

# Java distribué

Gestionnaire d'agendas

Pauline Réquéna

Guillaume Dubuisson Duplessis

16 mai 2009

# Table des matières

<b>1</b>	<b>Introduction</b>	<b>4</b>
<b>2</b>	<b>Analyse détaillée du sujet</b>	<b>5</b>
2.1	Description du sujet . . . . .	5
2.2	Structure du site . . . . .	6
2.3	Aspects techniques . . . . .	6
2.4	Diagramme de Gantt du mini-projet . . . . .	7
<b>3</b>	<b>Modélisation</b>	<b>8</b>
3.1	Identification des acteurs . . . . .	8
3.2	Diagramme de cas d'utilisation . . . . .	8
3.3	Les scénarios détaillés . . . . .	9
3.3.1	Scénario "s'authentifier" . . . . .	9
3.3.2	Scénario "Création d'un nouvel agenda" . . . . .	10
3.3.3	Scénario "Modifier les paramètres d'un agenda" . . . . .	11
3.3.4	Scénario "Supprimer un agenda" . . . . .	13
3.3.5	Scénario "Création d'un nouvel évènement" . . . . .	14
3.3.6	Scénario "Modifier les détails d'un évènement" . . . . .	16
3.3.7	Scénario "Supprimer un évènement" . . . . .	18
3.4	Diagramme d'activité . . . . .	18
3.4.1	Authentification . . . . .	19
3.4.2	Création d'agenda . . . . .	19
3.4.3	Création d'événement . . . . .	20
3.4.4	Modification d'agenda . . . . .	21
3.4.5	Modification d'événement . . . . .	22
3.4.6	Suppression d'agenda . . . . .	23
3.4.7	Suppression d'événement . . . . .	24
3.4.8	Diagramme d'activité global . . . . .	25
3.5	Diagramme de séquences . . . . .	26
3.5.1	Authentification . . . . .	26
3.5.2	Creer un agenda . . . . .	27
3.5.3	Modifier un agenda . . . . .	28
3.5.4	Supprimer un agenda . . . . .	29
3.5.5	Création d'un événement . . . . .	30
3.5.6	Modifier un événement . . . . .	31
3.5.7	Supprimer un événement . . . . .	32
3.6	Diagramme de classes . . . . .	33
3.6.1	Gestionnaire d'agendas . . . . .	33
3.6.2	Authentification . . . . .	34
3.6.3	Service DAO . . . . .	35
3.6.4	Les différents paquetages . . . . .	36
3.7	Modèle E/A et relationnel de la base de données . . . . .	37
3.7.1	Modèle entité/association . . . . .	37
3.7.2	Modèle relationnel . . . . .	37

---

<b>4 Exemple de fonctionnement</b>	<b>39</b>
4.1 Identification . . . . .	39
4.1.1 Authentification réussie . . . . .	39
4.1.2 Erreurs d'authentification . . . . .	39
4.2 Page d'accueil : structure et fonctionnement . . . . .	41
4.2.1 Le menu horizontal . . . . .	41
4.2.2 Le menu vertical de gauche . . . . .	41
4.2.3 Le calendrier d'affichage des agendas . . . . .	42
4.3 Créer un agenda . . . . .	44
4.3.1 Crédation réussie . . . . .	44
4.3.2 Erreurs de création . . . . .	44
4.4 Modifier les paramètres d'un agenda . . . . .	45
4.4.1 Suppression de l'agenda . . . . .	45
4.4.2 Modification des caractéristiques de l'agenda . . . . .	45
4.5 Créer un événement . . . . .	46
4.5.1 Crédation réussie . . . . .	46
4.5.2 Erreurs de création . . . . .	47
4.6 Modifier les paramètres d'un événement . . . . .	47
4.6.1 Suppresion de l'événement . . . . .	48
4.6.2 Modification des caractéristiques de l'événement . . . . .	48
<b>5 Pour aller plus loin : exemple d'utilisation du RMI</b>	<b>49</b>
5.1 Exemple envisagé . . . . .	49
5.2 Retranscription du fonctionnement . . . . .	49
5.3 Conclusion que nous pouvons tirer de cet exemple . . . . .	51
<b>6 Conclusion</b>	<b>52</b>

# 1 Introduction

A faire

## 2 Analyse détaillée du sujet

### 2.1 Description du sujet

Le sujet de notre mini-projet de Java Distribué consiste en la conception et l'implémentation d'un site web de gestion d'agendas. Ce site a pour fonction de gérer les différents types d'agendas d'un utilisateur. Par exemple, un utilisateur peut disposer d'un agenda « Travail » pour ses rendez-vous professionnels et d'un agenda « Perso » pour ses activités personnelles. Cette application permettra donc la gestion individuelle de ces 2 agendas.

Un utilisateur de ce gestionnaire devra donc pouvoir avoir accès aux fonctionnalités suivantes :

- **Consulter ses agendas sous une forme claire et fonctionnelle**

Nous avons opté pour l'affichage des agendas par semaine. Par ailleurs, afin de pouvoir bien distinguer les différents agendas lors de l'affichage des évènements, chaque agenda est caractérisé par une couleur.

- **Créer un nouvel agenda**

Un agenda est caractérisé par un nom, un lieu d'application, une description, ainsi qu'une couleur.

- **Modifier les paramètres d'un agenda préexistant**

Une fois qu'un agenda a été créé, l'utilisateur doit pouvoir s'il le souhaite en modifier les détails ou la couleur d'affichage.

- **Supprimer un agenda préexistant**

Lorsqu'un agenda n'est plus utilisé par l'internaute, celui-ci peut le supprimer, afin de ne pas encombrer son compte de données inutiles. Ce n'est pas une action anodine, car toute suppression est irréversible et entraîne la suppression de tous évènements de l'agenda.

- **Créer un nouvel évènement dans un agenda préexistant**

Un évènement appartient à un agenda. Il est caractérisé par un objet, une date, un lieu, des heures de début et de fin, ainsi qu'une description.

- **Modifier les caractéristiques d'un évènement préexistant**

Une fois qu'un évènement a été créé dans un agenda, l'utilisateur doit pouvoir s'il le souhaite en modifier les détails.

