

Alary Philippe

Dal Christine

NARDELLA MARINE

VERHAEGHE JULIEN

Système

de réservations

pour un spectacle

PROJET PHP

2017-2018

Contents

[1 Etude préalable 3](#_Toc512695439)

[1.1 Enoncé du projet 3](#_Toc512695440)

[1.2 Analyse préliminaire 4](#_Toc512695441)

[1.2.1 Présentation du problème 4](#_Toc512695442)

[1.2.2 Délimitation du projet 4](#_Toc512695443)

[1.2.3 Recueil des besoins fonctionnels 4](#_Toc512695444)

[1.2.4 Recueil des besoins opérationnels 5](#_Toc512695445)

[1.2.5 Modélisation du contexte (acteurs et messages) 5](#_Toc512695446)

[1.2.6 Vocabulaire 6](#_Toc512695447)

[1.3 Capture des besoins fonctionnels 6](#_Toc512695448)

[1.3.1 Cas d’utilisation 6](#_Toc512695449)

[1.3.2 Modèle du domaine (entités-associations) 16](#_Toc512695450)

[1.4 Capture des besoins techniques 21](#_Toc512695451)

[1.4.1 Premières décisions techniques 21](#_Toc512695452)

[1.4.2 Division en couches 21](#_Toc512695453)

[1.4.3 Use cases techniques 21](#_Toc512695454)

[1.4.4 Draft de l’architecture technique 22](#_Toc512695455)

[1.4.5 Problèmes techniques à résoudre 22](#_Toc512695456)

[1.4.6 Cadre Fonctionnel et Technique du prototype 22](#_Toc512695457)

[1.5 Risques et remèdes 22](#_Toc512695458)

[2 Analyse détaillée 23](#_Toc512695459)

[2.1 Spécification détaillée des cas d’utilisation 23](#_Toc512695460)

[2.1.1 Description textuelle des cas d’utilisation 23](#_Toc512695461)

[1.1.2 Diagrammes de séquence système 36](#_Toc512695462)

[2.2 Classes d’analyse participantes 39](#_Toc512695463)

[2.2.1 Diagramme de classes 39](#_Toc512695464)

[2.2.2Modèle logique du domaine 42](#_Toc512695465)

[2.3 Modélisation de la navigation 43](#_Toc512695466)

[2.4 Conception objet préliminaire 46](#_Toc512695467)

[2.4.1 Diagrammes d’interaction (séquence et communication) 46](#_Toc512695468)

[2.4.2 Classes de conception préliminaire 49](#_Toc512695469)

[3 Conception objet détaillée 52](#_Toc512695470)

[3.1 Calendrier prévisionnel – état d’avancement 52](#_Toc512695471)

[3.2 Standards de développement 53](#_Toc512695472)

[3.2.1 Nom des variables, couleurs, procédures d’erreur, triggers 53](#_Toc512695473)

[3.2.2 Outils coopératifs utilisés 53](#_Toc512695474)

[3.3 Modèle de données 54](#_Toc512695475)

[3.3.1 Schéma physique de la BD 54](#_Toc512695476)

[3.3.2 Fichier de création bd (SQL) 55](#_Toc512695477)

[3.4 Architecture 63](#_Toc512695478)

[3.4.1 Schéma d’architecture physique 63](#_Toc512695479)

[3.4.2 Schéma d’architecture software 63](#_Toc512695480)

[3.4.3 Organisation des répertoires du site (MVC) 63](#_Toc512695481)

[3.5 Conception détaillée de chaque Use Case 64](#_Toc512695482)

[3.5.1 Diagramme de classes de conception 64](#_Toc512695483)

[3.5.2 Fonctions utilisées 67](#_Toc512695484)

[3.5.3 Diagrammes d’activité 71](#_Toc512695485)

[3.5.4 jeu d’essai niveau1 71](#_Toc512695486)

[4 Annexe 72](#_Toc512695487)

[4.1 Interview 72](#_Toc512695488)

[4.2 Annexe 2 : liste des réservations 73](#_Toc512695489)

[4.3 Annexe 3 : plan de la salle 74](#_Toc512695490)

[4.4 Annexe 4 : liste triée des réservations 75](#_Toc512695491)

[4.5 Annexe 5 : compte-rendu de réunions 76](#_Toc512695492)

# 1 Etude préalable

## Enoncé du projet

Une troupe de théâtre wallon de Mellet permet aux spectateurs de demander des réservations pour un spectacle donné à une certaine date. La troupe dispose d’une salle pouvant contenir actuellement 130 personnes.

Dans cette salle, les chaises sont disposées par rangée. Les rangées sont identifiées par une lettre allant de A à J et peuvent contenir maximum 14 chaises.

Le système actuel se présente de la façon suivante : les personnes souhaitant assister à la représentation d’un spectacle peuvent contacter le responsable des réservations soit par téléphone, soit par mail. Le responsable dispose, par date de spectacle, de deux documents préimprimés :

* Une liste des réservations (annexe 2)
* Un plan de la salle (annexe 3)

Celui-ci, lorsqu’il reçoit une demande de réservation, indique sur la liste à la première ligne disponible le nom de la personne, le nombre de place, le numéro des chaises.

Pour cette dernière information, le responsable dispose du plan de la salle par date de spectacle. Lors d’une réservation, le responsable indique sur le plan pour la date correspondante le numéro de la réservation sur les places à réserver.

La veille du spectacle, le responsable des réservations remet à la personne qui s’occupera des entrées les deux documents. Cette personne encodera alors la liste des réservations dans un document Excel et l’imprimera en deux exemplaires. (annexe 4)

Le plan de salle sera utilisé pour repérer les places disponibles afin de les attribuer aux personnes se présentant sans réservation.

La troupe de théâtre désire informatiser le système de réservation pour ses représentations. Le système à mettre en place doit permettre de gérer les spectacles, les représentations, la disposition de la salle, ainsi que les réservations. Afin d’éviter que n’importe qui puisse accéder aux fonctionnalités, l’utilisateur devra tout d’abord s’authentifier lors de son arrivée sur l’application. Un « super-user » pourra définir les droits pour chaque rang d’utilisateur pouvant accéder à l’application.

L’utilisateur pouvant effectuer les réservations devra pouvoir le faire de la manière la plus simple possible. Cela signifie que l’utilisateur doit pouvoir facilement ajouter une réservation pour un spectateur à une certaine représentation en renseignant les sièges à réserver. Il doit aussi pouvoir avoir une vue d’ensemble sur les sièges encore disponibles lorsqu’il effectue une réservation.

L’utilisateur pouvant gérer les spectacles doit pouvoir ajouter facilement un nouveau spectacle dans le système. Il doit également pouvoir gérer la disposition de la salle pour une représentation donnée, c’est-à-dire qu’il ne doit pas pouvoir modifier la disposition pour une date à laquelle il n’y aura aucune représentation.

Le système doit aussi permettre d’obtenir la liste des réservations pour une date de représentation donnée sous format csv ainsi que d’envoyer un mail aux spectateurs possédant une adresse mail les informations d’un spectacle à venir ainsi que ses représentations.

## Analyse préliminaire

### 1.2.1 Présentation du problème

Une troupe de théâtre wallon de Mellet désire informatiser la structure de gestion des réservations pour leurs diverses représentations.

La troupe de théâtre wallon de Mellet désire mettre en place un site web de gestion

* Un système de réservation des places
* Consultation des plans de salles et des listes
* Extraction des données

### 1.2.2 Délimitation du projet

Le système ne doit gérer que l’aspect gestion. L’aspect financier n’est pas pris en compte par l’application. Le système ne doit gérer qu’une salle de représentation.

### 1.2.3 Recueil des besoins fonctionnels

**Salle**

Cette rubrique permettra de rechercher la disposition d’une salle à une date spécifiée. Le plan reprendra la disposition des chaises pour une date spécifiée. Il y sera également possible de copier le plan de salle d’une date vers une autre date.

**Spectacle**

Cette rubrique permettra d’ajouter un spectacle, une représentation et un personnel. Aussi cette rubrique permettra d’exporter le toutes boîtes.

**Spectateurs**

Cette rubrique permettra de modifier les informations d’un spectateur, d’envoyer un mail aux personnes ayant une adresse mail et souhaitant assister à un spectacle contenant un fichier attaché reprenant les informations du spectacle ou un autre document. Aussi cette rubrique permettra d’exporter les étiquettes.

**Réservations**

Cette rubrique contiendra le plan de la salle à une date spécifiée et permettra à la personne habilitée de réaliser les réservations, d’effectuer celles-ci en ayant une vue sur les sièges disponibles, non disponibles, occupés ou réservés et d’indiquer une éventuelle remarque concernant la réservation. Si la personne n’existe pas dans la base de données, il sera possible de l’ajouter.

**Liste**

Cette rubrique permettra de consulter soit la liste des réservations en fonction d’une date de représentation. Il sera également possible de rechercher une réservation en fonction d’une d’ une date de représentation ainsi que du nom des spectateurs ou du numéro de réservation. Cette rubrique permettra aussi de rechercher un spectateur en renseignant son nom et son prénom.

**Extraction**

Cette rubrique permettra d’exporter sous PDF soit la liste des réservations pour un spectacle à une date de représentation, soit la liste des spectateurs.

**Administration**

Cette rubrique permet à un « super user » de définir les droits de chaque utilisateur pouvant utiliser l’application.

### 1.2.4 Recueil des besoins opérationnels

**Sécurité**

L’ensemble des données devra être sauvegardé de manière régulière de façon à pouvoir, en cas de problème, restaurer une situation la plus récente possible. Les mots de passe seront chiffrés.

**Confidentialité**

Chaque utilisateur ne doit avoir accès qu’aux informations auxquelles il a droit.

**Volumes de données**

La salle ne peut accueillir que maximum 130 personnes par représentation et en conséquence, chaque représentation aura un maximum de 130 réservations. On estime le nombre de représentations d'une pièce entre 6 et 12. Une estimation rapide donnera environ 1560 réservations à conserver pour les différentes représentations d'une seule pièce.

On suppose que lorsqu'une pièce ne sera plus représentée, les informations concernant cette dernière sont archivées.

**Accès**

Le système doit être accessible 24h/24 depuis Internet. Aucun logiciel particulier ne doit être nécessaire pour les utilisateurs, hormis un navigateur internet (y compris un navigateur de téléphone portable).

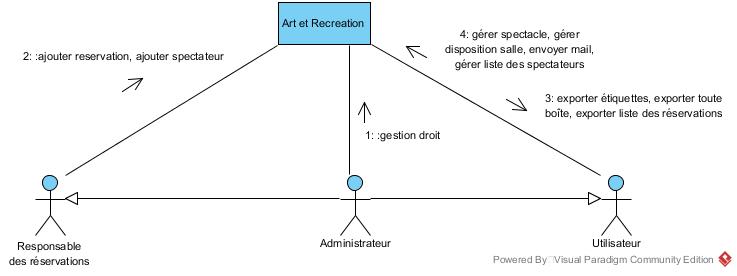
### 1.2.5 Modélisation du contexte (acteurs et messages)

Les acteurs suivants ont été identifiés :

- Administrateur : Il s’occupe de la gestion des droits, il peut aussi effectuer les mêmes opérations que le responsable des réservations.

- Responsable des réservations : Il peut ajouter un spectateur, effectuer les réservations et envoyer un mail aux spectateurs possédant une adresse mail.

- Utilisateur : Il peut gérer les spectacles, gérer les représentations, gérer la disposition de la salle, exporter la liste des réservations, exporter le toutes boîtes, exporter les étiquettes, gérer la liste des spectateurs et envoyer un mail aux spectateurs possédant une adresse mail.



### 1.2.6 Vocabulaire

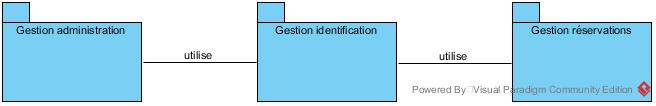
Un spectateur est un client qui vient assister à une représentation.

Un acteur est une personne faisant partie du personnel travaillant sur un spectacle.

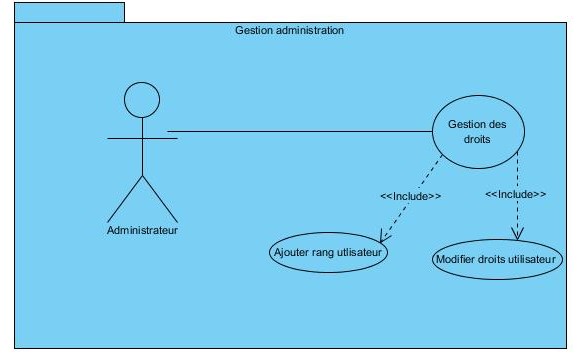
## Capture des besoins fonctionnels

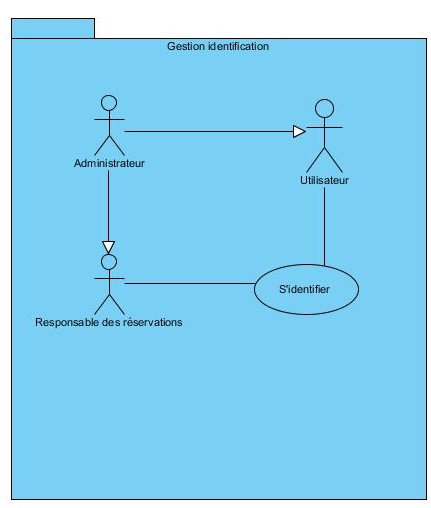
### Cas d’utilisation

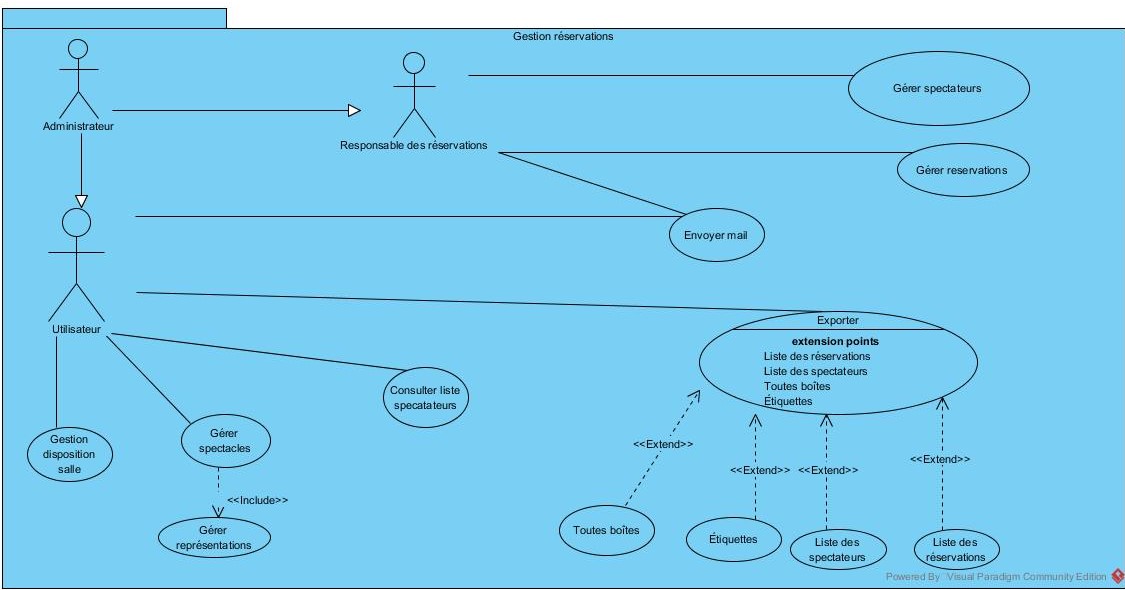
**a) Diagramme de packages**



**b) Diagramme de use case**

****

****

****

**c) Description succincte des cas d’utilisation**

Package gestion administration

*Ajouter rang utilisateur* : l’administrateur ajoutera de nouveaux utilisateurs.

