court

Tous

Durée

1h

30

Choix heure de début

Terrain A :

8h

8h30

10h

11h

11h30

12h

14h

14h30

15h

16h

18h

18h30

Voir plus

Terrain B :

9h

13h

15h

18h30

Voir plus

Terrain C :

9h30

10h

10h30

14h

14h30

16h

17h

17h30

Voir plus

Réservation d'entraînement

Date

12/12/2021

Court

Tous

Durée

1h

30

Choix heure de début

Terrain A :

8h

8h30

10h

11h

11h30

12h

14h

14h30

15h

16h

18h

18h30

Voir plus

Terrain B :

9h

13h

15h

18h30

Voir plus

Terrain C :

9h30

10h

10h30

14h

14h30

16h

17h

17h30

Voir plus

Réserver

Réservation d'entraînement

Votre réservation :

Date : 12/12/2021

Heure de début : 12h

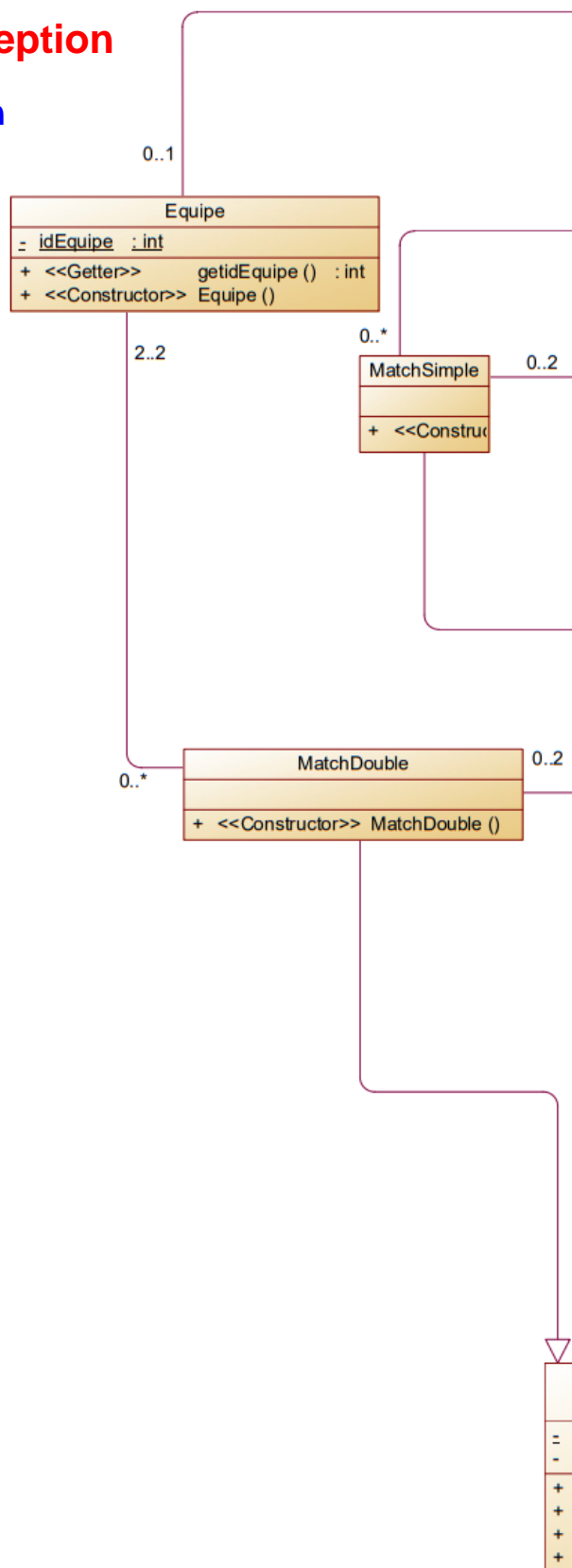
Heure de fin : 13h30

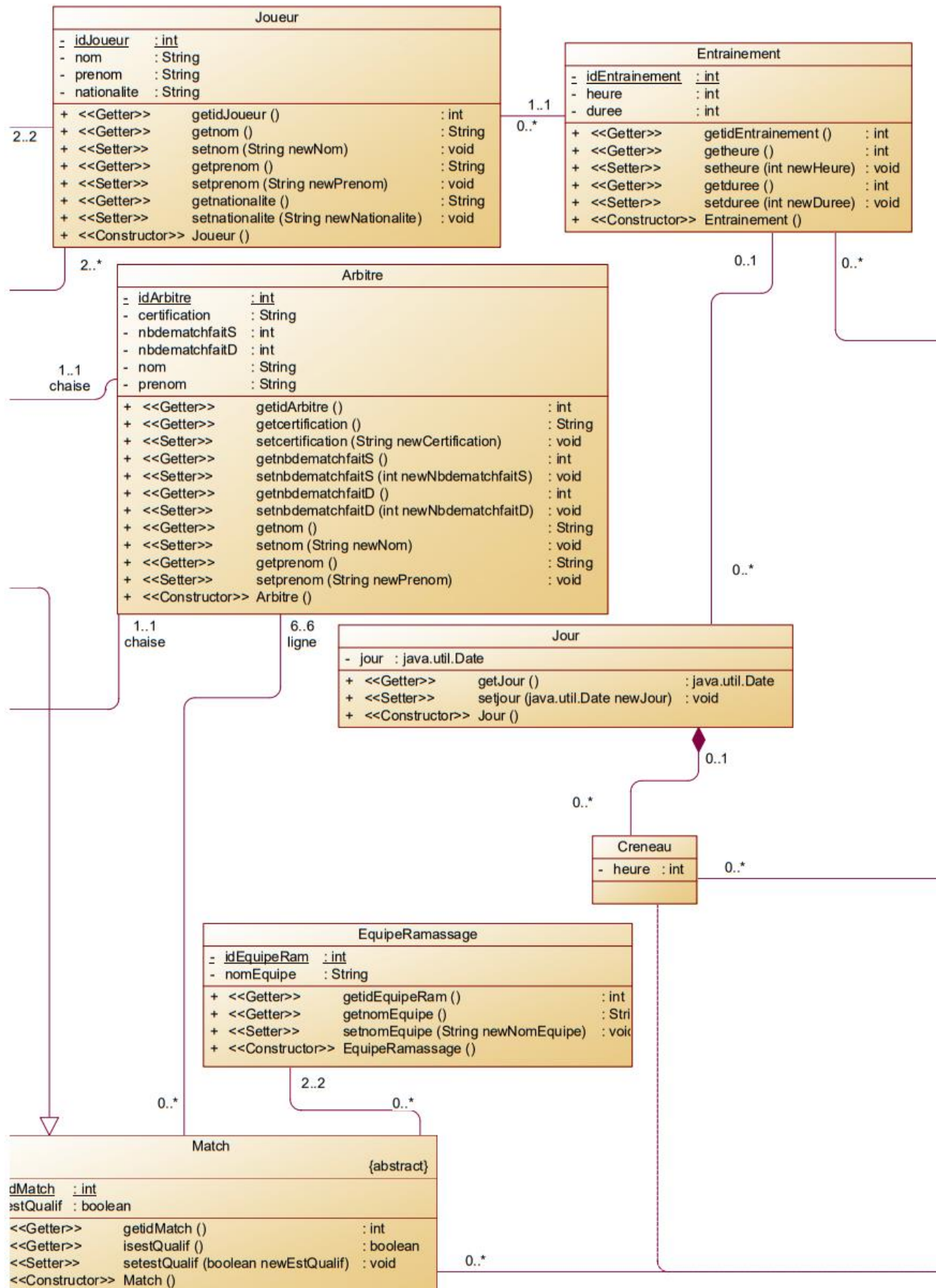
Court : A

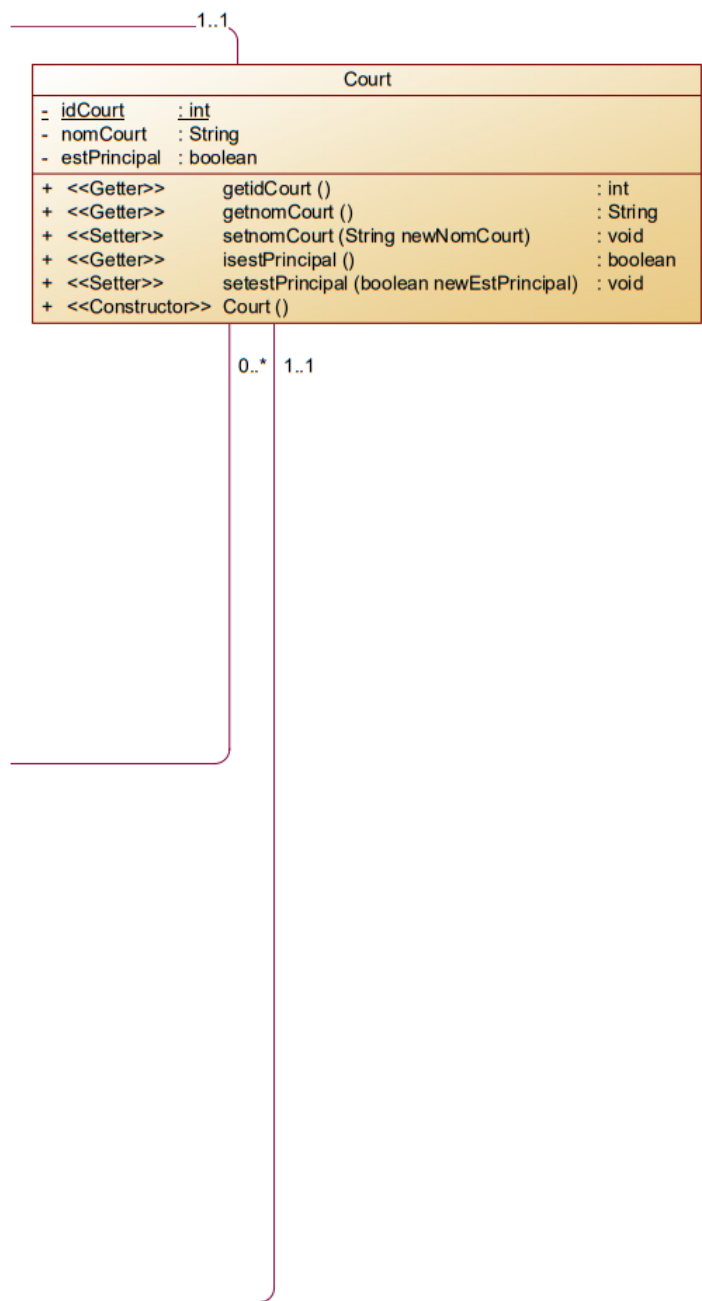
III Conception du modèle