- **Supprimer un évènement préexistant**

## 2.2 Structure du site

L'accès à ce site se fera par le biais d'une page d'authentification. En effet, les comptes de chaque utilisateur sont confidentiels. Pour se connecter au gestionnaire d'agendas, l'internaute devra donc en premier lieu saisir son login et son mot de passe.

Une fois l'authentification accomplie, l'internaute sera dirigé vers la page d'accueil, composée de plusieurs parties :

- **L'en-tête**

Elle est constituée du logo de l'application, d'un message d'accueil et du bouton de déconnexion.

- **Le menu de gauche**

Ce menu est composé tout d'abord de la liste des agendas de l'utilisateur connecté. Ce dernier peut sélectionner dans cette liste les agendas qu'il désire ou non afficher.

A travers ce menu, l'utilisateur peut également accéder aux différentes fonctionnalités de gestion des agendas, à savoir la création d'un nouvel agenda, la modification ou la suppression d'un agenda, ainsi que la création d'un nouvel évènement. La modification ou la suppression d'un évènement se fera par le biais du calendrier d'affichage.

- **Le corps de la page**

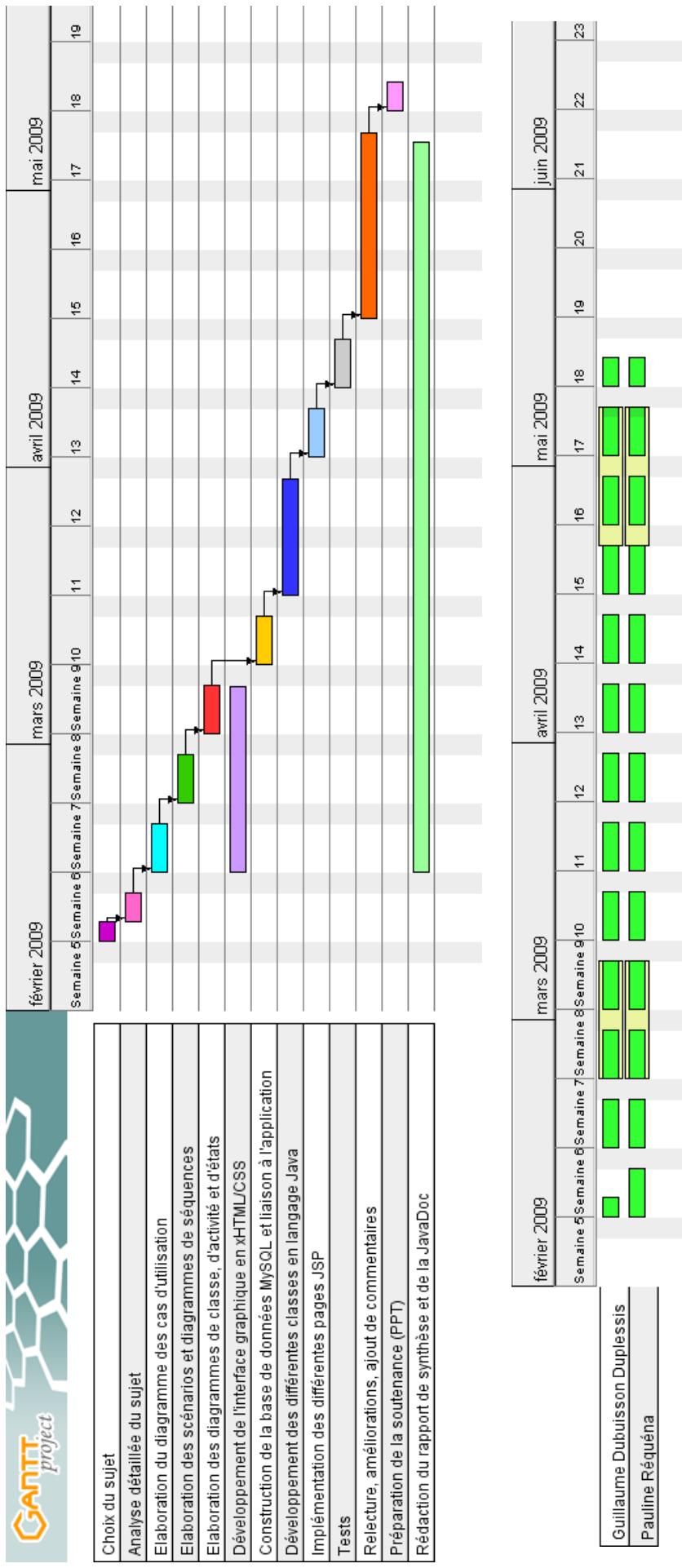
Il est composé du tableau d'affichage des agendas. En cliquant sur un des évènements affiché sur ce calendrier, l'utilisateur pourra en modifier les détails, ou le supprimer.

L'affichage des agendas étant hebdomadaire, l'utilisateur pourra passer d'une semaine à l'autre à l'aide de flèches directionnelles.

## 2.3 Aspects techniques

- Le planning de ce projet sera réalisé sur le logiciel GanttProject.
- La modélisation et le développement de ce projet seront effectués sur l'IDE Netbeans 6.5.
- Les interfaces des différentes pages web seront réalisées en xHTML/CSS.
- Des JSP permettront d'insérer des données dynamiques à ce contenu statique.
- Les informations enregistrées dans l'application seront stockées dans une base de données MySQL et la liaison de l'application à la base se fera via une connexion JDBC.