*Modifier droits utilisateur*: l’administrateur pourra ajouter ou retirer des droits à un utilisateur.

Package gestion identification

*S’identifier* : l’utilisateur devra entrer un login et un mot de passe afin d’accéder aux fonctionnalités de l’application.

Package gestion réservations

*Gérer réservations* : la personne responsable des réservations, ajoutera la personne en renseignant son nom, sa place et éventuellement un commentaire. Si la personne n’a jamais effectué de réservation, il faudra d’abord l’ajouter dans la base de données.

*Gérer spectateurs :* l’utilisateur peut modifier les informations d’un spectateur, il faudra renseigner son nom, son prénom.

*Gérer spectacles :* l’utilisateur peut ajouter un spectacle en renseignant le titre du spectacle et un résumé du spectacle. L’utilisateur peut ensuite ajouter un personnage en renseignant son nom et son prénom. Il peut aussi ajouter des acteurs en renseignant leur nom, leur prénom et leur fonction.

*Gérer représentations :* l’utilisateur peut ajouter des représentations à un spectacle en renseignant la date et l’heure de la représentation.

*Gestion disposition salle :* en fonction que l’utilisateur aura choisi une date de représentation, il aura la disposition de la salle. Il pourra de plus ajouter des sièges ou en retirer et copier ce plan de salle pour une autre date.

*Liste des spectateurs :* l’utilisateur peut exporter à la demande une liste des spectateurs pour une date donnée.

*Liste des réservations :* l’utilisateur peut exporter à la demande une liste de ses réservations pour une date donnée.

*Toutes boîtes :* l’utilisateur peut exporter à la demande une toutes boites qui reprend toutes les informations du spectacle : date, lieu, date, distribution, mise en scène, régie et décors, aide-mémoire et salle.

*Etiquettes :*l’utilisateur peut exporter à la demande des étiquettes qui reprennent le nom et l’adresse postale du spectateur.

*Consulter liste spectateurs :* l’utilisateur peut consulter la liste des spectateurs en cochant la case liste des spectateurs.

*Envoyer mail :* l’utilisateur peut envoyer un mail à ceux qui possèdent une adresse mail pour envoyer les informations importantes sur le spectacle et la réservation ou faire de la publicité.

**d) Besoins en IHM (écrans)**

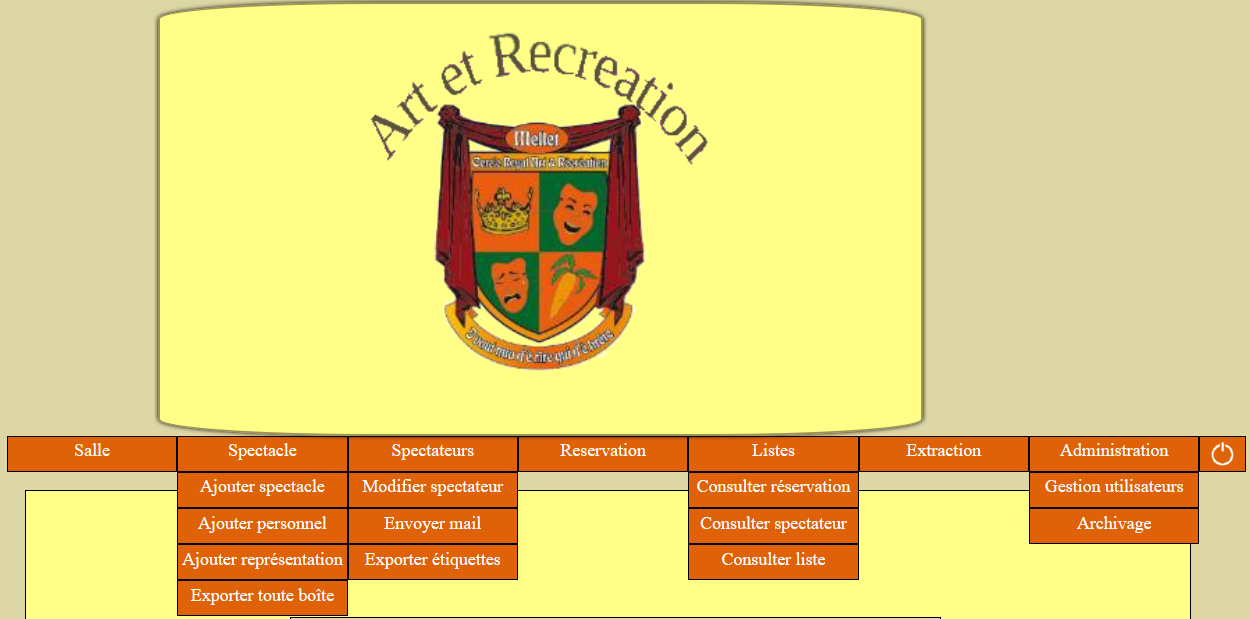
Logo et menu

Figure1

Cette image montre le logo et le menu ainsi que ses sous-menus. Nous avons effectivement repris les couleurs du logo pour rester en accord avec le thème du théâtre. Le menu permet de naviguer à travers chacune des rubriques de l’application.

Connexion



Figure2

Cette image représente la page de connexion à l’application. L’utilisateur doit rentrer son login et son mot de passe pour accéder aux fonctionnalités de l’application.

Accueil

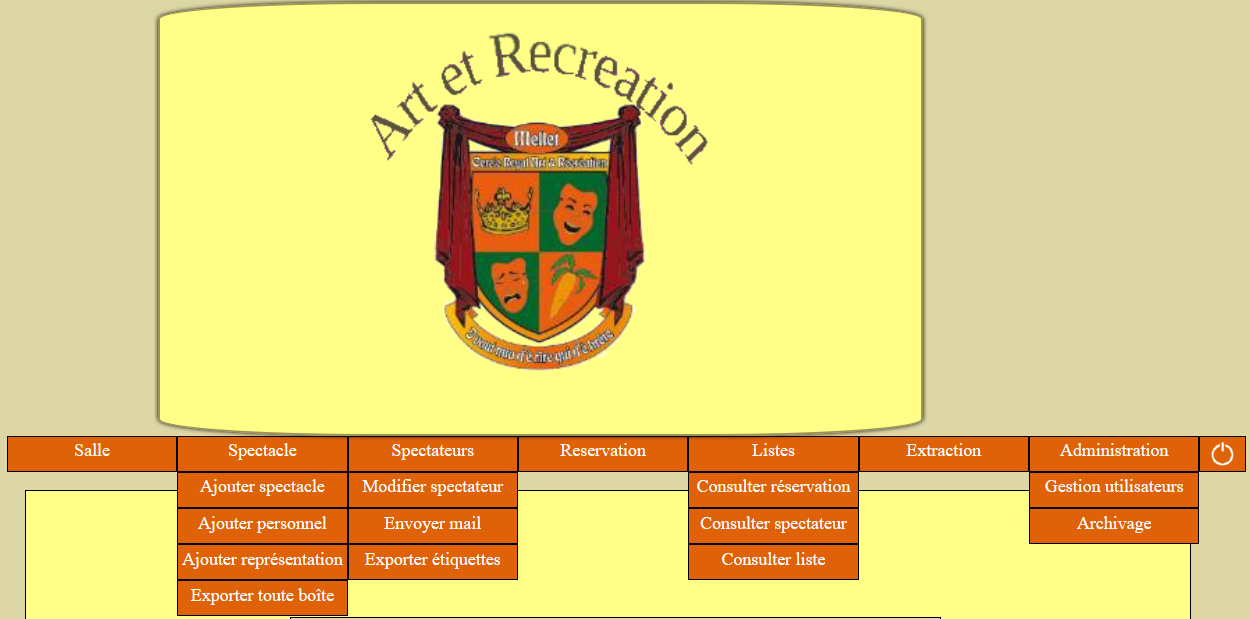


Figure3

Tout utilisateur (n’importe lequel), une fois qu’il sera connecté à l’application arrivera sur la page d’accueil. Sur celle-ci, on y retrouve une brève description qui explique chaque rubrique de l’application.

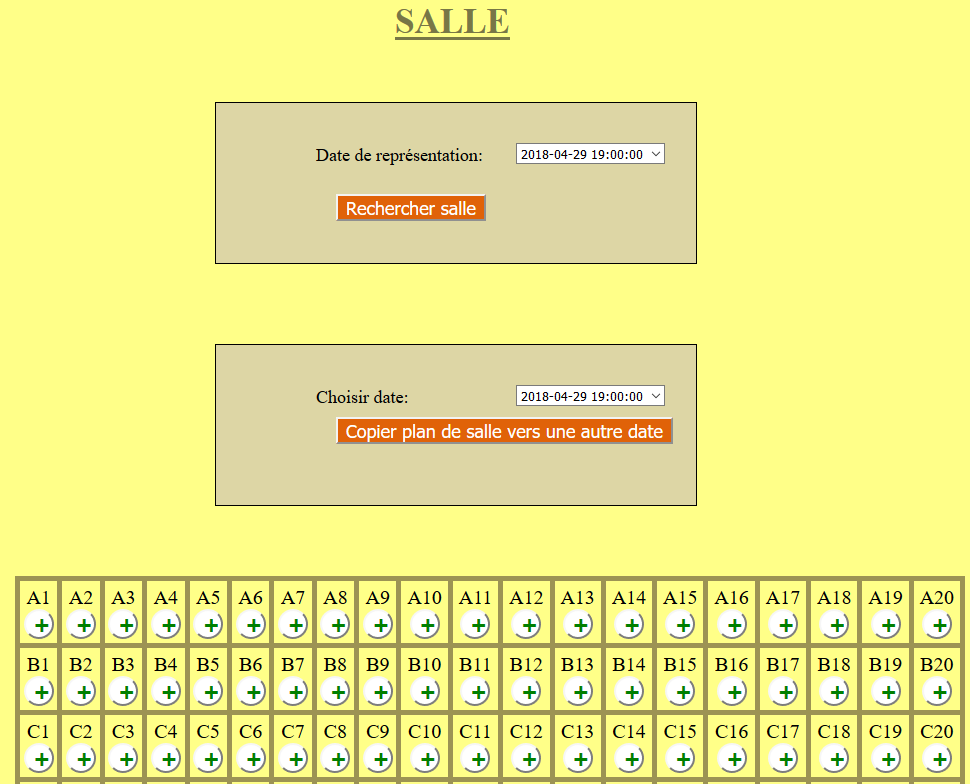
Salle

Figure 4

En arrivant sur cette page, seul le premier formulaire est visible. L’utilisateur devra renseigner la date de la représentation dont il veut afficher le plan de salle. Après avoir cliqué sur le bouton « rechercher salle », le second formulaire ainsi que le plan de la salle à la date renseignée s’afficheront.

Le plan de la salle affichera la position de chaque siège et pour chaque siège, il sera possible soit de l’enlever si c’est le bouton « - » qui est affiché, soit de l’ajouter si c’est le bouton « + » qui est affiché. Il est possible de copier le plan de salle de la date sélectionnée vers une autre date grâce au second formulaire.

Spectacle

Ajouter spectacle



Figure 5.1

Ce sous-menu permet d’ajouter un spectacle dans la base de données en renseignant le titre du spectacle, ainsi qu’un résumé du spectacle.

Ajouter personnel



Figure5.2

Ce sous-menu permet d’ajouter un membre du personnel dans la base de données en renseignant son nom, son prénom, sa fonction et choisir le titre du spectacle.



Figure 5.3

Si le membre du personnel est un acteur, il devient possible de renseigner le nom et le prénom du personnage qu’il interprète.

Ajouter représentation

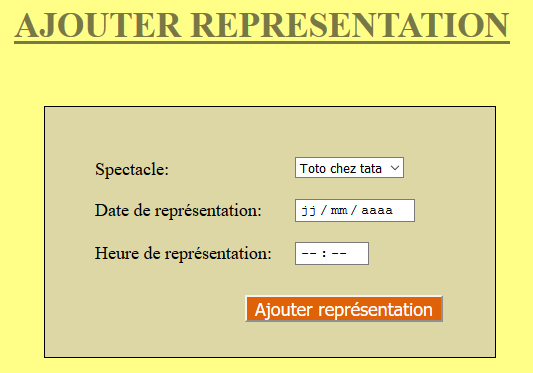


Figure 5.4

Ce sous-menu permet d’ajouter une représentation dans la base de données en renseignant le titre du spectacle, la date et l’heure de représentation.

Spectateur

Modifier un spectateur

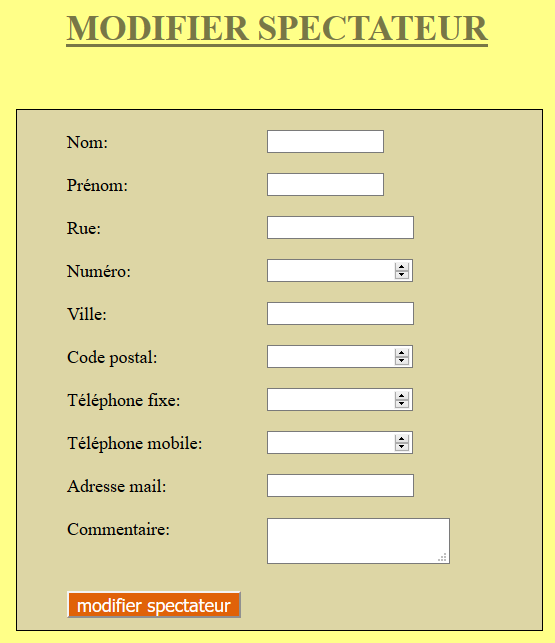


Figure 6.1

Ce sous-menu permet d’ajouter un nouveau spectateur dans la base de données en renseignant ses informations.

Envoyer un mail



Figure 6.2

Ce sous-menu permet d’envoyer un mail à tous les spectateurs de la base de données possédant une adresse mail. L’utilisateur peut décider d’envoyer le toutes boîtes et / ou un autre fichier en pièce jointe (affiche).

Exporter des étiquettes



Figure 6.3

Ce sous-menu permet d’exporter des étiquettes renseignant l’adresse postale des spectateurs. Une case à cocher permet à l’utilisateur de décider d’exporter ou non également des étiquettes pour les spectateurs possédant une adresse mail.