III.1 Diagrammes de classes de conception

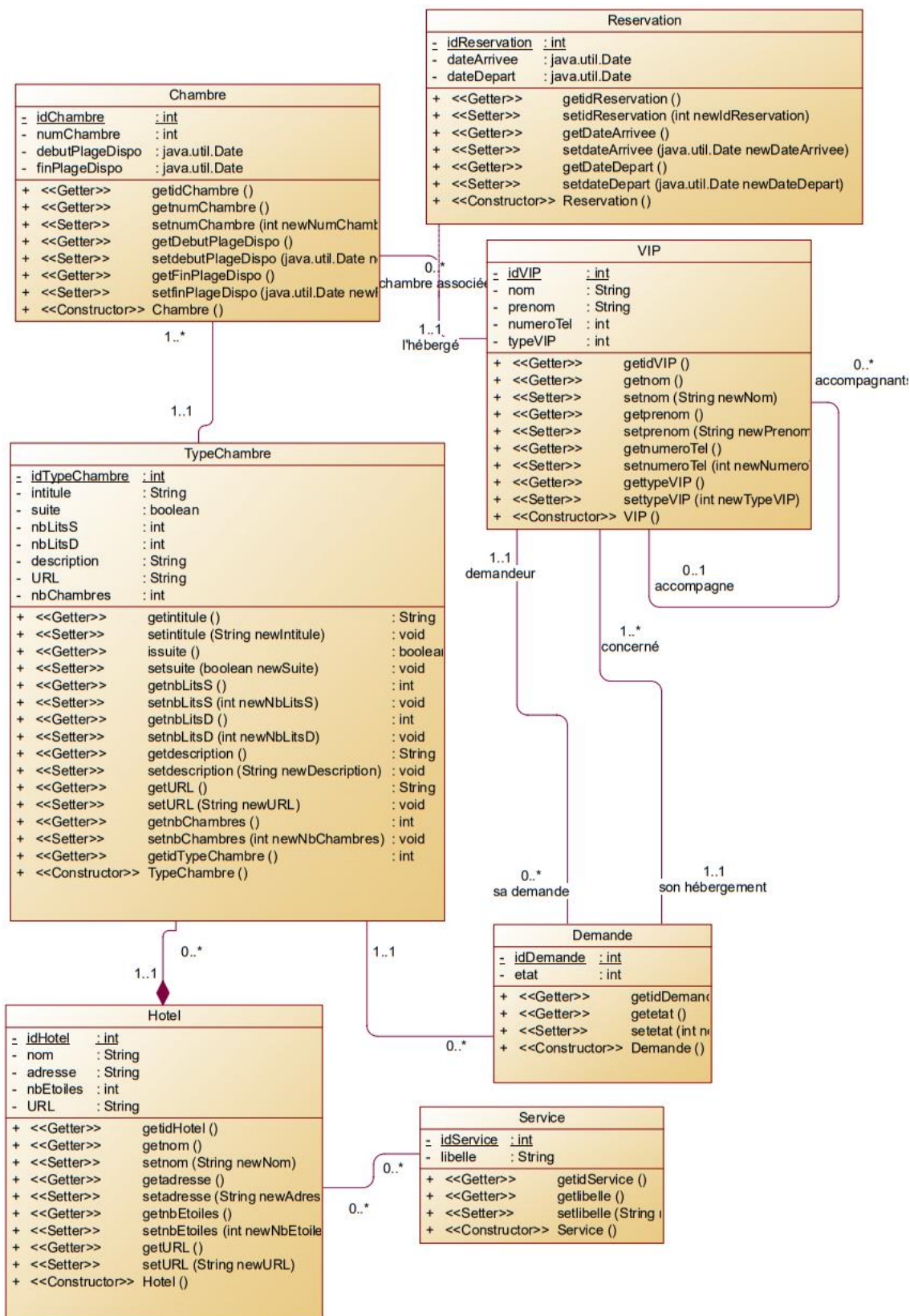
III.1.1 Diagramme de Classe – Planification







III.1.2 Diagramme de Classe – Hébergement



III.2 Classes Java générées

Voici quelques exemples de classes Java générées avec PowerAMC.