## 2.4 Diagramme de Gantt du mini-projet

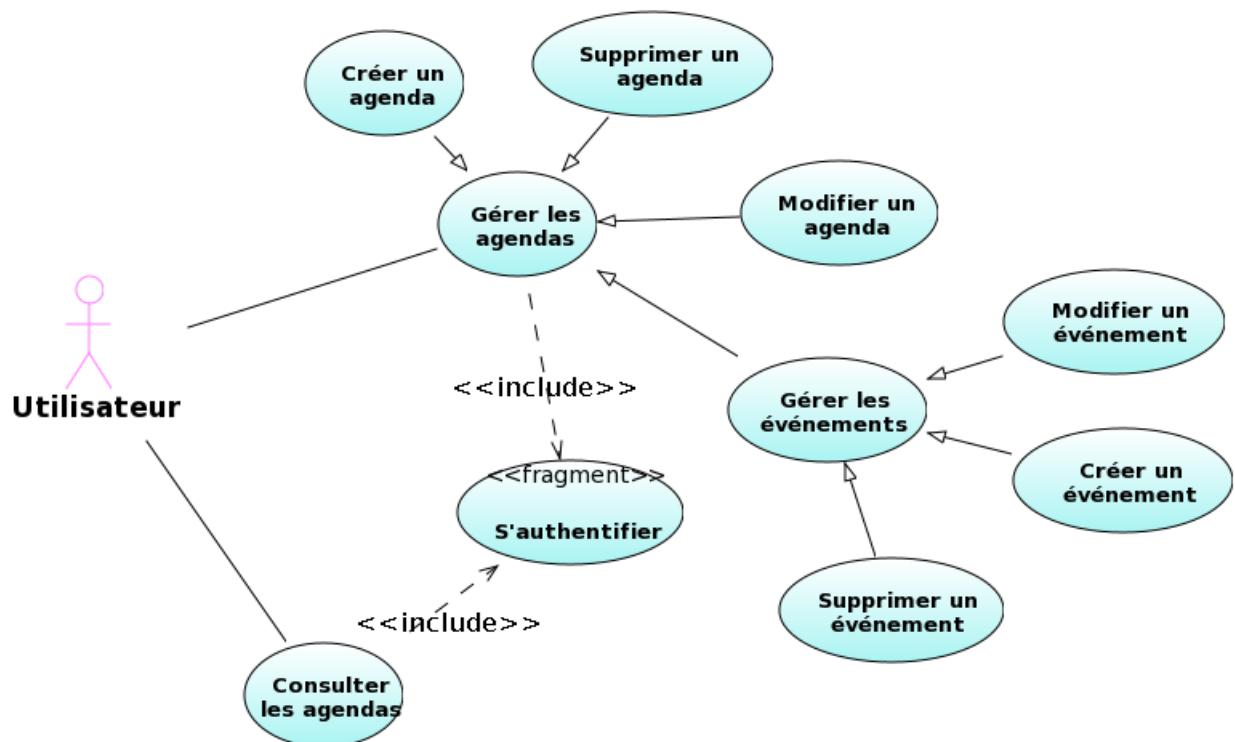


## 3 Modélisation

### 3.1 Identification des acteurs

Dans le cadre de ce gestionnaire d'agendas, nous pouvons déterminer deux principaux acteurs : l'utilisateur "lambda" qui gère ses agendas et l'administrateur du site. Dans un soucis de simplicité, nous ne développerons pas l'administration du site. Nous ne considérerons donc pas l'administrateur.

### 3.2 Diagramme de cas d'utilisation



Nous pouvons remarquer qu'il n'apparaît pas de scénario d'inscription au site internet. En effet, nous préférons nous concentrer sur la gestion des agendas en elle-même que sur la gestion des membres.

### 3.3 Les scénarios détaillés

#### 3.3.1 Scénario "s'authentifier"

**Titre :** s'authentifier

**Résumé :** ce cas d'utilisation permet à l'utilisateur de s'authentifier

**Acteur :** utilisateur

**Préconditions :**

- L'utilisateur n'est pas déjà authentifié
- La base de données stockant la liste utilisateur/mot de passe est accessible

**Scénario nominal :**

1. Le système demande à l'utilisateur un nom d'utilisateur et un mot de passe
2. L'utilisateur indique son nom d'utilisateur et son mot de passe
3. Le système vérifie qu'un utilisateur correspond au nom d'utilisateur
4. Le système vérifie que le mot de passe indiqué correspond au mot de passe associé au nom de l'utilisateur
5. Le système crée une session pour l'utilisateur

**Enchaînements alternatifs :**

*A1 : nom d'utilisateur provisoirement erroné*

L'enchaînement A1 démarre au point 3 du scénario nominal.

4. Le système indique à l'utilisateur que le nom d'utilisateur est erroné pour la première ou la deuxième fois
5. Le système incrémente le compteur des tentatives d'authentification

Le scénario nominal reprend au point 1.

*A2 : mot de passe provisoirement erroné*

L'enchaînement A2 démarre au point 4 du scénario nominal.

5. Le système indique à l'utilisateur que le mot de passe est erroné pour la première ou la deuxième fois
6. Le système incrémente le compteur des tentatives d'authentification

Le scénario nominal reprend au point 1.

**Enchaînements d'erreur :**

*E1 : nom d'utilisateur définitivement erroné*

L'enchaînement E1 démarre au point 3 du scénario nominal.

4. Le système indique à l'utilisateur que le nom d'utilisateur est erroné pour la 3ème fois
5. Le système indique que toute tentative d'authentification sera refusée
6. Le système enregistre la machine comme étant bannie pour une durée déterminée (par exemple 1h)

*E2 : mot de passe définitivement erroné*

L'enchaînement E2 démarre au point 4 du scénario nominal.

5. Le système indique à l'utilisateur que le mot de passe est erroné pour la 3ème fois
6. Le système indique que toute tentative d'authentification sera refusée
7. Le système enregistre la machine comme étant bannie pour une durée déterminée (par exemple 1h)

### 3.3.2 Scénario "Création d'un nouvel agenda"

**Titre :** Crédit d'un nouvel agenda

**Résumé :** Ce scénario permet à l'utilisateur de créer un nouvel agenda dans son calendrier.

**Acteur :** utilisateur

**Préconditions :**

- L'utilisateur est authentifié
- La base de données stockant la table des agendas est accessible

**Scénario nominal :**

1. L'utilisateur indique qu'il souhaite créer un nouvel agenda
2. Le système affiche le formulaire de création d'agendas.
3. L'utilisateur renseigne le formulaire avec le nom de l'agenda, le lieu, une description, et une couleur représentative
4. Le système vérifie que le champ "Nom" est bien renseigné
5. Le système vérifie qu'aucun agenda du même nom n'existe déjà dans la base
6. Le système crée un nouvel agenda dans la base de données.

**Enchaînements alternatifs :**

*A1 : le champ nom est vide*

L'enchaînement A1 démarre au point 4 du scénario nominal.

5. Le système indique à l'utilisateur que le champ « Nom » n'est pas rempli et que la création est donc impossible.

Le scénario nominal reprend au point 2.