Réservation

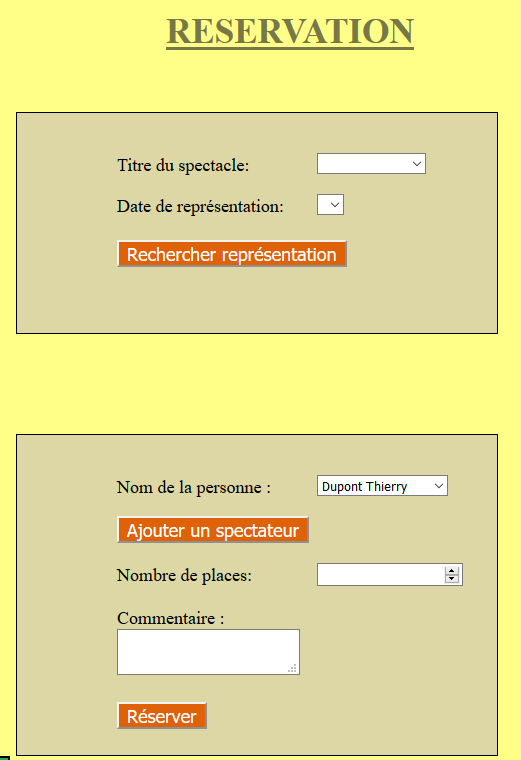


Figure 7.1

Cette page permet d’enregistrer de nouvelles réservations pour un certain spectacle représenté à une certaine date à une certaine heure.

En arrivant sur cette page, l’utilisateur à sa disposition un formulaire dans lequel il devra renseigner le titre du spectacle et la date de représentation.

Lorsqu’il cliquera sur le bouton « Rechercher représentation», un second formulaire ainsi que le plan de la salle pour la date donnée apparaîtra.



Figure 7.2

Lorsqu’il cliquera sur le bouton « Ajouter un spectateur », un formulaire apparaîtra pour ajouter un spectateur dans la base de données.

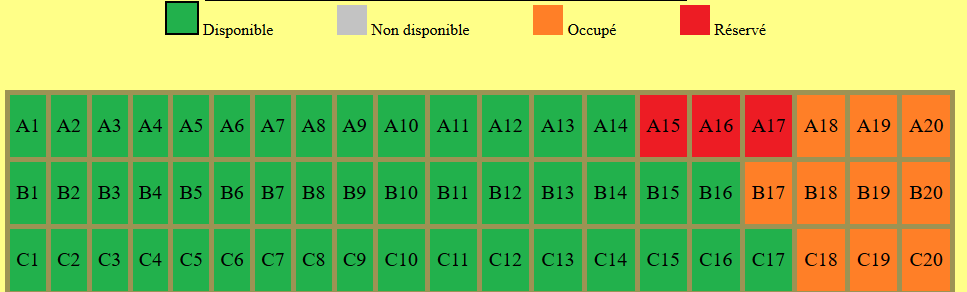


Figure 7.2

Le plan de la salle informera l’utilisateur de la répartition des sièges ainsi que de leur état (disponible, non disponible, réservé, occupé) grâce à un code couleur.

En s’aidant du plan de la salle, l’utilisateur pourra effectuer des réservations en renseignant le nom de la personne ayant demandé la réservation, les places à réserver pour cette personne ainsi qu’un éventuel commentaire.

Listes

Consulter réservation

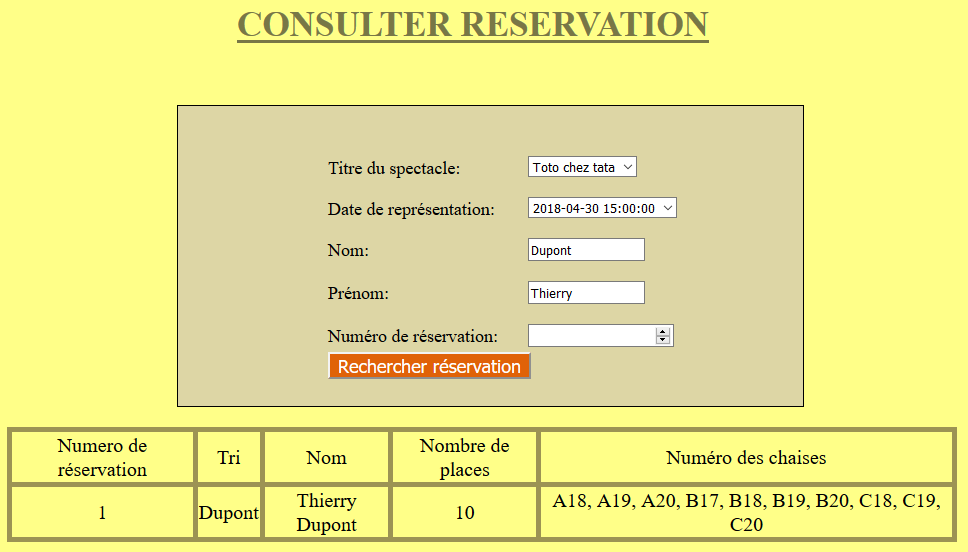


Figure 8.1

Cette page permet d’afficher une réservation soit en renseignant le titre du spectacle, la date de représentation, nom, prénom ou soit en renseignant tous les champs du formulaire.

Consulter spectateur



Figure 8.2

Cette page permet de rechercher un spectateur par nom et par son prénom.

Consulter liste

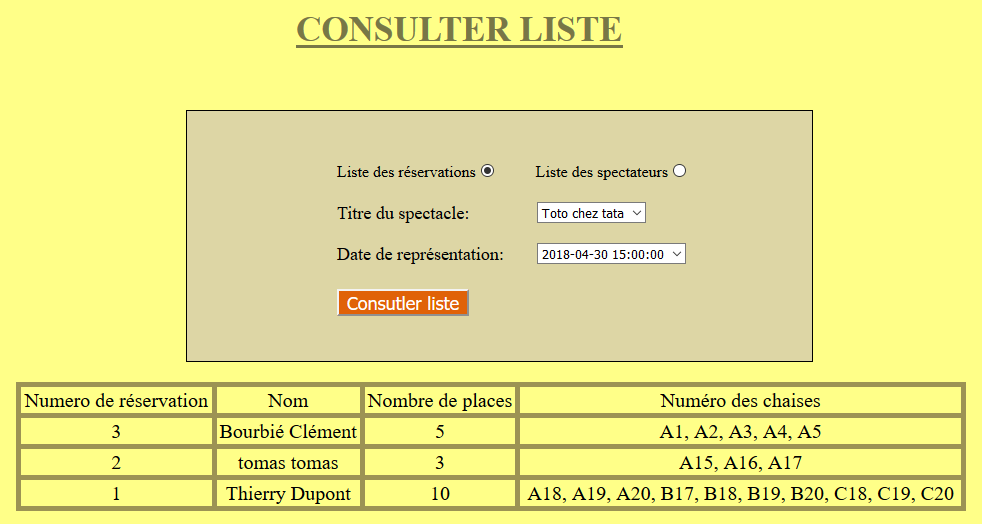


Figure 8.2

Cette page permet d’afficher la liste des réservations en renseignant le titre du spectacle et la date de représentation.



Figure 8.3

Cette page permet d’afficher la liste des spectateurs.

Extraction



Figure 9.1

Cette page permet d’extraire sous format PDF la liste des réservations effectuées en renseignant le titre du spectacle et la date de représentation.

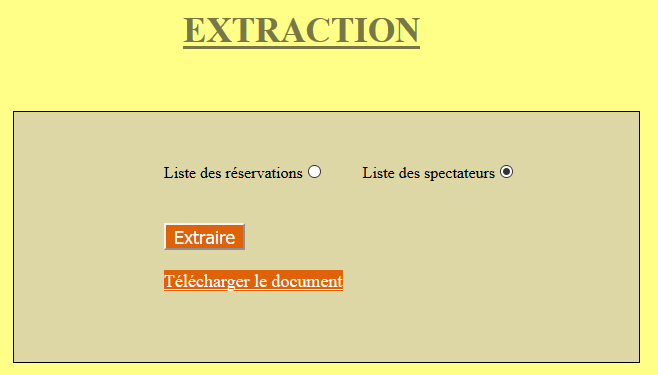


Figure 9.2

Cette page permet d’extraire sous format PDF la liste spectateurs.

Administration

Gestion utilisateurs

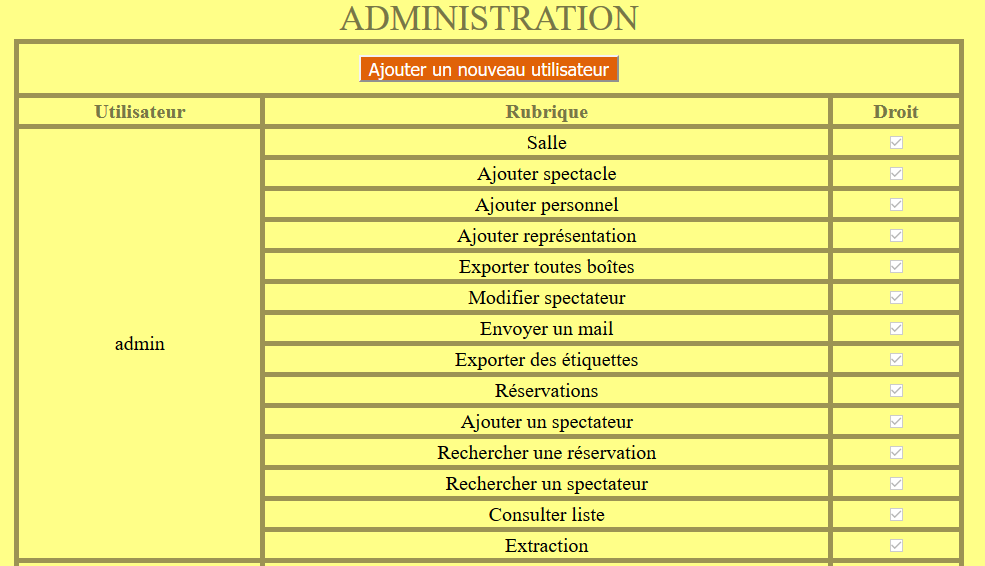


Figure 10.1

Cette page permet de gérer les différents rangs d’utilisateur ainsi que leurs droits.

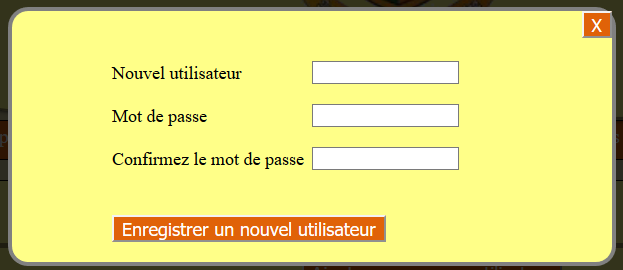


Figure 10.2

Ce formulaire permet d’ajouter un nouvel utilisateur.

Archivage



Figure 10.2

Ce formulaire permet d’archiver les données d’un spectacle.

**e) Classement des cas d’utilisation**

|  |  |  |
| --- | --- | --- |
| Cas d’utilisation | Priorité | Risque |
| **Package administration** |  |  |
| Gestion droits |  |  |
| Ajouter rang utilisateur | Haute | Bas |
| Modifier droit utilisateur | Haute | Moyen |
| **Package identification** |  |  |
| S’identifier | Haute | Bas |
| **Package réservation** |  |  |
| Gérer réservations | Haute | Haut |
| Gérer spectateurs | Haute | Moyen |
| Gérer spectacles | Haute | Moyen |
| Gérer représentations | Haute | Basse |
| Gestion disposition salle | Haute | Haut |
| Consulter liste  spectateurs | Moyen | Bas |
| Exporter | Moyen | Moyen |
| Etiquettes | Bas | Moyen |
| Toutes boîtes | Bas | Haut |
| Liste des spectateurs | Moyen | Bas |
| Liste des réservations | Moyen | Bas |
| Envoyer mail | Bas | Bas |

### Modèle du domaine (entités-associations)

1. **Schéma E/A**

****

1. Dictionnaire des données et contraintes d’intégrité

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Dictionary report

Project THEATRE

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

ADMINISTRATION

Login : varchar(20) login de l’utilisateur

Password : varchar(20) mot de passe de l’utilisateur

Salle : varchar (20) droit disposition de la salle

ajouterSpectacle : char(1) droit ajouter un spectacle

ajouterPersonnel : char(1) droit d’ajouter un personnel

ajouterRepresentation : char(1) droit d’ajouter une représentation

exporterToutesBoites : char(1) droit d’exporter toutes boîtes

modifierSpectateur : char(1) droit de modifier un spectateur

envoyerMail : char(1) droit d’envoyer un mail aux spectateurs

exporterEtiquettes : char(1) droit d’exporter des étiquettes

reservations : char(1) droit de faire des réservations

ajouterSpectateur : char(1) droit d’ajouter un spectateur

rechercherReservation : char(1) droit de rechercher une réservation du spectateur

rechercherSpectateur : char(1) droit de rechercher un spectateur

consulterListe : char(1) droit de consulter liste spectateurs, liste réservations)

extraction : char(1) droit d’extraction des listes(spectateurs et réservations)

|  |
| --- |
| id: login |

CHAISE

Id :char (3) caractère correspondant au numéro de ligne et de colonne

tri :num(1) permet de trier convenablement les chaises s’incrémente automatiquement

etat: char (1) état de la place

|  |
| --- |
| id: disposer.REPRESENTATION,  role: [0-1] in concerner  role: [1-1] in disposer |

PERSONNEL

id: int (11) numéro du personnel s’incrémente automatiquement

nom: varchar (20) nom de la personne

prenom: varchar (20) prénom de la personne

fonction: varchar (20) fonction de la personne

|  |
| --- |
| id: id  role: [0-N] in organiser |

REPRESENTATION

date: date date de la représentation

heure : time heure de la représentation

|  |
| --- |
| id: date,heure  role: [0-N] in disposer  role: [1-1] in présenter |

RESERVATIONS

id: num (3) numéro de réservation s’incrémente automatiquement

nbSièges: num (3) nombre de place réservée

remarque: varchar (150) commentaire

|  |
| --- |
| id: id  role: [0-N] in concerner  role: [1-1] in porter sur  role: [1-1] in reserver |

SPECTACLE

titre: varchar (20) titre du spectacle

personnages[0-N]: varchar (20) nom des personnages

resume: varchar (150) résumé du spectacle

|  |
| --- |
| id: titre  role: [0-N] in organiser  role: [0-N] in porter sur  role: [0-N] in présenter |

SPECTATEURS

id: num (3) numéro du spectateur s’incrémente automatiquement

nom: varchar (20) nom du spectateur

prenom: varchar (20) prénom du spectateur

adresse: compound (208) adresse du spectateur

rue: char (150) rue de l’habitation du spectateur

numero: num (4) numéro de l’habitation du spectateur

localite: char (50) localite du spectateur

codePostal: num(4) code postal du spectateur

tel\_fixe: num (20) téléphone fixe du spectateur

tel\_mobile: num (20) téléphone mobile du spectateur

adresse\_mail: varchar (100) adresse mail du spectateur

commentaire: varchar (150) remarque

|  |
| --- |
| id: id  role: [0-N] in reserver |

**concerner** (

[0-1] : CHAISE

[0-N] : RESERVATIONS )

**disposer** (

[0-N] : REPRESENTATION

[1-1] : CHAISE )

**organiser** (

[0-N] : SPECTACLE

[0-N] : PERSONNEL )

**porter sur** (

[1-1] : RESERVATIONS

[0-N] : SPECTACLE )

**présenter** (

[0-N] : SPECTACLE

[1-1] : REPRESENTATION )

**reserver** (

[0-N] : SPECTATEURS

[1-1] : RESERVATIONS )

**Contraintes d’intégrité**

1. La réservation est effectuée seulement par la personne habilitée
2. Les champs tel\_fixe, tel\_mobile et adresse\_mail doivent être des champs uniques
3. état : d =disponible

n = non disponible

o = occupé

r = réservé

1. salle : Z = aucun droit, M = droit maintenance
2. ajouterSpectacle : Z = aucun droit, M = droit maintenance
3. ajouterPersonnel : Z = aucun droit, M = droit maintenance
4. ajouterRepresentation : Z = aucun droit, M = droit maintenance
5. exporterToutesBoites : Z = aucun droit, M = droit maintenance
6. modifierSpectateur : Z = aucun droit, M = droit maintenance
7. envoyerMail : Z = aucun droit, M = droit maintenance
8. exporterEtiquettes : Z = aucun droit, M = droit maintenance
9. reservations : Z = aucun droit, M = droit maintenance
10. ajouterSpectateur : Z = aucun droit, M = droit maintenance
11. rechercherReservation : Z = aucun droit, M = droit maintenance
12. rechercherSpectateur : Z = aucun droit, M = droit maintenance
13. consulterListe : Z = aucun droit, M = droit maintenance
14. extraction : Z = aucun droit, M = droit maintenance

## Capture des besoins techniques

### Premières décisions techniques

Nous suivons une architecture web classique, avec un client (navigateur internet) qui accède à un serveur http.