III.2.1 Package Planification – Arbitre

```
/*  
*****  
* Module: Arbitre.java  
* Author: swann  
* Purpose: Defines the Class Arbitre  
*****  
*/  
  
package planning_des_matches;  
  
import java.util.*;  
  
public class Arbitre {  
    private int idArbitre;  
    private String certification;  
    private int nbdematchfaitS;  
    private int nbdematchfaitD;  
    private String nom;  
    private String prenom;  
  
    public java.util.Collection<MatchDouble> matchDouble;  
    public java.util.Collection<Match> match;  
  
    public int getIdArbitre() {  
        return idArbitre;  
    }  
  
    public String getCertification() {  
        return certification;  
    }  
  
    /** @param newCertification */  
    public void setCertification(String newCertification) {  
        certification = newCertification;  
    }  
  
    public int getNbdematchfaitS() {  
        return nbdematchfaitS;  
    }  
  
    /** @param newNbdematchfaitS */  
    public void setNbdematchfaitS(int newNbdematchfaitS) {  
        nbdematchfaitS = newNbdematchfaitS;  
    }  
  
    public int getNbdematchfaitD() {  
        return nbdematchfaitD;  
    }  
}
```

```
/** @param newNbdematchfaitD */
public void setnbdematchfaitD(int newNbdematchfaitD) {
    nbdematchfaitD = newNbdematchfaitD;
}

public String getnom() {
    return nom;
}

/** @param newNom */
public void setnom(String newNom) {
    nom = newNom;
}

public String getprenom() {
    return prenom;
}

/** @param newPrenom */
public void setprenom(String newPrenom) {
    prenom = newPrenom;
}

public Arbitre() {
    // TODO: implement
}

/** @pdGenerated default getter */
public java.util.Collection<MatchDouble> getMatchDouble() {
    if (matchDouble == null)
        matchDouble = new java.util.HashSet<MatchDouble>();
    return matchDouble;
}

/** @pdGenerated default iterator getter */
public java.util.Iterator getIteratorMatchDouble() {
    if (matchDouble == null)
        matchDouble = new java.util.HashSet<MatchDouble>();
    return matchDouble.iterator();
}

/** @pdGenerated default setter
 * @param newMatchDouble */
public void setMatchDouble(java.util.Collection<MatchDouble> newMatchDouble) {
    removeAllMatchDouble();
    for (java.util.Iterator iter = newMatchDouble.iterator(); iter.hasNext();)
        addMatchDouble((MatchDouble)iter.next());
}

/** @pdGenerated default add
 * @param newMatchDouble */
public void addMatchDouble(MatchDouble newMatchDouble) {
    if (newMatchDouble == null)
        return;
    if (this.matchDouble == null)
        this.matchDouble = new java.util.HashSet<MatchDouble>();
    if (!this.matchDouble.contains(newMatchDouble))
```

```

    {
        this.matchDouble.add(newMatchDouble);
        newMatchDouble.setChaise(this);
    }
}

/** @pdGenerated default remove
 * @param oldMatchDouble */
public void removeMatchDouble(MatchDouble oldMatchDouble) {
    if (oldMatchDouble == null)
        return;
    if (this.matchDouble != null)
        if (this.matchDouble.contains(oldMatchDouble))
        {
            this.matchDouble.remove(oldMatchDouble);
            oldMatchDouble.setChaise((Arbitre)null);
        }
}

/** @pdGenerated default removeAll */
public void removeAllMatchDouble() {
    if (matchDouble != null)
    {
        MatchDouble oldMatchDouble;
        for (java.util.Iterator iter = getIteratorMatchDouble(); iter.hasNext();)
        {
            oldMatchDouble = (MatchDouble)iter.next();
            iter.remove();
            oldMatchDouble.setChaise((Arbitre)null);
        }
    }
}

/** @pdGenerated default getter */
public java.util.Collection<Match> getMatch() {
    if (match == null)
        match = new java.util.HashSet<Match>();
    return match;
}

/** @pdGenerated default iterator getter */
public java.util.Iterator getIteratorMatch() {
    if (match == null)
        match = new java.util.HashSet<Match>();
    return match.iterator();
}

/** @pdGenerated default setter
 * @param newMatch */
public void setMatch(java.util.Collection<Match> newMatch) {
    removeAllMatch();
    for (java.util.Iterator iter = newMatch.iterator(); iter.hasNext();)
        addMatch((Match)iter.next());
}

/** @pdGenerated default add
 * @param newMatch */
public void addMatch(Match newMatch) {
    if (newMatch == null)