*A2 : il y a déjà un agenda du même nom dans la base*

L'enchaînement A2 démarre au point 5 du scénario nominal.

6. Le système indique à l'utilisateur que ce nom est déjà utilisé et qu'il doit en choisir un nouveau.

Le scénario nominal reprend au point 2.

### 3.3.3 Scénario "Modifier les paramètres d'un agenda"

**Titre :** Modification d'un agenda

**Résumé :** Ce scénario permet à l'utilisateur de modifier les caractéristiques d'un agenda existant, s'il veut par exemple changer sa couleur ou son nom.

**Acteur :** utilisateur

**Préconditions :**

- L'utilisateur est authentifié
- L'agenda doit exister
- La base de données stockant la table des agendas est accessible

**Scénario nominal :**

1. L'utilisateur indique qu'il souhaite modifier un de ses agendas
2. Le système affiche le formulaire de modification d'agendas
3. L'utilisateur sélectionne un agenda à modifier
4. Le système vérifie que l'agenda existe dans la base de données
5. Le système vérifie que l'agenda appartient bien à l'utilisateur
6. Le système va chercher dans la base les caractéristiques de l'agenda et les affiche dans le formulaire.
7. L'utilisateur renseigne les différents champs, suivant les modifications qu'il désire effectuer et clique sur « Modifier ».
8. Le système vérifie que le champ « Nom » est bien renseigné.
9. Le système vérifie qu'aucun agenda du même nom n'existe pas déjà dans la base.
10. Le système modifie les informations de l'agenda dans la base de données

**Enchaînements alternatifs :**

*A1 : l'agenda n'existe pas*

L'enchaînement A1 démarre au point 4 du scénario nominal.

5. Le système indique à l'utilisateur que l'agenda n'existe pas donc il n'y a pas de modification possible

Le scénario nominal reprend au point 2.

*A2 : l'agenda n'appartient pas à l'utilisateur*

L'enchaînement A2 démarre au point 5 du scénario nominal.

6. Le système indique à l'utilisateur qu'il n'a pas l'autorisation de modifier cet agenda

Le scénario nominal reprend au point 2.

*A3 : le champ nom est vide*

L'enchaînement A3 démarre au point 8 du scénario nominal.

9. Le système indique à l'utilisateur que le champ « Nom » n'est pas rempli et que la modification est donc impossible

Le scénario nominal reprend au point 6.

*A4 : il y a déjà un agenda du même nom dans la base*

L'enchaînement A4 démarre au point 9 du scénario nominal.

10. Le système indique à l'utilisateur que ce nom est déjà utilisé et qu'il doit en choisir un nouveau

Le scénario nominal reprend au point 6.

### 3.3.4 Scénario "Supprimer un agenda"

**Titre :** Suppression d'un agenda

**Résumé :** Ce scénario permet à l'utilisateur de supprimer un agenda existant, s'il n'en a plus l'utilité. Cette action a pour conséquence de supprimer tous les évènements relatifs à cet agenda.

**Acteur :** utilisateur

**Préconditions :**

- L'utilisateur est authentifié
- L'agenda doit exister
- La base de données stockant la table des agendas et des événements est accessible

**Scénario nominal :**

1. L'utilisateur indique qu'il souhaite supprimer un agenda
2. Le système affiche le formulaire de modification d'agendas
3. L'utilisateur sélectionne un agenda à supprimer
4. Le système vérifie que l'agenda existe dans la base de données.
5. Le système vérifie que l'agenda appartient bien à l'utilisateur
6. Le système va chercher dans la base les caractéristiques de l'agenda et les affiche dans le formulaire.
7. L'utilisateur confirme la suppression
8. Le système supprime l'agenda dans la base de données, ainsi que tous les évènements liés à cet agenda

**Enchaînements alternatifs :**

*A1 : l'agenda n'existe pas*

L'enchaînement A1 démarre au point 4 du scénario nominal.

5. Le système indique à l'utilisateur que l'agenda n'existe pas donc il n'y a pas de modification possible

Le scénario nominal reprend au point 2.

*A2 : l'agenda n'appartient pas à l'utilisateur*

L'enchaînement A2 démarre au point 5 du scénario nominal.

6. Le système indique à l'utilisateur qu'il n'a pas l'autorisation de modifier cet agenda

Le scénario nominal reprend au point 2.

### 3.3.5 Scénario "Création d'un nouvel évènement"

**Titre :** Crédation d'un nouvel évènement

**Résumé :** Ce scénario permet à l'utilisateur de créer un nouvel évènement dans un agenda de son calendrier

**Acteur :** utilisateur

**Préconditions :**

- L'utilisateur est authentifié
- L'agenda doit exister
- La base de données stockant la table des agendas et des événements est accessible

**Scénario nominal :**

1. L'utilisateur indique qu'il souhaite créer un nouvel événement
2. Le système affiche le formulaire de création d'évènement.
3. L'utilisateur renseigne le formulaire avec l'objet de l'évènement, la date, le lieu, les heures de début et de fin, l'agenda auquel il appartient, et une description
4. Le système vérifie que l'agenda sélectionné existe bien dans la base.
5. Le système vérifie que l'agenda sélectionné appartient bien à l'utilisateur.
6. Le système vérifie que les champs « Objet », « Date », « Heure de début » et « Heure de fin » sont bien renseignés.
7. Le système vérifie qu'aucun évènement du même agenda n'existe déjà dans la base sur une plage horaire commune.
8. Le système crée un nouvel évènement dans la base de données.

**Enchaînements alternatifs :**

*A1 : l'agenda n'existe pas*

L'enchaînement A1 démarre au point 4 du scénario nominal.

5. Le système indique à l'utilisateur que l'agenda n'existe pas donc il n'y a pas de modification possible

Le scénario nominal reprend au point 2.

*A2 : l'agenda n'appartient pas à l'utilisateur*

L'enchaînement A2 démarre au point 5 du scénario nominal.

6. Le système indique à l'utilisateur qu'il n'a pas l'autorisation de modifier cet agenda

Le scénario nominal reprend au point 2.

*A3 : un champ obligatoire n'est pas renseigné*

L'enchaînement A3 démarre au point 6 du scénario nominal.