Côté serveur, le langage de programmation utilisé est le PHP 7 en orienté objet. Nous n’utiliserons pas de framework.

Côté client, les langages utilisés seront l’HTML 5, le CSS 3. Du javascript sera également présent, notamment avec la librairie JQuery et la JQuery UI, en utilisant AJAX au maximum, afin de rendre l’application plus réactive et augmenter le confort d’utilisation.

La persistance des données sera gérée par une base de données relationnelle, MariaDB 10.2.

### 1.4.2 Division en couches

Notre application sera divisée en plusieurs couches : la couche présentation, la couche logique, la couche métier et la couche persistance.

La couche présentation aura pour but d’afficher l’IHM à l’utilisateur.

La couche logique fera le lien entre la couche présentation et la couche métier.

La couche métier contiendra toutes les données.

La couche persistance permettra l’accès à la base de données.

### 1.4.3 Use cases techniques

Gestion de l’identification

*But*: restriction des droits.

*Description :* Pour utiliser le système, un utilisateur doit s'authentifier avec un login et un mot de passe.

Gestion des droits

*But :* définition des utilisateurs et des droits de chaque utilisateur.

*Description :* L’administrateur peut définir les droits pour chaque rang d’utilisateur en cochant ou décochant la case correspondante. Il peut aussi ajouter un nouveau rang d’utilisateur.

### 1.4.4 Draft de l’architecture technique

Nous utiliserons le design patterns suivant:

* MVC : il divise le projet en structure.

### 1.4.5 Problèmes techniques à résoudre

Voici les problèmes techniques et les questions que nous nous posons auquel devrait pouvoir répondre notre prototype :

* Utilisation « inconditionnelle » du javascript et AJAX.
* Utilisation des design patterns et PDO.
* Authentification.

### 1.4.6 Cadre Fonctionnel et Technique du prototype

Notre prototype permettra à un utilisateur d’effectuer des réservations, lister les réservations selon les critères de recherche demandés. Afin de déceler d’éventuelles complications, il devra être capable d’exporter au format PDF la liste des réservations pour une certaine date.

Notre prototype servira ensuite de base sur laquelle développer la version complète de chacune des fonctionnalités demandées.

## Risques et remèdes

Accessibilité depuis navigateurs mobiles.

Utilisation de PHP OO. Lire la doc PHP.

# Analyse détaillée

## Spécification détaillée des cas d’utilisation

### Description textuelle des cas d’utilisation

|  |
| --- |
| **S’identifier**  **Acteur principal**: l’utilisateur (peu importe le rang).  **Objectifs**: lorsque l’utilisateur désire accéder aux fonctionnalités de l’application, il doit renseigner son login et son mot de passe. |
| **Préconditions**: néant.  **Postconditions**: l’utilisateur est connecté.  **Scénario nominal**   1. L’utilisateur rentre son login puis son mot de passe (ou l’inverse). 2. Le système vérifie dans la base de données s’il y a un login dans la base de données qui correspond à celui renseigné par l’utilisateur. 3. Le système vérifie ensuite si le mot de passe rentré est exact. 4. L’utilisateur a accès aux fonctionnalités autorisées par son rang d’utilisateur.   **Alternatives**  2a Le login est incorrect.   1. Le système affiche un message d’erreur (« Login incorrect »). 2. Le cas d’utilisation reprend à l’étape 1 du scénario nominal.   3a Le mot de passe est incorrect.   1. Le système affiche un message d’erreur (« Mot de passe incorrect »). 2. Le cas d’utilisation reprend à l’étape 1 du scénario nominal.   **Exigences supplémentaires**  L’utilisateur ne doit pas avoir accès aux fonctionnalités qui ne lui sont pas autorisées.  Référence Figure 2 p11 |

|  |
| --- |
| **Ajouter rang utilisateur**  **Acteur principal**: l’administrateur.  **Objectifs**: permettre à l’administrateur d’ajouter un nouveau d’utilisateur. |
| **Préconditions**: l’utilisateur doit être de rang administrateur.  **Postconditions**: le nouveau utilisateur doit avoir été ajouté.  **Scénario nominal**   1. L’administrateur clique sur le bouton « Ajouter un nouveau utilisateur ». 2. L’administrateur indique un nom pour le nouveau utilisateur.   **Alternatives**  2a Le nom renseigné est déjà pris par un autre utilisateur.   1. Le système indique de façon visible et précise que le nom est déjà utilisé. 2. Le cas d’utilisation reprend à l’étape 2 du scénario nominal.   **Exigences supplémentaires**  Néant.  Référence Figure 10.2 p24 |

|  |
| --- |
| **Modifier droit utilisateur**  **Acteur principal**: l’administrateur.  **Objectifs** : permettre à l’administrateur de modifier les droits d’utilisateurs déjà existants. |
| **Préconditions**:   * L’utilisateur doit être de rang administrateur. * L’utilisateur dont il faut modifier les droits doit déjà exister.   **Postconditions**: La modification doit être effective.  **Scénario nominal**   1. L’administrateur coche ou décoche la case correspondant au droit qu’il désire modifier.   **Alternatives**  Néant.  **Exigences supplémentaires**  Néant.  Référence Figure 10.1 p24 |

|  |
| --- |
| **Gérer spectacles**  **Acteur principal**: l’utilisateur (n’importe lequel ayant le droit de gérer les spectacles).  **Objectifs**: pouvoir ajouter un spectacle dans la base de données. |
| **Préconditions**: l’utilisateur doit être connecté à un rang bénéficiant du droit correspondant.  **Postconditions**: le spectacle ainsi que ses informations doivent figurer dans la base de données.  **Scénario nominal**   1. L’utilisateur renseigne les informations du spectacle. 2. L’utilisateur clique sur le bouton « Ajouter spectacle ». 3. Le spectacle est ajouté dans la base de données. 4. L’utilisateur peut ensuite ajouter le personnel travaillant sur ce spectacle et pour les acteurs, le personnage interprété.   **Alternatives**  3a Un spectacle porte déjà le titre renseigné.   1. Le système affiche un message d’erreur (« Le titre existe déjà »). 2. Le cas d’utilisation reprend à l’étape 1 du scénario nominal.   **Exigences supplémentaires**  Néant.  Référence Figure 5.1 p14, 5.2-5.3 p15 |

|  |
| --- |
| **Gérer représentations**  **Acteur principal**: l’utilisateur (n’importe lequel ayant le droit de gérer les représentations).  **Objectifs**: pouvoir ajouter une date de représentation pour un spectacle dans la base de données. |
| **Préconditions**: l’utilisateur doit être connecté à un rang bénéficiant du droit correspondant.  **Postconditions**: une ou plusieurs dates de réservation doivent être ajoutées dans la base de données.  **Scénario nominal**   1. L’utilisateur choisit un spectacle dans la liste déroulante. 2. L’utilisateur choisit une date. 3. L’utilisateur clique sur le bouton « Ajouter représentation ».   **Alternatives**  3a Une représentation est déjà programmée à cette date.  1 Le système affiche un message d’erreur («Une représentation existe déjà à cette date »).  2 Le cas d’utilisation reprend à l’étape 2 du scénario nominal.  **Exigences supplémentaires**  Néant.  Référence Figure 5.4 p16 |

|  |
| --- |
| **Gérer salle**  **Acteur principal**: l’utilisateur (n’importe lequel ayant le droit de gérer les représentations).  **Objectifs**: pouvoir modifier la disposition de la salle pour une date donnée. |
| **Préconditions**: l’utilisateur doit être connecté à un rang bénéficiant du droit correspondant.  **Postconditions**: les modifications doivent avoir été enregistrées.  **Scénario nominal**   1. L’utilisateur indique la date souhaitée. 2. L’utilisateur modifie à sa guise la disposition des sièges et peut copier cette disposition vers une autre date qu’il renseignera.   **Alternatives**  1a Il n’y a aucune représentation à la date indiquée.  1 Le système affichera un message d’erreur (« Il n’y a aucune représentation à la date indiquée »).  2 Le cas d’utilisation reprend à l’étape 1 du scénario nominal.  2a salle  1 le système affichera un message d’erreur (« Il n’y a aucune représentation à la date indiquée »)  2 Le cas d’utilisation reprend à l’étape 2 du scénario nominal  **Exigences supplémentaires**  Néant.  Référence Figure 4 p13 |

|  |
| --- |
| **Gérer réservations**  **Acteur principal**: le responsable des réservations.  **Objectifs**: pouvoir effectuer des réservations. |
| **Préconditions**: l’utilisateur doit être connecté à un rang bénéficiant du droit correspondant.  **Postconditions** : la réservation doit être stockée en base de données.  **Scénario nominal**   1. L’utilisateur indique la date du spectacle. 2. L’utilisateur clique sur le bouton « Rechercher représentation ». 3. L’utilisateur renseigne le nom de la personne ayant demandé la réservation, les places à réserver et éventuellement un commentaire. 4. L’utilisateur clique sur le bouton « Réserver ».   **Alternatives**  1a Il n’y a aucune représentation à la date indiquée.   1. Le système affiche un message d’erreur (« Aucune représentation à la date indiquée. ». 2. Le cas d’utilisation reprend à l’étape 1 du scénario nominal.   3a Le spectateur n’existe pas dans la base de données.   1. Le système affiche un message d’erreur (« Spectateur inexistant. ») et propose à l’utilisateur de le rediriger vers la page d’ajout de spectateurs. 2. Une fois le spectateur ajouté dans la base de données, l’utilisateur retourne sur la page de gestion des réservations. 3. Le cas d’utilisation reprend à l’étape 1 du scénario nominal.   **Exigences supplémentaires**  Néant.  Référence Figure 7.1 p19 et 7.2 p20 |

|  |
| --- |
| **Consulter liste des spectateurs**  **Acteur principal**: l’utilisateur (n’importe lequel ayant le droit de consulter cette liste).  **Objectifs**: pouvoir consulter la liste des spectateurs. |
| **Préconditions**: l’utilisateur doit être connecté à un rang bénéficiant du droit correspondant.  **Postconditions**: néant.  **Scénario nominal**   1. L’utilisateur coche la case <<Liste des spectateurs>>. 2. L’utilisateur clique sur le bouton « Consulter liste». 3. L’utilisateur peut consulter la liste.   **Alternatives**  Néant.  **Exigences supplémentaires**  Néant.  Référence Figure 8.1 p22 |

|  |
| --- |
| **liste des réservations**  **Acteur principal**: l’utilisateur (n’importe lequel ayant le droit d’exporter).  **Objectifs**: permettre à l’utilisateur d’obtenir la liste des réservations pour une date donnée. |
| **Préconditions**: l’utilisateur doit être connecté à un rang bénéficiant du droit correspondant.  **Postconditions** : l’utilisateur doit avoir la liste des réservations sous format Excel.  **Scénario nominal**   1. Il sélectionne le bouton << liste des réservations >> 2. L’utilisateur peut exporter la liste.   **Alternatives**  1a Il n’y a aucune représentation à la date sélectionnée.   1. Le système affiche un message d’erreur (« Aucune représentation à la date sélectionnée »). 2. Le cas d’utilisation reprend à l’étape 2 du scénario nominal.   **Exigences supplémentaires**  Néant.  Référence 9.1 p23 |

|  |
| --- |
| **liste des spectateurs**  **Acteur principal**: l’utilisateur (n’importe lequel ayant le droit d’exporter).  **Objectifs**: permettre à l’utilisateur d’obtenir la liste des spectateurs pour une date donnée. |
| **Préconditions**: l’utilisateur doit être connecté à un rang bénéficiant du droit correspondant.  **Postconditions** : l’utilisateur doit avoir la liste des spectateurs sous format Excel.  **Scénario nominal**   1. Il sélectionne le bouton << liste des spectateurs >> 2. L’utilisateur peut exporter la liste.   **Alternatives**  Néant.  **Exigences supplémentaires**  Néant.  Référence Figure 9.2 p23 |

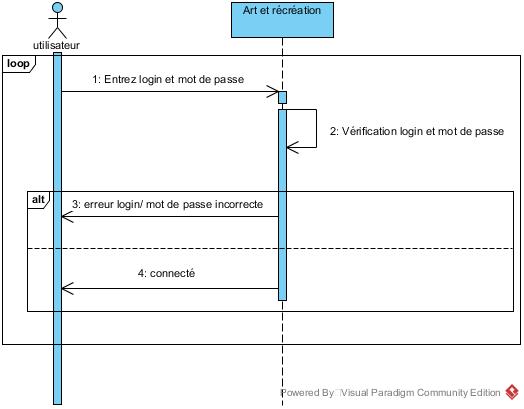
|  |
| --- |
| **Envoyer mail**  **Acteur principal**: l’utilisateur (n’importe lequel ayant le droit d’envoyer les mails).  **Objectifs**: pouvoir envoyer un mail contenant les informations d’un spectacle à tous les spectateurs possédant une adresse mail. |
| **Préconditions**: l’utilisateur doit être connecté à un rang bénéficiant du droit correspondant.  **Postconditions**: chaque spectateur possédant une adresse mail doit avoir reçu le mail.  **Scénario nominal**   1. L’utilisateur coche ou non la case pour ajouter le toutes boîtes en pièce jointe. 2. L’utilisateur ajoute ou non un fichier supplémentaire à ajouter en pièce jointe. 3. L’utilisateur clique sur le bouton « Envoyer ».   **Alternatives**  Néant.  **Exigences supplémentaires**  Néant.  Référence Figure 6.2 p18 |