```

```
        return;
    if (this.match == null)
        this.match = new java.util.HashSet<Match>();
    if (!this.match.contains(newMatch))
    {
        this.match.add(newMatch);
        newMatch.addLigne(this);
    }
}

/** @pdGenerated default remove
 * @param oldMatch */
public void removeMatch(Match oldMatch) {
    if (oldMatch == null)
        return;
    if (this.match != null)
        if (this.match.contains(oldMatch))
        {
            this.match.remove(oldMatch);
            oldMatch.removeLigne(this);
        }
}

/** @pdGenerated default removeAll */
public void removeAllMatch() {
    if (match != null)
    {
        Match oldMatch;
        for (java.util.Iterator iter = getIteratorMatch(); iter.hasNext();)
        {
            oldMatch = (Match)iter.next();
            iter.remove();
            oldMatch.removeLigne(this);
        }
    }
}
}
```

III.2.2 Package Hébergement – TypeChambre

```
/* *****  
 * Module:   TypeChambre.java  
 * Author:   swann  
 * Purpose:  Defines the Class TypeChambre  
 * ***** */  
  
package Hebergement;  
  
import java.util.*;  
  
public class TypeChambre {  
    private int idTypeChambre;  
    private String intitule;  
    private boolean suite;  
    private int nbLitsS;  
    private int nbLitsD;  
    private String description;  
    private String URL;  
    private int nbChambres;  
  
    public Hotel hotel;  
    public java.util.Collection<Chambre> chambre;  
    public java.util.Collection<Demande> demande;  
  
    public String getIntitule() {  
        return intitule;  
    }  
  
    /** @param newIntitule */  
    public void setIntitule(String newIntitule) {  
        intitule = newIntitule;  
    }  
  
    public boolean issuite() {  
        return suite;  
    }  
  
    /** @param newSuite */  
    public void setSuite(boolean newSuite) {  
        suite = newSuite;  
    }  
  
    public int getNbLitsS() {  
        return nbLitsS;  
    }  
  
    /** @param newNbLitsS */  
    public void setNbLitsS(int newNbLitsS) {  
        nbLitsS = newNbLitsS;  
    }  
  
    public int getNbLitsD() {  
        return nbLitsD;  
    }  
}
```

```
/** @param newNbLitsD */
public void setnbLitsD(int newNbLitsD) {
    nbLitsD = newNbLitsD;
}

public String getdescription() {
    return description;
}

/** @param newDescription */
public void setdescription(String newDescription) {
    description = newDescription;
}

public String getURL() {
    return URL;
}

/** @param newURL */
public void setURL(String newURL) {
    URL = newURL;
}

public int getnbChambres() {
    return nbChambres;
}

/** @param newNbChambres */
public void setnbChambres(int newNbChambres) {
    nbChambres = newNbChambres;
}

public int getidTypeChambre() {
    return idTypeChambre;
}

public TypeChambre() {
    // TODO: implement
}

/** @pdGenerated default parent getter */
public Hotel getHotel() {
    return hotel;
}

/** @pdGenerated default parent setter
 * @param newHotel */
public void setHotel(Hotel newHotel) {
    if (this.hotel == null || !this.hotel.equals(newHotel))
    {
        if (this.hotel != null)
        {
            Hotel oldHotel = this.hotel;
            this.hotel = null;
            oldHotel.removeTypeChambre(this);
        }
    }
}
```

```

        if (newHotel != null)
        {
            this.hotel = newHotel;
            this.hotel.addTypeChambre(this);
        }
    }

    /** @pdGenerated default getter */
    public java.util.Collection<Chambre> getChambre() {
        if (chambre == null)
            chambre = new java.util.HashSet<Chambre>();
        return chambre;
    }

    /** @pdGenerated default iterator getter */
    public java.util.Iterator getIteratorChambre() {
        if (chambre == null)
            chambre = new java.util.HashSet<Chambre>();
        return chambre.iterator();
    }

    /** @pdGenerated default setter
     * @param newChambre */
    public void setChambre(java.util.Collection<Chambre> newChambre) {
        removeAllChambre();
        for (java.util.Iterator iter = newChambre.iterator(); iter.hasNext(); )
            addChambre((Chambre)iter.next());
    }

    /** @pdGenerated default add
     * @param newChambre */
    public void addChambre(Chambre newChambre) {
        if (newChambre == null)
            return;
        if (this.chambre == null)
            this.chambre = new java.util.HashSet<Chambre>();
        if (!this.chambre.contains(newChambre))
        {
            this.chambre.add(newChambre);
            newChambre.setTypeChambre(this);
        }
    }

    /** @pdGenerated default remove
     * @param oldChambre */
    public void removeChambre(Chambre oldChambre) {
        if (oldChambre == null)
            return;
        if (this.chambre != null)
            if (this.chambre.contains(oldChambre))
            {
                this.chambre.remove(oldChambre);
                oldChambre.setTypeChambre((TypeChambre)null);
            }
    }

    /** @pdGenerated default removeAll */
    public void removeAllChambre() {