7. Le système indique à l'utilisateur que certains des champs « Objet », « Date », « Heure de début » ou « Heure de fin » ne sont pas renseignés.

Le scénario nominal reprend au point 2.

*A4 : un évènement simultané existe déjà*

L'enchaînement A4 démarre au point 7 du scénario nominal.

8. Le système indique à l'utilisateur qu'il doit changer les horaires de son évènement car cette plage horaire est déjà occupée.

Le scénario nominal reprend au point 2.

### 3.3.6 Scénario "Modifier les détails d'un évènement"

**Titre :** Modification d'un évènement

**Résumé :** Ce scénario permet à l'utilisateur de modifier les détails d'un évènement existant dans un agenda de son calendrier. Il peut par exemple vouloir changer la plage horaire d'un rendez-vous, ou le lieu d'une soirée etc.

**Acteur :** utilisateur

**Préconditions :**

- L'utilisateur est authentifié
- L'agenda doit exister
- La base de données stockant la table des agendas et des événements est accessible

**Scénario nominal :**

1. L'utilisateur sélectionne l'évènement qu'il veut modifier.
2. Le système vérifie que l'évènement sélectionné existe bien dans la base.
3. Le système vérifie que l'évènement sélectionné appartient bien à l'utilisateur.
4. Le système va chercher dans la base les caractéristiques de l'évènement et affiche le formulaire de modification d'évènement.
5. L'utilisateur renseigne le formulaire en modifiant les champs voulus et clique sur le bouton « Modifier ».
6. Le système vérifie que les champs « Objet », « Date », « Heure de début » et « Heure de fin » sont bien renseignés.
7. Le système vérifie qu'aucun évènement du même agenda n'existe déjà dans la base sur une plage horaire commune.
8. Le système modifie les caractéristiques de l'évènement dans la base de données.

**Enchaînements alternatifs :**

*A1 : l'événement n'existe pas*

L'enchaînement A1 démarre au point 4 du scénario nominal.

5. Le système indique à l'utilisateur que l'évènement n'existe pas donc il n'y a pas de modification possible

Le scénario nominal reprend au point 1.

*A2 : l'événement n'appartient pas à l'utilisateur*

L'enchaînement A2 démarre au point 3 du scénario nominal.

6. Le système indique à l'utilisateur qu'il n'a pas l'autorisation de modifier cet événement

Le scénario nominal reprend au point 1.

*A3 : un champ obligatoire n'est pas renseigné*

L'enchaînement A3 démarre au point 6 du scénario nominal.

7. Le système indique à l'utilisateur que certains des champs « Objet », « Date », « Heure de début » ou « Heure de fin » ne sont pas renseignés.

Le scénario nominal reprend au point 5.

*A4 : un évènement simultané existe déjà*

L'enchaînement A4 démarre au point 7 du scénario nominal.

8. Le système indique à l'utilisateur qu'il doit changer les horaires de son évènement car cette plage horaire est déjà occupée.

Le scénario nominal revient au point 5.

### 3.3.7 Scénario "Supprimer un évènement"

**Titre :** Suppression d'un évènement

**Résumé :** Ce scénario permet à l'utilisateur de supprimer un évènement existant

**Acteur :** utilisateur

**Préconditions :**

- L'utilisateur est authentifié
- L'agenda doit exister
- La base de données stockant la table des agendas et des événements est accessible

**Scénario nominal :**

1. L'utilisateur sélectionne l'évènement qu'il veut modifier
2. Le système vérifie que l'évènement existe dans la base de données
3. Le système vérifie que l'évènement appartient bien à l'utilisateur
4. Le système va chercher dans la base les caractéristiques de l'évènement sélectionné et les affiche dans le formulaire
5. L'utilisateur clique sur « Supprimer »
6. Le système supprime l'évènement de la base de données

**Enchaînements alternatifs :**

*A1 : l'événement n'existe pas*

L'enchaînement A1 démarre au point 2 du scénario nominal.

5. Le système indique à l'utilisateur que l'évènement n'existe pas donc il n'y a pas de modification possible

Le scénario nominal reprend au point 1.

*A2 : l'événement n'appartient pas à l'utilisateur*

L'enchaînement A2 démarre au point 3 du scénario nominal.

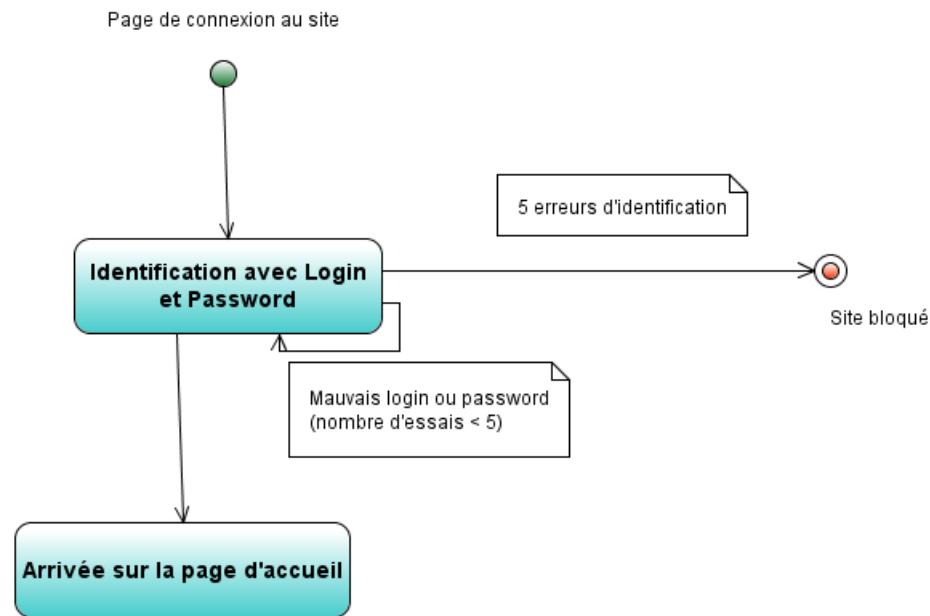
6. Le système indique à l'utilisateur qu'il n'a pas l'autorisation de modifier cet événement

Le scénario nominal reprend au point 1.