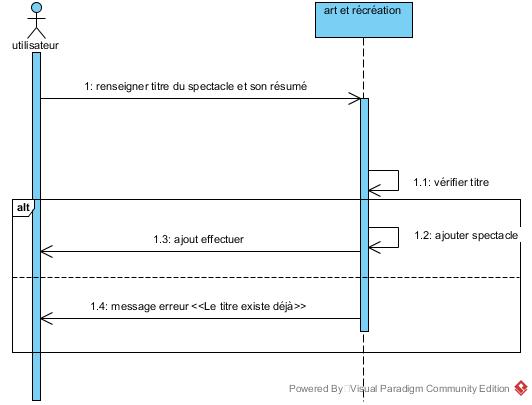
|  |
| --- |
| **Toutes boîtes**  **Acteur principal**: l’utilisateur (n’importe lequel ayant le droit d’exporter le toutes boîtes).  **Objectifs**: pouvoir exporter un toutes boîtes contenant les informations d’un spectacle. |
| **Préconditions**: l’utilisateur doit être connecté à un rang bénéficiant du droit correspondant.  **Postconditions**: l’utilisateur doit avoir le toutes boîtes.  **Scénario nominal**   1. L’utilisateur choisit le spectacle dont il désire obtenir le toutes boîtes. 2. L’utilisateur clique sur le bouton « Exporter toutes boîtes ». 3. L’utilisateur enregistre le fichier sur son appareil.   **Alternatives**  Néant.  **Exigences supplémentaires**  Néant. |

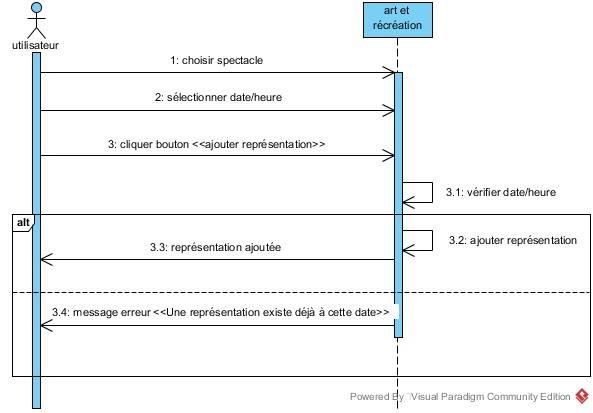
|  |
| --- |
| **Etiquettes**  **Acteur principal**: l’utilisateur (n’importe lequel ayant le droit d’exporter les étiquettes).  **Objectifs**: pouvoir obtenir des étiquettes contenant l’adresse postale des spectateurs. |
| **Préconditions**: l’utilisateur doit être connecté à un rang bénéficiant du droit correspondant.  **Postconditions**: l’utilisateur a les étiquettes.  **Scénario nominal**   1. L’utilisateur coche ou non la case pour exporter également les étiquettes des spectateurs possédant une adresse mail. 2. L’utilisateur clique sur le bouton « Exporter etiquettes ». 3. L’utilisateur enregistre le fichier sur son appareil.   **Alternatives**  Néant.  **Exigences supplémentaires**  Néant.  Référence Figure 6.3 p18 |

### Diagrammes de séquence système

S’identifier

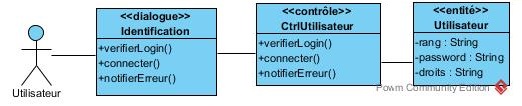


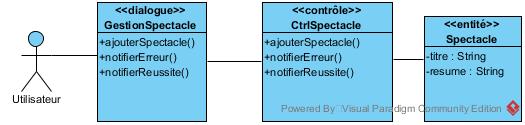
Gérer spectacle

Gérer représentation

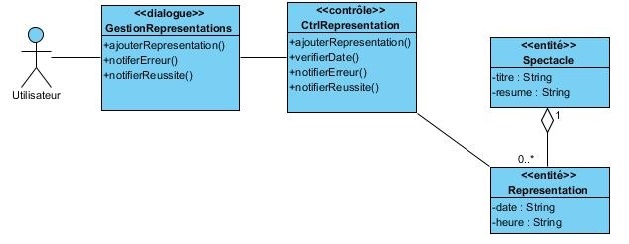
## Classes d’analyse participantes

### Diagramme de classes

S’identifier

Gérer spectacle

Gérer représentation

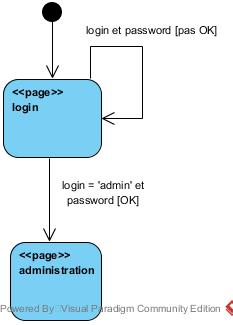


### 2.2.2Modèle logique du domaine

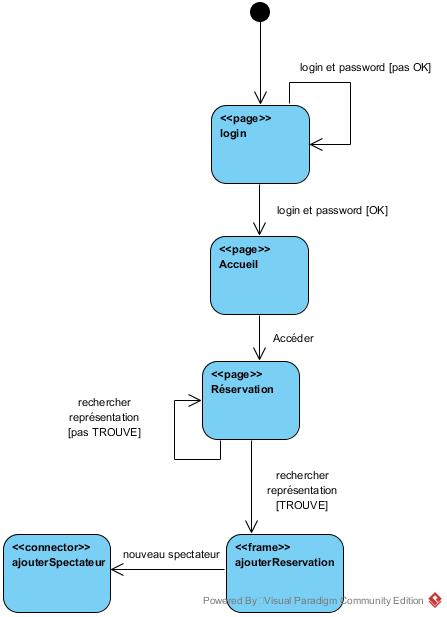


## Modélisation de la navigation

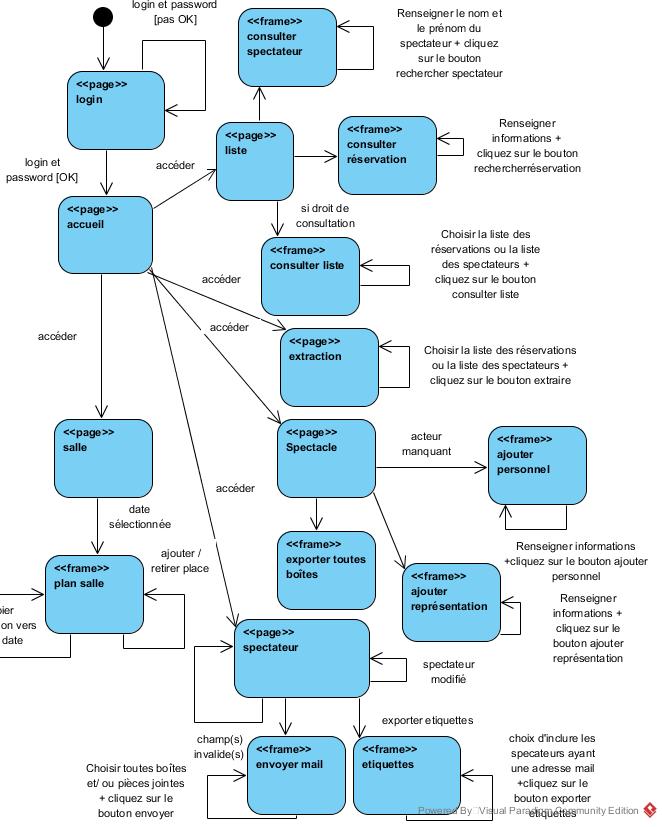
Administrateur



Responsable des réservations

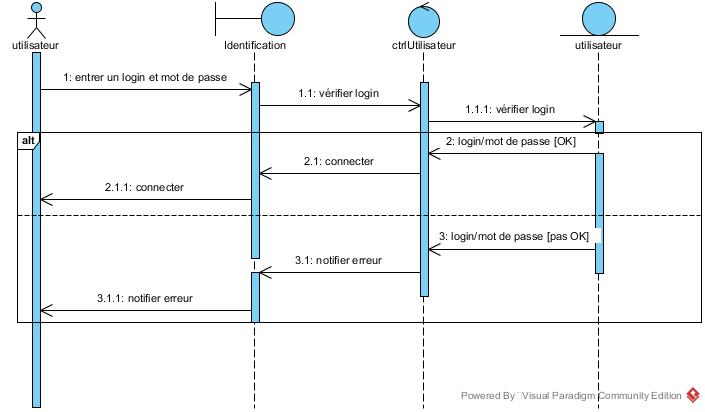


Utilisateur

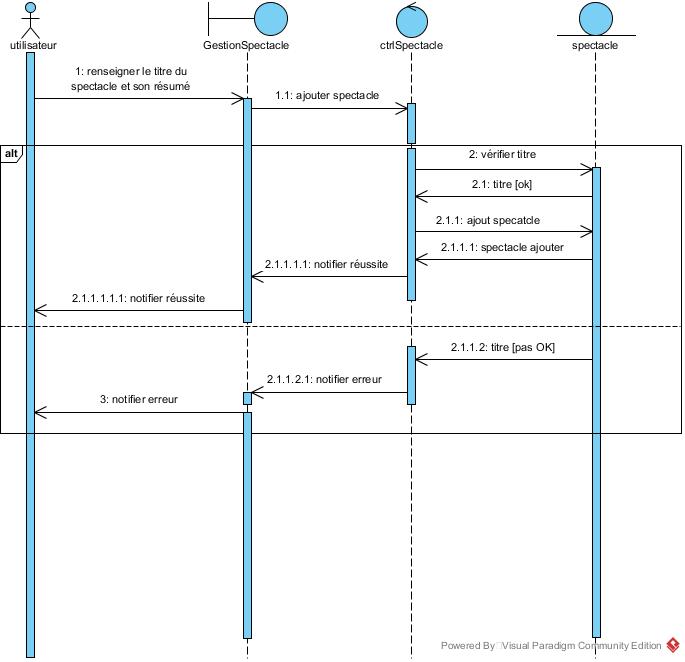


## Conception objet préliminaire

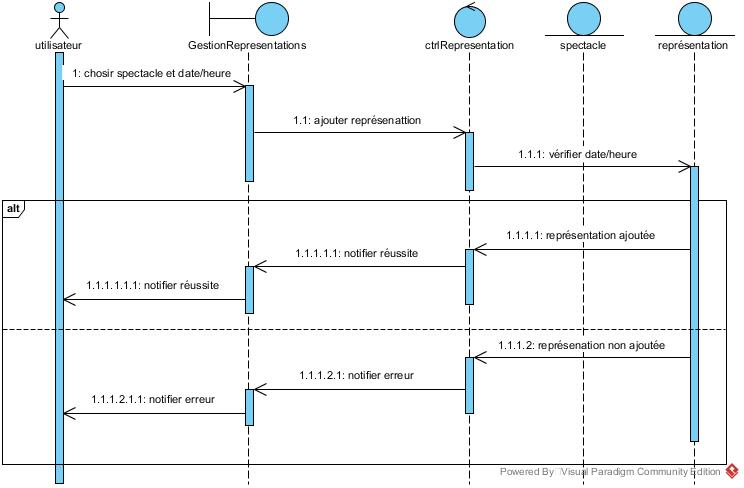
### Diagrammes d’interaction (séquence et communication)

S’identifier

Gérer spectacle

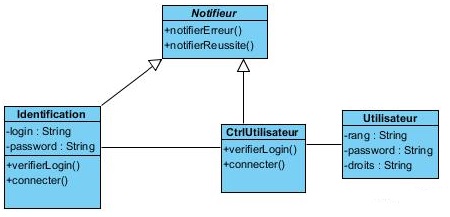


Gérer représentation

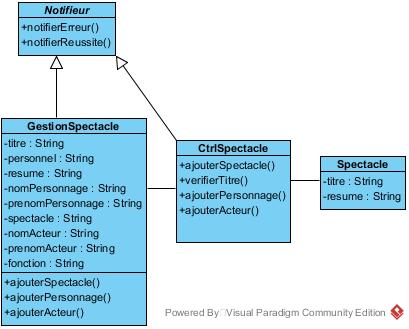


### Classes de conception préliminaire

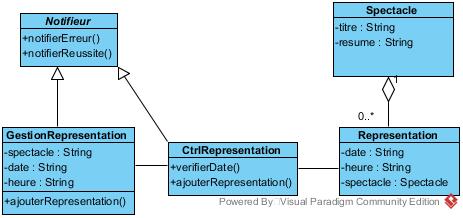
S’identifier



Gérer spectacle



Gérer représentation



# Conception objet détaillée

## Calendrier prévisionnel – état d’avancement

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Cas d’utilisation** | **Priorité** | **Nombre d’heures prévues** | **Nombre d’heures réelles** | **Programmeur** | **Date prévue** | **Date effective** |
| **Package administration** |  |  |  |  |  |  |
| Ajouter rang utilisateur | 1 | 2 | 2 | Marine | 14/03/2018 | 13/03/2018 |
| Modifier droit utilisateur | 1 | 4 | 10 | Julien  Marine | 14/03/2018 | 12/03/2018 |
| **Package identification** |  |  |  |  |  |  |
| S’identifier | 1 | 1 | 2 | Julien | 14/03/2018 | 10/03/2018 |
| **Package réservation** |  |  |  |  |  |  |
| Gérer réservations | 1 | 8 | 13 | Marine  Julien | 14/03/2018 | 26/03/2018 |
| Gérer spectateurs | 1 | 4 | 6 | Marine | 14/03/2018 | 24/03/2018 |
| Gérer spectacles | 1 | 4 | 6 | Julien | 14/03/2018 | 24/03/2018 |
| Gérer représentations | 1 | 3 | 2 | Marine | 20/03/2018 | 23/03/2018 |
| Gestion disposition salle | 1 | 15 | 10 | Julien | 20/03/2018 | 25/03/2018 |
| Consulter liste  spectateurs | 2 | 4 |  | Marine | 29/04/2018 |  |
| Etiquettes | 3 | 15 |  | Julien | 10/05/2018 |  |
| Toutes boîtes | 3 | 20 |  | Marine | 10/05/2018 |  |
| Liste des spectateurs | 2 | 10 |  | Marine | 15/04/2018 |  |
| Liste des réservations | 2 | 12 |  | Julien | 15/04/2018 |  |
| Envoyer mail | 3 | 2 |  | Julien | 29/04/2018 |  |

## Standards de développement

### Nom des variables, couleurs, procédures d’erreur, triggers

* Le nom de chaque méthode et de chaque variable sera en lower camel case.
* Les méthodes accédant à la base de données s’appelleront :
* « get… » si elles effectuent une sélection dans la base de données
* « ajouter… » si elles effectuent une insertion de données
* « modifier… » si elles modifient une valeur existante.
* « delete… » si elles suppriment une valeur existante.