```

```

        if (chambre != null)
        {
            Chambre oldChambre;
            for (java.util.Iterator iter = getIteratorChambre(); iter.hasNext();)
            {
                oldChambre = (Chambre)iter.next();
                iter.remove();
                oldChambre.setTypeChambre((TypeChambre)null);
            }
        }
    }

    /** @pdGenerated default getter */
    public java.util.Collection<Demande> getDemande() {
        if (demande == null)
            demande = new java.util.HashSet<Demande>();
        return demande;
    }

    /** @pdGenerated default iterator getter */
    public java.util.Iterator getIteratorDemande() {
        if (demande == null)
            demande = new java.util.HashSet<Demande>();
        return demande.iterator();
    }

    /** @pdGenerated default setter
     * @param newDemande */
    public void setDemande(java.util.Collection<Demande> newDemande) {
        removeAllDemande();
        for (java.util.Iterator iter = newDemande.iterator(); iter.hasNext();)
            addDemande((Demande)iter.next());
    }

    /** @pdGenerated default add
     * @param newDemande */
    public void addDemande(Demande newDemande) {
        if (newDemande == null)
            return;
        if (this.demande == null)
            this.demande = new java.util.HashSet<Demande>();
        if (!this.demande.contains(newDemande))
        {
            this.demande.add(newDemande);
            newDemande.setTypeChambre(this);
        }
    }

    /** @pdGenerated default remove
     * @param oldDemande */
    public void removeDemande(Demande oldDemande) {
        if (oldDemande == null)
            return;
        if (this.demande != null)
            if (this.demande.contains(oldDemande))
            {
                this.demande.remove(oldDemande);
                oldDemande.setTypeChambre((TypeChambre)null);
            }
    }

```



```
}

/** @pdGenerated default removeAll */
public void removeAllDemande() {
    if (demande != null)
    {
        Demande oldDemande;
        for (java.util.Iterator iter = getIteratorDemande(); iter.hasNext();)
        {
            oldDemande = (Demande)iter.next();
            iter.remove();
            oldDemande.setTypeChambre((TypeChambre)null);
        }
    }
}

}
```

III.3 Scripts SQL générés

Voici quelques exemples de scripts SQL permettant de créer des tables MySQL. Ils ont été générées avec PowerAMC.

III.3.1 Package Planning – Court

```
/*=====*/
/* Table : court */
/*=====*/
create table court
(
  idcourt          int not null,
  nomcourt         varchar(254),
  estprincipal     bool,
  primary key (idcourt)
);
```

III.3.2 Package Hébergement – Demande

```
/*=====*/
/* Table : demande */
/*=====*/
create table demande
(
  iddemande        int not null,
  idhotel           int not null,
  idtypechambre     int not null,
  demandeur        int not null,
  etat             int,
  primary key (iddemande)
);
```