## 3.4 Diagramme d'activité

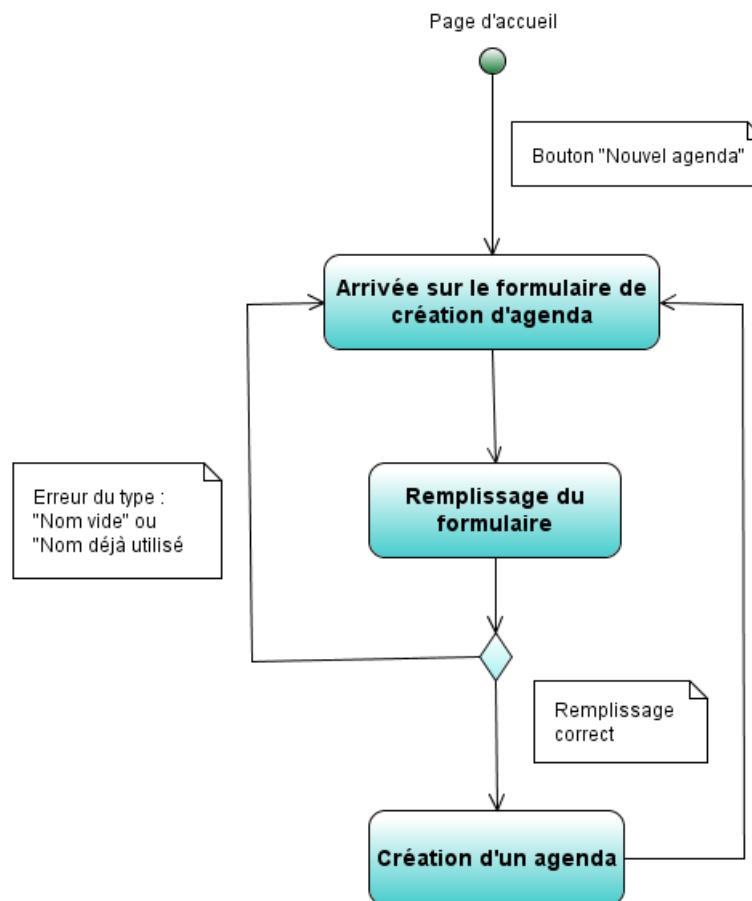
Nous avons vu les différents scénarios du projet, nous allons maintenant voir les différents diagrammes d'activité correspondant.

### 3.4.1 Authentification



Comme nous pouvons le voir sur ce diagramme, l'utilisateur indique son login et son mot de passe. Il a 5 tentatives pour arriver à se connecter, après quoi il n'aura plus accès à la connexion.

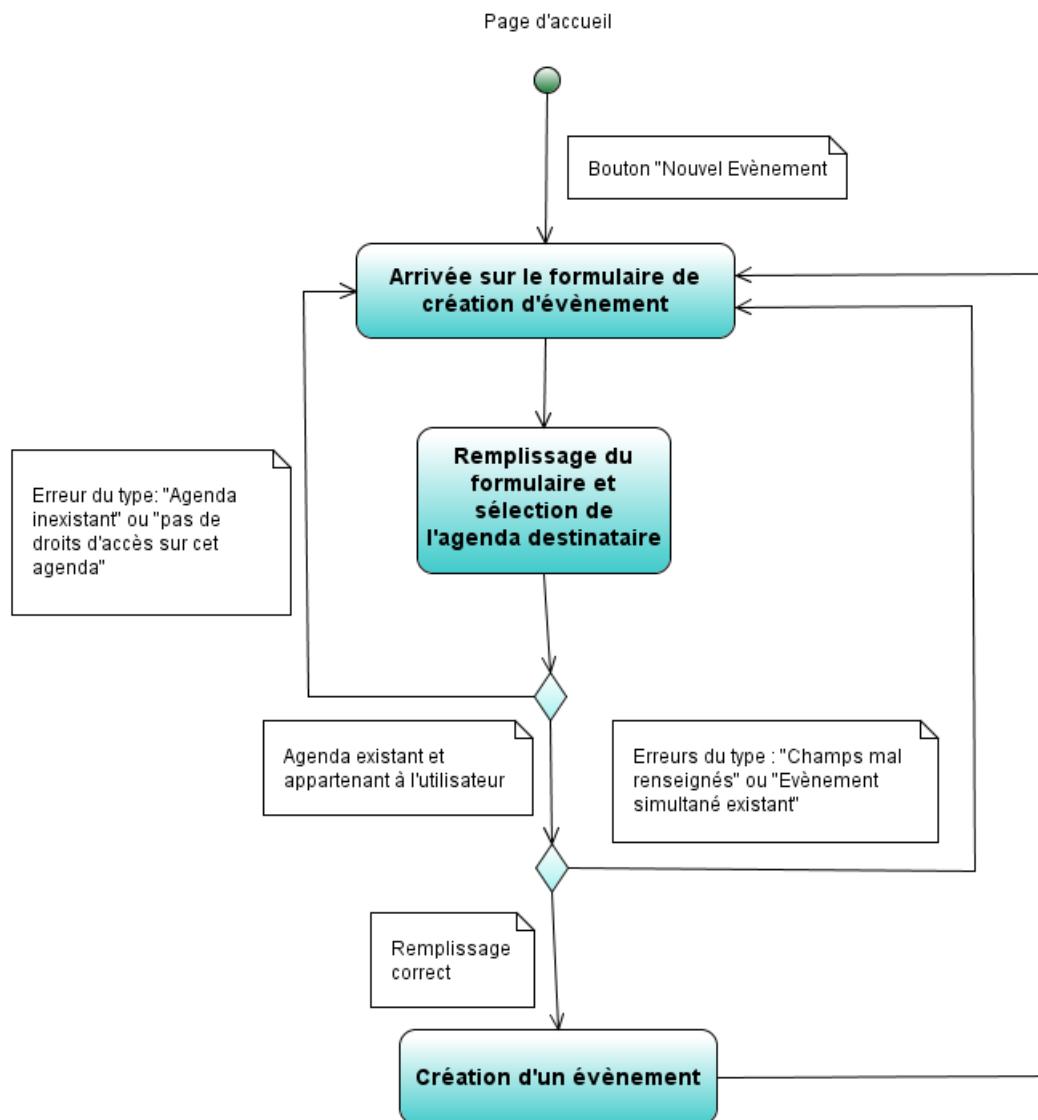
### 3.4.2 Crédation d'agenda



Comme nous pouvons le voir sur ce diagramme, la création d'un agenda consiste en le remplissage d'un formulaire. Les données communiquées par le formulaire sont vérifiés. Dans le