### 3.2.2 Outils coopératifs utilisés

* Git avec l’interface « SourceTree » : partager les sources de l’application à développer

## Modèle de données

### Schéma physique de la BD

### Fichier de création bd (SQL)

-- phpMyAdmin SQL Dump

-- version 4.7.4

-- https://www.phpmyadmin.net/

--

-- Hôte : 127.0.0.1

-- Généré le : sam. 28 avr. 2018 à 15:41

-- Version du serveur : 10.1.28-MariaDB

-- Version de PHP : 7.1.11

SET SQL\_MODE = "NO\_AUTO\_VALUE\_ON\_ZERO";

SET AUTOCOMMIT = 0;

START TRANSACTION;

SET time\_zone = "+00:00";

/\*!40101 SET @OLD\_CHARACTER\_SET\_CLIENT=@@CHARACTER\_SET\_CLIENT \*/;

/\*!40101 SET @OLD\_CHARACTER\_SET\_RESULTS=@@CHARACTER\_SET\_RESULTS \*/;

/\*!40101 SET @OLD\_COLLATION\_CONNECTION=@@COLLATION\_CONNECTION \*/;

/\*!40101 SET NAMES utf8mb4 \*/;

--

-- Base de données : `theatre`

--

DELIMITER $$

--

-- Procédures

--

CREATE DEFINER=`root`@`localhost` PROCEDURE `getReservations` (`p\_date` DATE, `p\_heure` TIME) BEGIN

DECLARE currentNbSieges, currentResId int;

DECLARE currentNom, currentPrenom varchar(20);

DECLARE chaise varchar(1500);

DECLARE done int;

DECLARE nom, prenom varchar(20);

DECLARE rId, nbSieges int;

DECLARE cId varchar(1500);

DECLARE selReservations CURSOR FOR SELECT s.nom, s.prenom, r.id as resId, r.nbSieges, c.id as cId

from spectateurs as s

INNER JOIN reservations as r ON s.id=r.SpecId

INNER JOIN chaise as c ON c.ResId=r.id

WHERE date=p\_date AND heure=p\_heure

ORDER BY s.nom, r.id, c.tri;

declare continue handler for not found set done=1;

create TEMPORARY table tableReservations

(

nom varchar(20),

prenom varchar(20),

rId int,

nbSieges int,

cId varchar(1500)

);

set done = 0;

OPEN selReservations;

FETCH NEXT FROM selReservations

INTO nom, prenom, rId, nbSieges, cId;

main\_loop: LOOP

if done = 1

then

leave main\_loop;

end if;

set currentResId = rId;

set currentNom = nom;

set currentPrenom = prenom;

set currentNbSieges = nbSieges;

set chaise = cId;

chaise\_loop: LOOP

FETCH NEXT FROM selReservations

INTO nom, prenom, rId, nbSieges, cId;

if done = 1 OR rId != currentResId

then

leave chaise\_loop;

end if;

set chaise = concat(chaise, ", ", cId);

end loop chaise\_loop;

insert into tableReservations(nom, prenom, rId, nbSieges, cId) VALUES(

currentNom, currentPrenom, currentResId, currentNbSieges, chaise);

end loop main\_loop;

CLOSE selReservations;

select \* from tableReservations;

END$$

DELIMITER ;

-- --------------------------------------------------------

--

-- Structure de la table `administration`

--

CREATE TABLE `administration` (

`login` varchar(20) NOT NULL,

`password` varchar(50) NOT NULL,

`salle` varchar(20) NOT NULL DEFAULT 'Z',

`ajouterSpectacle` char(1) NOT NULL DEFAULT 'Z',

`ajouterPersonnel` char(1) NOT NULL DEFAULT 'Z',

`ajouterRepresentation` char(1) NOT NULL DEFAULT 'Z',

`exporterToutesBoites` char(1) NOT NULL DEFAULT 'Z',

`modifierSpectateur` char(1) NOT NULL DEFAULT 'Z',

`envoyerMail` char(1) NOT NULL DEFAULT 'Z',

`exporterEtiquettes` char(1) NOT NULL DEFAULT 'Z',

`reservations` char(1) NOT NULL DEFAULT 'Z',

`ajouterSpectateur` char(1) NOT NULL DEFAULT 'Z',

`rechercherReservation` char(1) NOT NULL DEFAULT 'Z',

`rechercherSpectateur` char(1) NOT NULL DEFAULT 'Z',

`consulterListe` char(1) NOT NULL DEFAULT 'Z',

`extraction` char(1) NOT NULL DEFAULT 'Z'

) ENGINE=InnoDB DEFAULT CHARSET=utf8;

--

-- Déchargement des données de la table `administration`

--

INSERT INTO `administration` (`login`, `password`, `salle`, `ajouterSpectacle`, `ajouterPersonnel`, `ajouterRepresentation`, `exporterToutesBoites`, `modifierSpectateur`, `envoyerMail`, `exporterEtiquettes`, `reservations`, `ajouterSpectateur`, `rechercherReservation`, `rechercherSpectateur`, `consulterListe`, `extraction`) VALUES

('admin', '\*A4B6157319038724E3560894F7F932C8886EBFCF', 'M', 'M', 'M', 'M', 'M', 'M', 'M', 'M', 'M', 'M', 'M', 'M', 'M', 'M');

-- --------------------------------------------------------

--

-- Structure de la table `adresse`

--

CREATE TABLE `adresse` (

`id` int(11) NOT NULL,

`rue` char(150) NOT NULL,

`numero` varchar(10) NOT NULL,

`localite` char(50) NOT NULL,

`codePostal` int(11) NOT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8;

-- --------------------------------------------------------

--

-- Structure de la table `chaise`

--

CREATE TABLE `chaise` (

`tri` int(11) NOT NULL,

`date` date NOT NULL,

`heure` time NOT NULL,

`id` char(3) NOT NULL,

`etat` char(1) NOT NULL DEFAULT 'N',

`ResId` int(11) DEFAULT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8;

-- --------------------------------------------------------

--

-- Structure de la table `organiser`

--

CREATE TABLE `organiser` (

`id` int(11) NOT NULL,

`titre` varchar(20) NOT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8;

-- --------------------------------------------------------

--

-- Structure de la table `personnages`

--

CREATE TABLE `personnages` (

`titre` varchar(20) NOT NULL,

`nom` varchar(20) NOT NULL,

`id` int(11) NOT NULL,

`prenom` varchar(20) NOT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8;

-- --------------------------------------------------------

--

-- Structure de la table `personnel`

--

CREATE TABLE `personnel` (

`id` int(11) NOT NULL,

`nom` varchar(20) NOT NULL,

`prenom` varchar(15) NOT NULL,

`fonction` varchar(20) NOT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8;

-- --------------------------------------------------------

--

-- Structure de la table `representation`

--

CREATE TABLE `representation` (

`date` date NOT NULL,

`heure` time NOT NULL,

`nbrLigne` int(11) NOT NULL DEFAULT '10',

`nbrColonne` int(11) NOT NULL DEFAULT '14',

`titre` varchar(20) NOT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8;

-- --------------------------------------------------------

--

-- Structure de la table `reservations`

--

CREATE TABLE `reservations` (

`id` int(11) NOT NULL,

`nbSieges` int(11) NOT NULL,

`remarque` varchar(150) NOT NULL,

`SpecId` int(11) NOT NULL,

`titre` varchar(20) NOT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8;

-- -------------------------------------------------------

--

-- Structure de la table `spectacle`

--

CREATE TABLE `spectacle` (

`titre` varchar(20) NOT NULL,

`resume` varchar(150) NOT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8;

-- --------------------------------------------------------

--

-- Structure de la table `spectateurs`

--

CREATE TABLE `spectateurs` (

`id` int(11) NOT NULL,

`nom` varchar(20) NOT NULL,

`prenom` varchar(20) NOT NULL,

`telFixe` bigint(20) DEFAULT NULL,

`telMobile` bigint(20) DEFAULT NULL,

`adresseMail` varchar(100) DEFAULT NULL,

`commentaire` varchar(150) DEFAULT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8;

--

-- Index pour les tables déchargées

--

--

-- Index pour la table `administration`

--

ALTER TABLE `administration`

ADD PRIMARY KEY (`login`);

--

-- Index pour la table `adresse`

--

ALTER TABLE `adresse`

ADD PRIMARY KEY (`id`);

--

-- Index pour la table `chaise`

--

ALTER TABLE `chaise`

ADD PRIMARY KEY (`date`,`heure`,`id`),

ADD KEY `FKconcerner` (`ResId`),

ADD KEY `IND\_tri` (`tri`);

--

-- Index pour la table `organiser`

--

ALTER TABLE `organiser`

ADD PRIMARY KEY (`id`,`titre`),

ADD KEY `FKorg\_PER` (`id`),

ADD KEY `FKorg\_SPE` (`titre`);

--

-- Index pour la table `personnages`

--

ALTER TABLE `personnages`

ADD PRIMARY KEY (`titre`,`id`) USING BTREE,

ADD UNIQUE KEY `FKjouer\_par\_ID` (`id`);

--

-- Index pour la table `personnel`

--

ALTER TABLE `personnel`

ADD PRIMARY KEY (`id`);

--

-- Index pour la table `representation`

--

ALTER TABLE `representation`

ADD PRIMARY KEY (`date`,`heure`) USING BTREE,

ADD KEY `FKpresenter` (`titre`);

--

-- Index pour la table `reservations`

--

ALTER TABLE `reservations`

ADD PRIMARY KEY (`id`),

ADD KEY `FKreserver` (`SpecId`),

ADD KEY `FKporter\_sur` (`titre`);

--

-- Index pour la table `spectacle`

--

ALTER TABLE `spectacle`

ADD PRIMARY KEY (`titre`);

--

-- Index pour la table `spectateurs`

--

ALTER TABLE `spectateurs`

ADD PRIMARY KEY (`id`),

ADD UNIQUE KEY `UC\_spec\_telFixe` (`telFixe`),

ADD UNIQUE KEY `UC\_spec\_telMob` (`telMobile`),

ADD UNIQUE KEY `UC\_spec\_mail` (`adresseMail`);

--

-- AUTO\_INCREMENT pour les tables déchargées

--

--

-- AUTO\_INCREMENT pour la table `chaise`

--

ALTER TABLE `chaise`

MODIFY `tri` int(11) NOT NULL AUTO\_INCREMENT, AUTO\_INCREMENT=801;

--

-- AUTO\_INCREMENT pour la table `personnel`

--

ALTER TABLE `personnel`

MODIFY `id` int(11) NOT NULL AUTO\_INCREMENT;

--

-- AUTO\_INCREMENT pour la table `reservations`

--

ALTER TABLE `reservations`

MODIFY `id` int(11) NOT NULL AUTO\_INCREMENT, AUTO\_INCREMENT=3;

--

-- AUTO\_INCREMENT pour la table `spectateurs`

--

ALTER TABLE `spectateurs`

MODIFY `id` int(11) NOT NULL AUTO\_INCREMENT, AUTO\_INCREMENT=8;

--

-- Contraintes pour les tables déchargées

--

--

-- Contraintes pour la table `adresse`

--

ALTER TABLE `adresse`

ADD CONSTRAINT `FKSPE\_adr\_FK` FOREIGN KEY (`id`) REFERENCES `spectateurs` (`id`);

--

-- Contraintes pour la table `chaise`

--

ALTER TABLE `chaise`

ADD CONSTRAINT `FKconcerner` FOREIGN KEY (`ResId`) REFERENCES `reservations` (`id`),

ADD CONSTRAINT `FKdisposer` FOREIGN KEY (`date`,`heure`) REFERENCES `representation` (`date`, `heure`);

--

-- Contraintes pour la table `organiser`

--

ALTER TABLE `organiser`

ADD CONSTRAINT `FKorg\_PER` FOREIGN KEY (`id`) REFERENCES `personnel` (`id`),

ADD CONSTRAINT `FKorg\_SPE` FOREIGN KEY (`titre`) REFERENCES `spectacle` (`titre`);

--

-- Contraintes pour la table `personnages`

--

ALTER TABLE `personnages`

ADD CONSTRAINT `FKSPE\_per` FOREIGN KEY (`titre`) REFERENCES `spectacle` (`titre`),

ADD CONSTRAINT `FKjouer\_par\_FK` FOREIGN KEY (`id`) REFERENCES `personnel` (`id`);

--

-- Contraintes pour la table `representation`

--

ALTER TABLE `representation`

ADD CONSTRAINT `FKpresenter` FOREIGN KEY (`titre`) REFERENCES `spectacle` (`titre`);

--

-- Contraintes pour la table `reservations`

--

ALTER TABLE `reservations`

ADD CONSTRAINT `FKporter\_sur` FOREIGN KEY (`titre`) REFERENCES `spectacle` (`titre`),

ADD CONSTRAINT `FKreserver` FOREIGN KEY (`SpecId`) REFERENCES `spectateurs` (`id`);

COMMIT;

/\*!40101 SET CHARACTER\_SET\_CLIENT=@OLD\_CHARACTER\_SET\_CLIENT \*/;

/\*!40101 SET CHARACTER\_SET\_RESULTS=@OLD\_CHARACTER\_SET\_RESULTS \*/;

/\*!40101 SET COLLATION\_CONNECTION=@OLD\_COLLATION\_CONNECTION \*/;

## Architecture

### Schéma d’architecture physique

**Outils nécessaires sur le client**

Un navigateur internet qui supporte le HTML5, le CSS3, ainsi que le JavaScript8

**Outils nécessaires sur le serveur**

Le serveur doit être un serveur PHP7 avec un moteur MariaDB 10.2.13

### Schéma d’architecture software

**Langages**

* PHP version 7 : interaction avec le serveur.
* HTML version 5 : structurer le site.
* CSS version 3 : design du site + responsive design.
* JavaScript version 8 : Interactivité avec l’utilisateur.
* SQL version 2011 : traitement des données du site.

**SGBD**

* MariaDB version 10 .2.13

**Librairies**

* Jquery version 3.1.1 : librairie JavaScript visant à simplifier son utilisation.

**Logiciels**

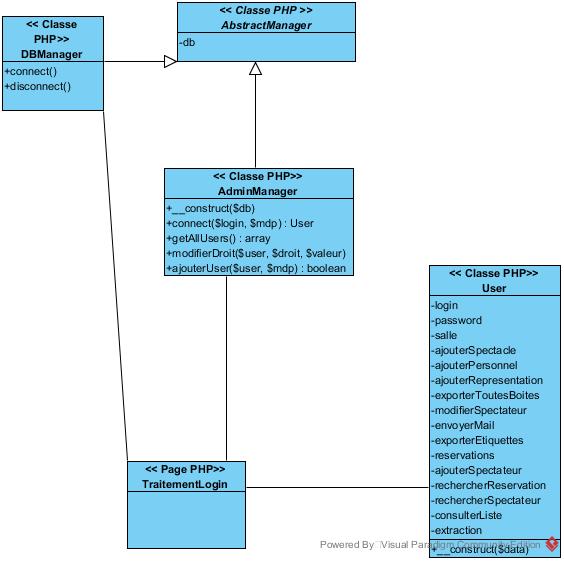
* JasperSoft Studio version 6.5.1 : génération de rapport en utilisant les données de la base de données.

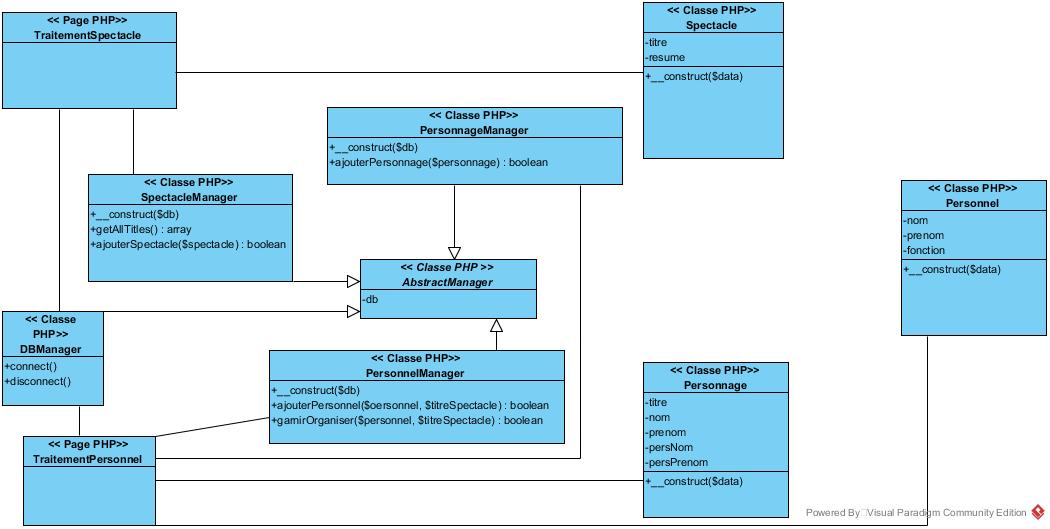
### 3.4.3 Organisation des répertoires du site (MVC)

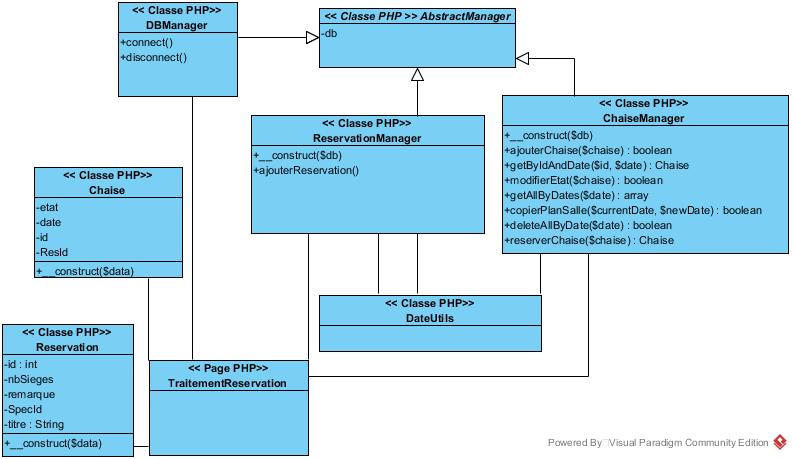
|  |  |  |  |
| --- | --- | --- | --- |
| Theatre | css |  | Les feuilles de styles |
| image |  | Image logo |
| js |  | Les librairies ou code personnel |
| manager | Couche persistance | Classes accédant à la base de données |
| model | Couche métier | Classes qui contiennent les données |
| traitements | Couche logique | Fichiers PHP qui effectuent différents traitements |
| utils | Couche logique | Fichiers contenant des fonctions utiles |
| index.php | Couche présentation | Page principale et présentation des données |

## Conception détaillée de chaque Use Case

### Diagramme de classes de conception

**S’identifier**

**Gérer spectacle**

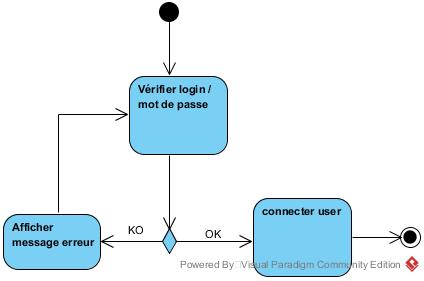
**Gérer réservation**

### Fonctions utilisées

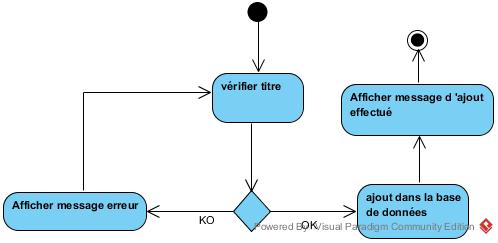
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Fonctions administration** | | | | | |
| **Nom** | **paramètres** | **type** | **utilité** | **Type de retour** | **Fichier source** |
| connect | $login  $mdp | String  String | Se connecter à l’application | User|String | AdminManager |
| getAllUsers | / | / | Avoir les utilisateurs de l’application | array | AdminManager |
| modifierDroit | $user  $droit  $valeur | String  String  String | Modifier les droits des utilisateurs | / | AdminManager |
| ajouterUser | $user  $mdp | String  String | Ajouter un utilisateur | boolean | AdminManager |
| **Fonctions spectacles** | | | | | |
| **Nom** | **paramètres** | **type** | **utilité** | **Type de retour** | **Fichier source** |
| getAllTitles | / | / | Avoir tous les titres des spectacles | array | SpectacleManager |
| ajouterSpectacle | $spectacle | Spectacle | Ajouter un spectacle | boolean | SpectacleManager |
| ajouterPersonnel | $personnel  $titreSpectacle | Personnel  String | Ajouter un personnel | boolean | PersonnelManager |
| garnirOrganiser | $personnel  $titreSpectacle | int  String | Garnir la table organiser | boolean | PersonnelManager |
| ajouterPersonnage | $personnage | Personnage | Ajouter un personnage | boolean | PersonnageManager |
| **Fonctions représentation** | | | | | |
| **Nom** | **paramètres** | **type** | **utilité** | **Type de retour** | **Fichier source** |
| ajouterRepresentation | $representation | Representation | Ajouter une représentation | boolean | RepresentationManager |
| getAllDates | / | / | Avoir toutes les dates des représentations | array | RepresentationManager |
| getAllDatesByTitle | $titre | String | Avoir toutes les dates de représentations en fonction du spectacle | array | RepresentationManager |
| getAllRepresentationsByTitle | $titre | String | Avoir toutes les représentations en fonction du spectacle | array | RepresentationManager |
| Fonctions spectateurs | | | | | |
| **Nom** | **paramètres** | **type** | **utilité** | **Type de retour** | **Fichier source** |
| ajouterSpectator | $spectator | Spectator | Ajouter un spectateur | boolean | SpectatorManager |
| ajouterAdresse | $spectator | Spectator | Ajouter l’adresse d’un spectateur | boolean | SpectatorManager |
| getAllSpectatorsMails | / | / | Avoir tous les spectateurs ayant une adresse mail | array | SpectatorManager |
| getAllSpectators | / | / | Avoir tous les spectateurs | array|null | SpectatorManager |
| getSpectatorByNomPrenom | $spectateur | Spectator | Avoir un spectateur par son nom et son prénom | null|array | SpectatorManager |
| modifierSpectatorById | $spectator | Spectator | Modifier un spectateur par son id | boolean | SpectatorManager |
| modifierAdresseById | $spectator | Spectator | Modifier l’adresse d’un spectateur par son id | boolean | SpectatorManager |
| autocompleteNom | / | / | Autocompletion du nom | null|array | SpectatorManager |
| autocompletePrenom | / | / | Autocompletion du prenom | null|array | SpectatorManager |
| Fonctions réservation | | | | | |
| **Nom** | **paramètres** | **type** | **utilité** | **Type de retour** | **Fichier source** |
| ajouterReservation | $reservation | *Reservation* | **Ajouter une réservation** | **boolean** | **ReservationManager** |
| ajouterChaise | $chaise | *Chaise* | **Ajouter une chaise** | **boolean** | **ChaiseManager** |
| getByIdAndDate | $id  $date  $heure | *String*  *date date* |  | *Chaise|null* | **ChaiseManager** |
| modifierEtat | $chaise | *Chaise* | **Modifier l’état d’une chaise** | *boolean* | **ChaiseManager** |
| getAllByDates | $date  $heure | *$date*  *date* |  | *array* | **ChaiseManager** |
| copierPlanSalle | $currentDate  $newDate | *String*  *String* | **Copier le plan d’une salle vers une autre date** | *boolean* | **ChaiseManager** |
| deleteAllByDate | $date | *String* |  | *boolean* | **ChaiseManager** |
| reserverChaise | $chaise | *Chaise* | **Reserver une chaise** | *boolean* | **ChaiseManager** |
| getReservationByNom | $spectateur  $date  $heure | *Spectator*  *Date*  *date* | **Avoir une reservation par son nom** | *boolean* | **ListeManager** |
| getReservationByNumReservation | $id | *int* | **Avoir une réservation par son numéro de réservation** | null|array | **ListeManager** |
| getAllReservationsByDate | $date  $heure | *String*  *String* | **Avoir toutes les réservations en fonction d’une date** | null|array | **ListeManager** |

### Diagrammes d’activité

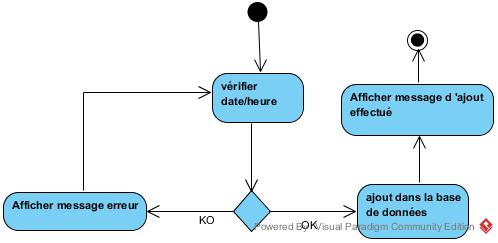
**S’identifier**

****

**Gérer spectacle**

****

**Gérer représentation**

****

### jeu d’essai niveau1

# Annexe

## 4.1 Interview

***Quelles sont les fonctionnalités que vous considérez comme importantes ou secondaires ?***

Les fonctionnalités les plus importantes pour nous sont les réservations, la disposition de la salle ainsi que la liste des réservations. Le reste des fonctionnalités sont plutôt secondaires (toutes boîtes, envoyer un mail, …).

***Désirez-vous reprendre les couleurs de votre blason pour l’application ?***

Oui, afin de rester dans le thème.

***Comment préférez-vous effectuer les réservations ?***

Nous préférons sélectionner les places directement sur le plan de la salle.

***Voulez-vous que l’application puisse permettre de gérer plusieurs salles ?***

Oui, nous avons comme projet de nous agrandir si nous en avons les moyens.

***Voulez-vous que l’application puisse permettre d’envisager plusieurs représentations à une même date ?***

Oui, toujours dans le cadre de notre projet d’agrandissement.

***Souhaitez-vous que l’application puisse permettre un regroupement des places dans le cas d’une seconde réservation d’un même spectateur ?***

Oui, s’il y a des places disponibles à proximité, nous essayerons de faire plaisir. Mais s’il n’y en a pas assez, nous ne pourrons pas satisfaire cette demande.

***Comment préférez-vous ajouter un personnel ?***

Le personnel peut être très différent d’un spectacle à un autre, donc il est préférable d’ajouter du personnel à un spectacle une fois ce dernier créé. Nous voulons aussi, en fonction de son rôle, spécifier des informations supplémentaires ou non.

***Êtes-vous intéressés par une page d’accueil expliquant brièvement les rubriques et permettant d’y accéder ?***

Oui, cela peut être intéressant.

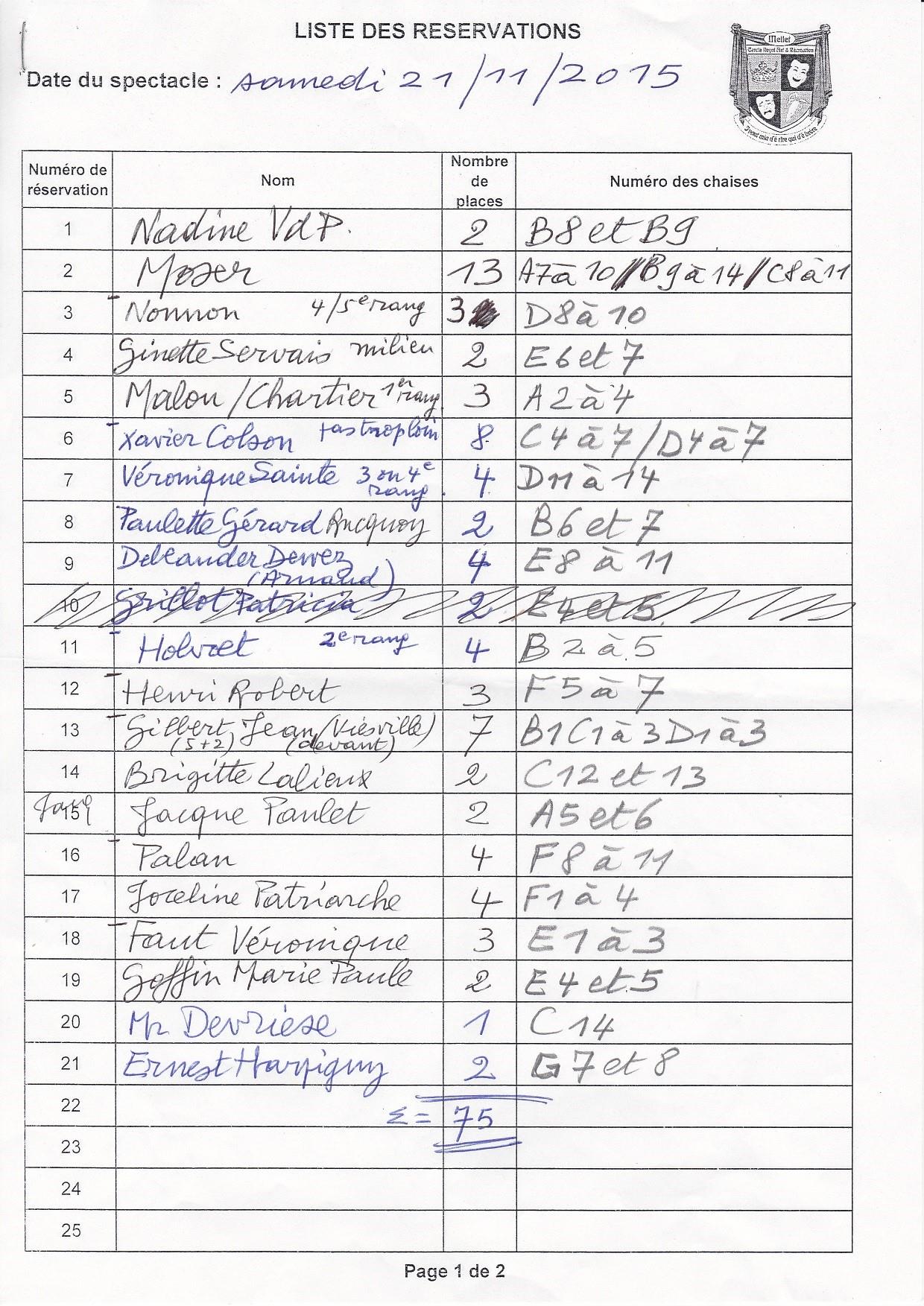
***Préférez-vous encoder l’adresse postale d’un spectateur sur un seul champ ou sur plusieurs champs (rue, numéro, …) ?***

Ça sera plus facile pour nous d’encoder l’adresse sur plusieurs champs distincts.