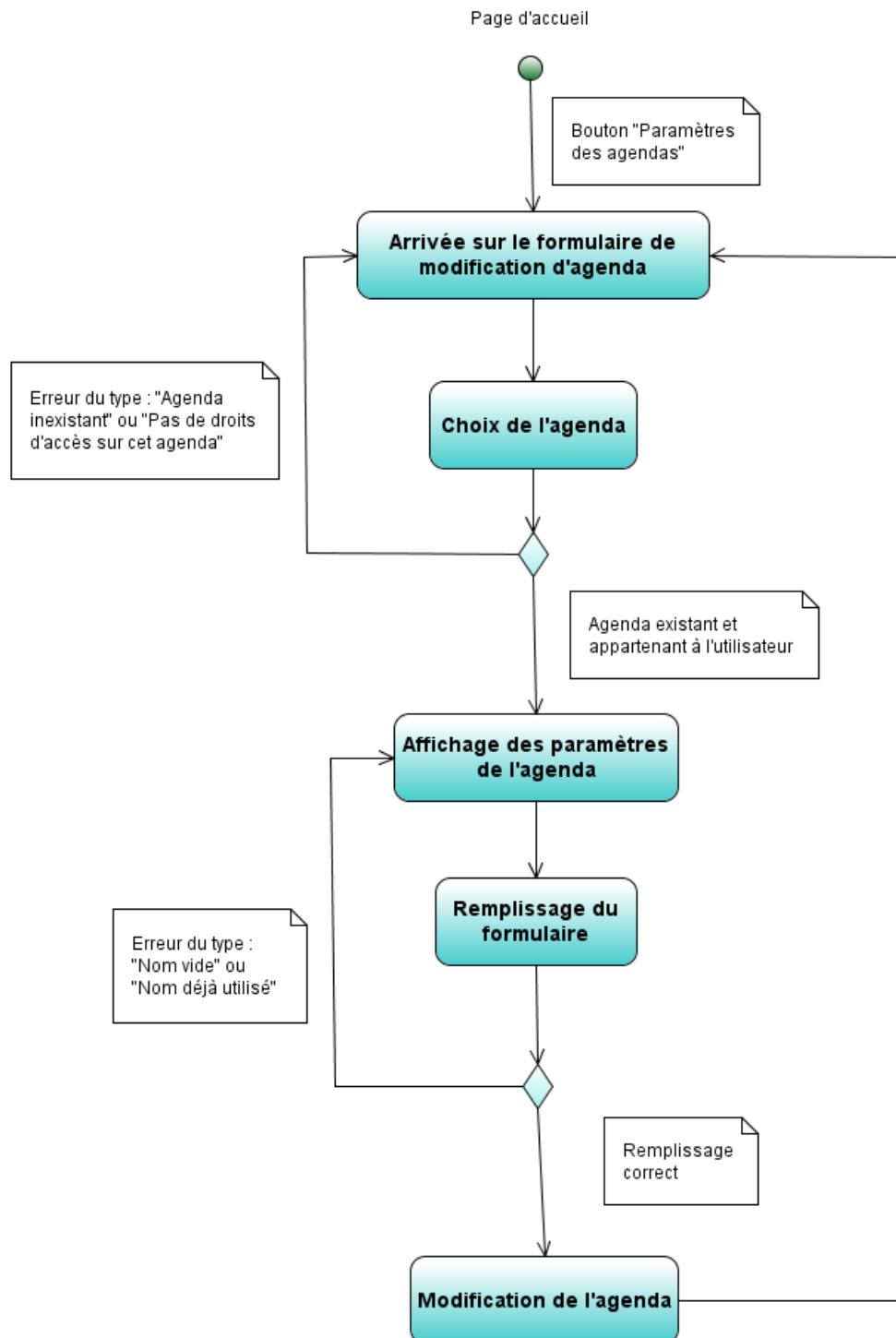
cas où ces données seraient erronées (nom vide ou déjà utilisé), l'utilisateur est invité à remplir de nouveau le formulaire. Enfin si le formulaire est correctement rempli, l'agenda est créé.