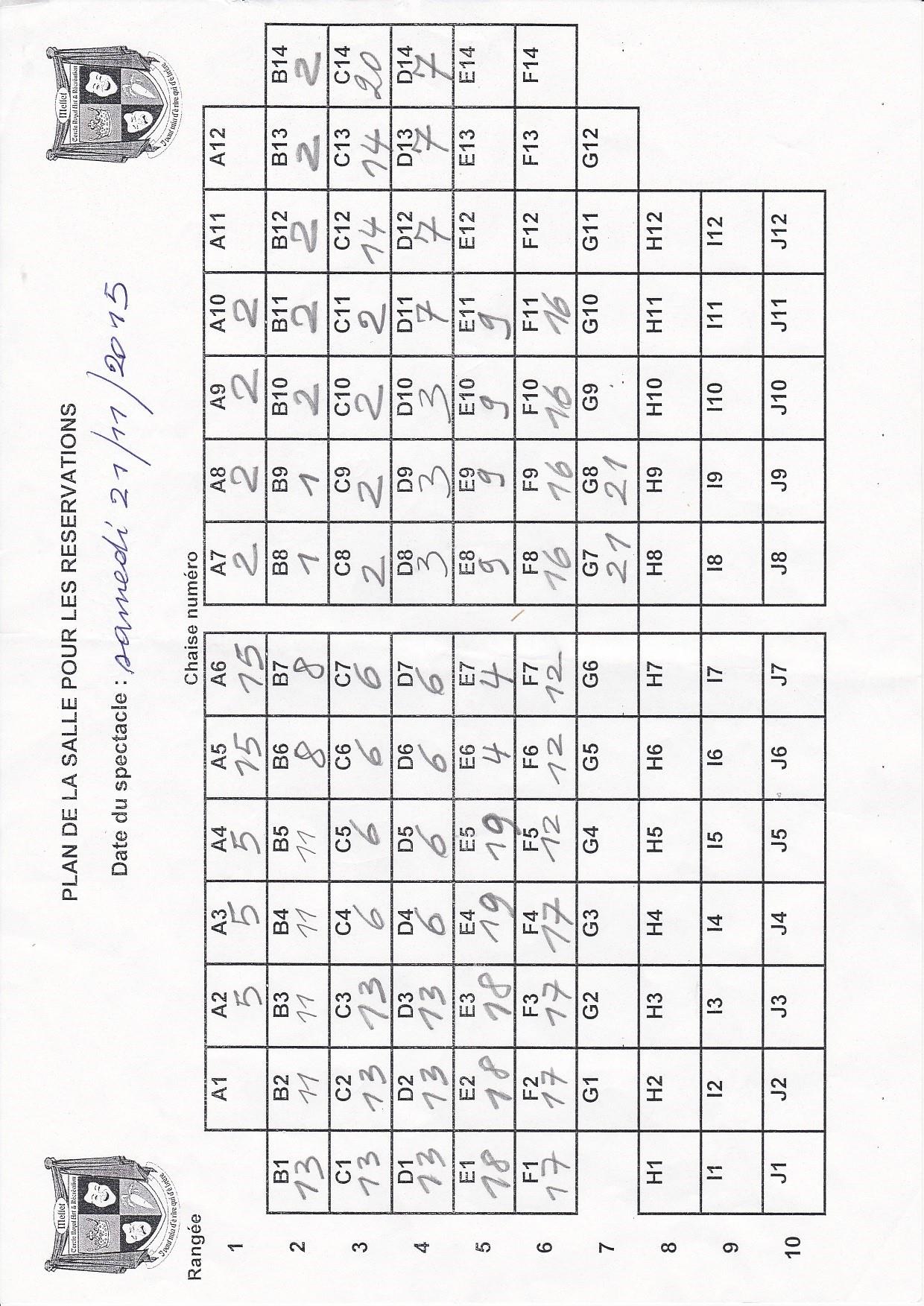
***Quel contenu désirez-vous dans le mail à envoyer à tous les spectateurs ?***

Nous désirons avoir la possibilité d’envoyer notre toutes boîtes ainsi que l’affiche reprenant les informations générales de notre spectacle.

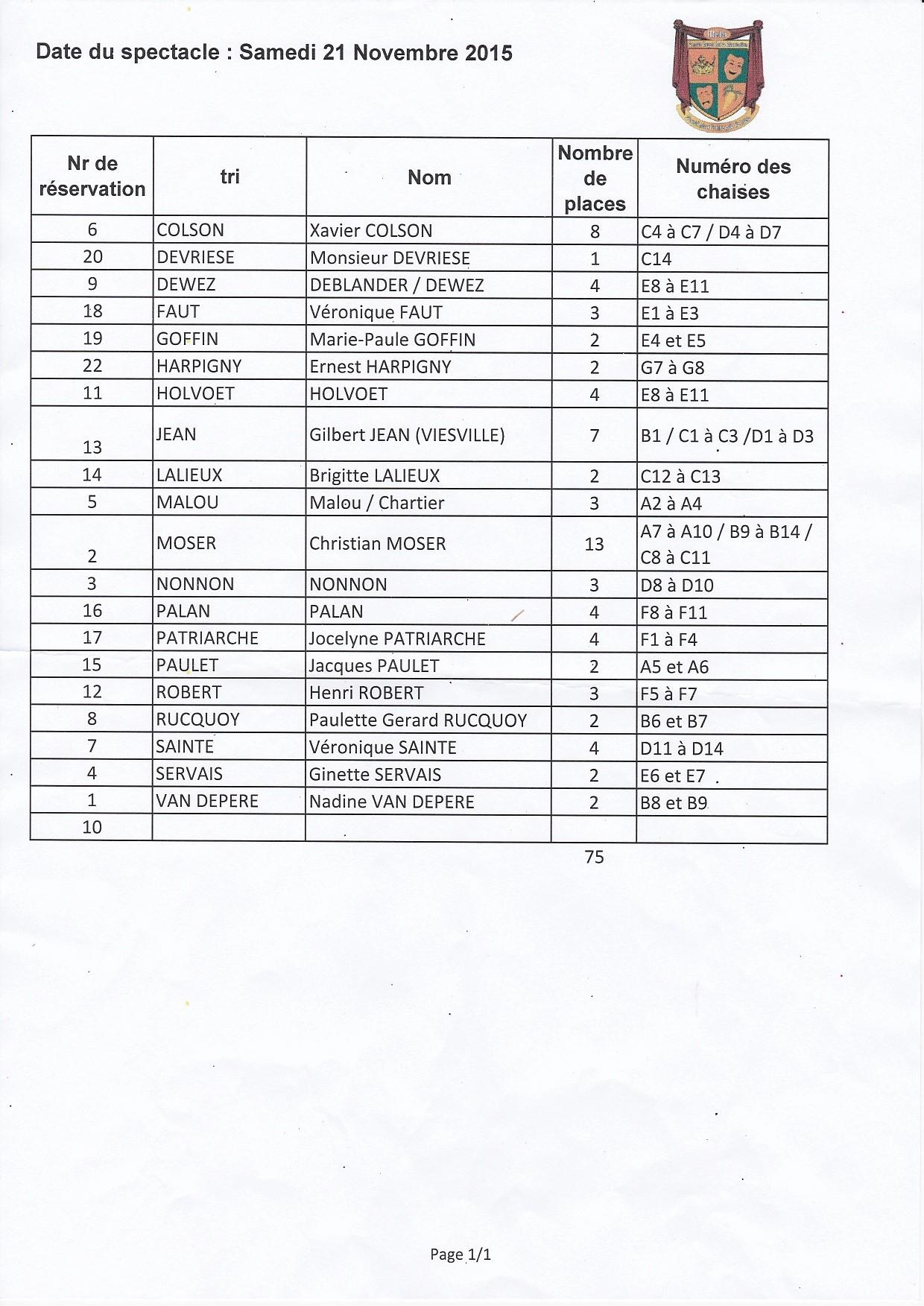
## 4.2 Annexe 2 : liste des réservations



## 4.3 Annexe 3 : plan de la salle



## 4.4 Annexe 4 : liste triée des réservations



## 4.5 Annexe 5 : compte-rendu de réunions

|  |  |
| --- | --- |
| **COMPTE-RENDU DE REUNION no 1** | |
| DATE : 09 février 2018 DUREE : 1 h | |
| Chef de projet : Verhaeghe Julien  PARTICIPANTS : Nardella Marine, Verhaeghe Julien | |
| PROBLEMATIQUE  Discussion :  Ébauche IHM  Modèle entités / associations | |
| A PREPARER  Ébauche IHM  Modèle entités / associations  Analyses préliminaires | RESPONSABLE  Nardella Marine  Verhaeghe Julien |
| Date prochaine réunion : 16/02/2018 | |

|  |  |
| --- | --- |
| **COMPTE-RENDU DE REUNION no 2** | |
| DATE : 16/02/2018 DUREE : 1h | |
| Chef de projet : Verhaeghe Julien  PARTICIPANTS : Nardella Marine, Verhaeghe Julien | |
| PROBLEMATIQUE  Discussion :  IHM  Modèle entité association  UC | |
| A PREPARER  Correction du modèle entité association  Correction de l’IHM  Faire le diagramme de UC | RESPONSABLE  Verhaeghe Julien  Nardella Marine |
| Date prochaine réunion : 23/02/2018 | |

|  |  |
| --- | --- |
| **COMPTE-RENDU DE REUNION no 3** | |
| DATE : 23/02/2018 DUREE : 1h | |
| Chef de projet : Verhaeghe Julien  PARTICIPANTS : Nardella Marine, Verhaeghe Julien | |
| PROBLEMATIQUE  Discussion :  UC  Diagramme de navigation  Modèle logique physique  IHM  Rapport | |
| A PREPARER  Correction diagramme de uc, navigation, modèle logique  Correction de l’IHM  Faire diagrammes de séquence, communication, classes participantes  Finir chapitre 1 et 2 | RESPONSABLE  Verhaeghe Julien  Nardella Marine |
| Date prochaine réunion : 02/03/2018 | |

|  |  |
| --- | --- |
| **COMPTE-RENDU DE REUNION no 4** | |
| DATE : 02/03/2018 DUREE : 1h | |
| Chef de projet : Verhaeghe Julien  PARTICIPANTS : Nardella Marine, Verhaeghe Julien | |
| PROBLEMATIQUE  Discussion :  Rapport | |
| A PREPARER  Présentation orale | RESPONSABLE  Verhaeghe Julien  Nardella Marine |
| Date prochaine réunion : 09/03/2018 | |

|  |  |
| --- | --- |
| **COMPTE-RENDU DE REUNION no 5** | |
| DATE : 10/03/2018 DUREE : 1h | |
| Chef de projet : Verhaeghe Julien  PARTICIPANTS : Nardella Marine, Verhaeghe Julien | |
| PROBLEMATIQUE  Discussion :  Maquette  Début de la programmation | |
| A PREPARER  Modifier rapport  Programmation | RESPONSABLE  Verhaeghe Julien  Nardella Marine |
| Date prochaine réunion : 16/03/2018 | |

|  |  |
| --- | --- |
| **COMPTE-RENDU DE REUNION no 6** | |
| DATE : 18/03/2018 DUREE : 1h | |
| Chef de projet : Verhaeghe Julien  PARTICIPANTS : Nardella Marine, Verhaeghe Julien | |
| PROBLEMATIQUE  Discussion :  Rapport  Programmation | |
| A PREPARER  Modifier rapport chapitre 1 & 2  Débuter rédaction rapport chapitre 3  Suite de la Programmation | RESPONSABLE  Verhaeghe Julien  Nardella Marine |
| Date prochaine réunion : 23/03/2018 | |

|  |  |
| --- | --- |
| **COMPTE-RENDU DE REUNION no 7** | |
| DATE : 24/03/2018 DUREE : 1h | |
| Chef de projet : Verhaeghe Julien  PARTICIPANTS : Nardella Marine, Verhaeghe Julien | |
| PROBLEMATIQUE  Discussion :  Rapport  Programmation | |
| A PREPARER  Continuer rédaction rapport chapitre 3  Suite de la Programmation | RESPONSABLE  Verhaeghe Julien  Nardella Marine |
| Date prochaine réunion : 31/03/2018 | |

|  |  |
| --- | --- |
| **COMPTE-RENDU DE REUNION no 8** | |
| DATE : 30/03/2018 DUREE : 1h | |
| Chef de projet : Verhaeghe Julien  PARTICIPANTS : Nardella Marine, Verhaeghe Julien | |
| PROBLEMATIQUE  Discussion :  Chapitre 3  Débugage | |
| A PREPARER  Chapitre 3  Correction de bugs  Suite de la programmation | RESPONSABLE  Verhaeghe Julien  Nardella Marine |
| Date prochaine réunion : 06/04/2018 | |

|  |  |
| --- | --- |
| **COMPTE-RENDU DE REUNION no 9** | |
| DATE : 09/04/2018 DUREE : 1h | |
| Chef de projet : Verhaeghe Julien  PARTICIPANTS : Nardella Marine, Verhaeghe Julien | |
| PROBLEMATIQUE  Discussion :  Débugage | |
| A PREPARER  Chapitre 3  Correction de bugs  Suite de la programmation | RESPONSABLE  Verhaeghe Julien  Nardella Marine |
| Date prochaine réunion : 14/04/2018 | |

|  |  |
| --- | --- |
| **COMPTE-RENDU DE REUNION no 10** | |
| DATE : 20/04/2018 DUREE : 1h | |
| Chef de projet : Verhaeghe Julien  PARTICIPANTS : Nardella Marine, Verhaeghe Julien | |
| PROBLEMATIQUE  Discussion :  Débugage  Rapports JasperSoft Studio | |
| A PREPARER  Correction de bugs  Suite de la programmation  Rapport | RESPONSABLE  Verhaeghe Julien  Nardella Marine |
| Date prochaine réunion : 23/04/2018 | |

|  |  |
| --- | --- |
| **COMPTE-RENDU DE REUNION no 11** | |
| DATE : 23/04/2018 DUREE : 30min | |
| Chef de projet : Verhaeghe Julien  PARTICIPANTS : Nardella Marine, Verhaeghe Julien | |
| PROBLEMATIQUE  Discussion :  Rapports JasperSoft Studio  Email | |
| A PREPARER  Tests unitaires  Suite de la programmation  Rapport | RESPONSABLE  Verhaeghe Julien  Nardella Marine |
| Date prochaine réunion : 28/04/2018 | |

|  |  |
| --- | --- |
| **COMPTE-RENDU DE REUNION no 12** | |
| DATE : 28/04/2018 DUREE : 1h | |
| Chef de projet : Verhaeghe Julien  PARTICIPANTS : Nardella Marine, Verhaeghe Julien | |
| PROBLEMATIQUE  Discussion :  Autoloader  Archivage | |
| A PREPARER  Rapport  Archivage  Autoloader  Correction de bugs | RESPONSABLE  Verhaeghe Julien  Nardella Marine |
| Date prochaine réunion : 04/05/2018 | |