### 3.4.3 Crédit d'événement

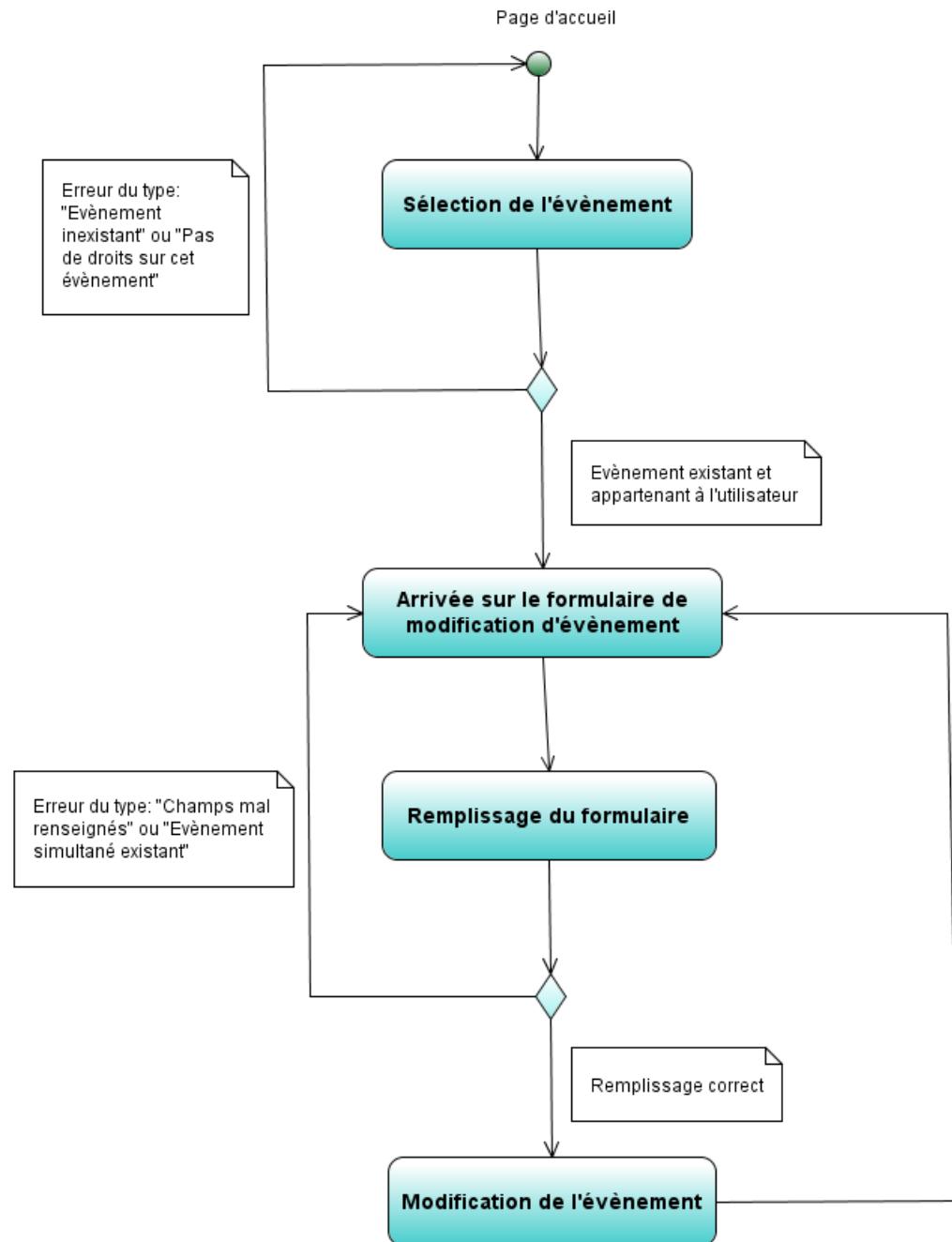


De la même manière que la création d'agenda, l'utilisateur est invité à remplir un formulaire pour créer des événements. L'événement est créé quand toutes les conditions sont remplies : champs bien renseignés, aucun événement simultané n'existe, l'agenda dans lequel l'utilisateur souhaite ajouter un événement lui appartient.

### 3.4.4 Modification d'agenda

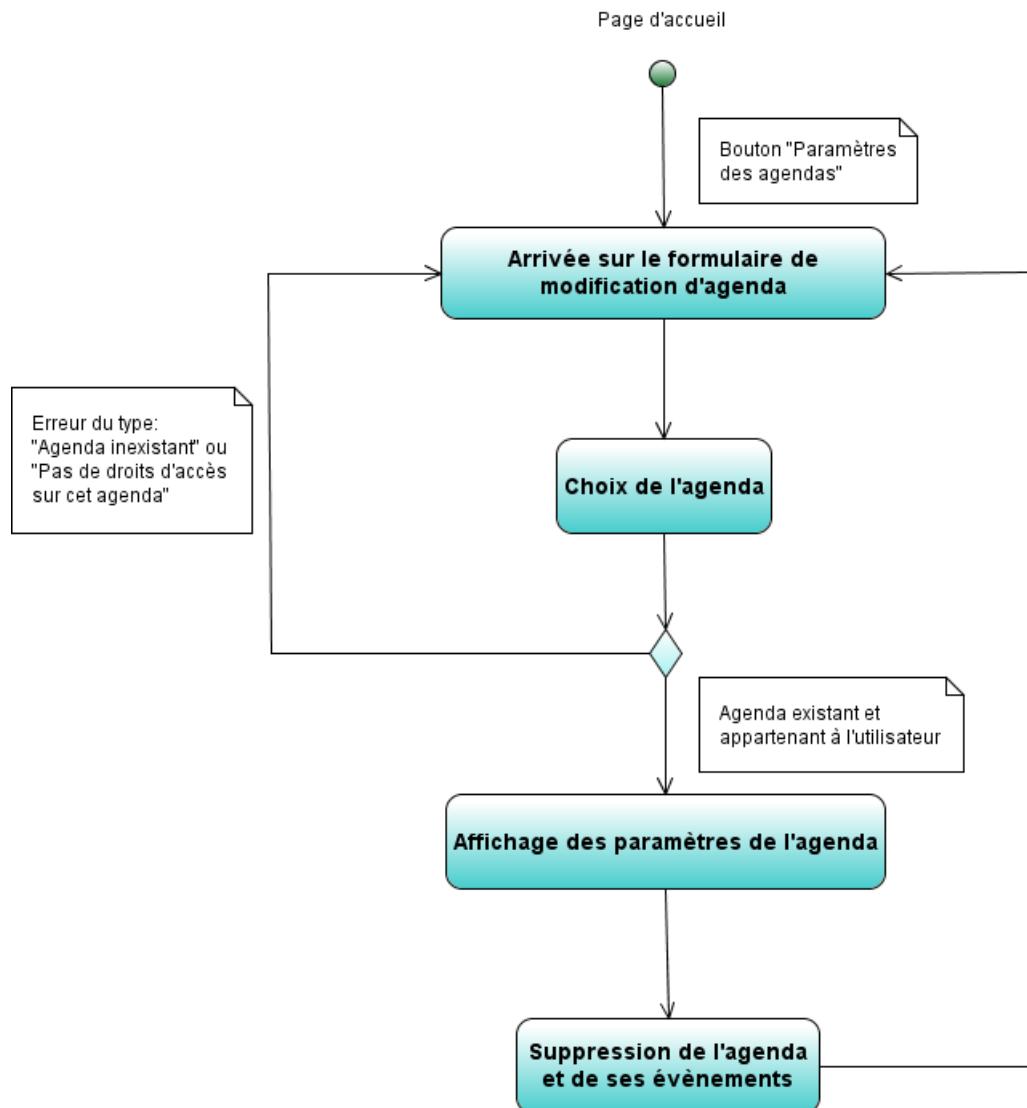


### 3.4.5 Modification d'événement



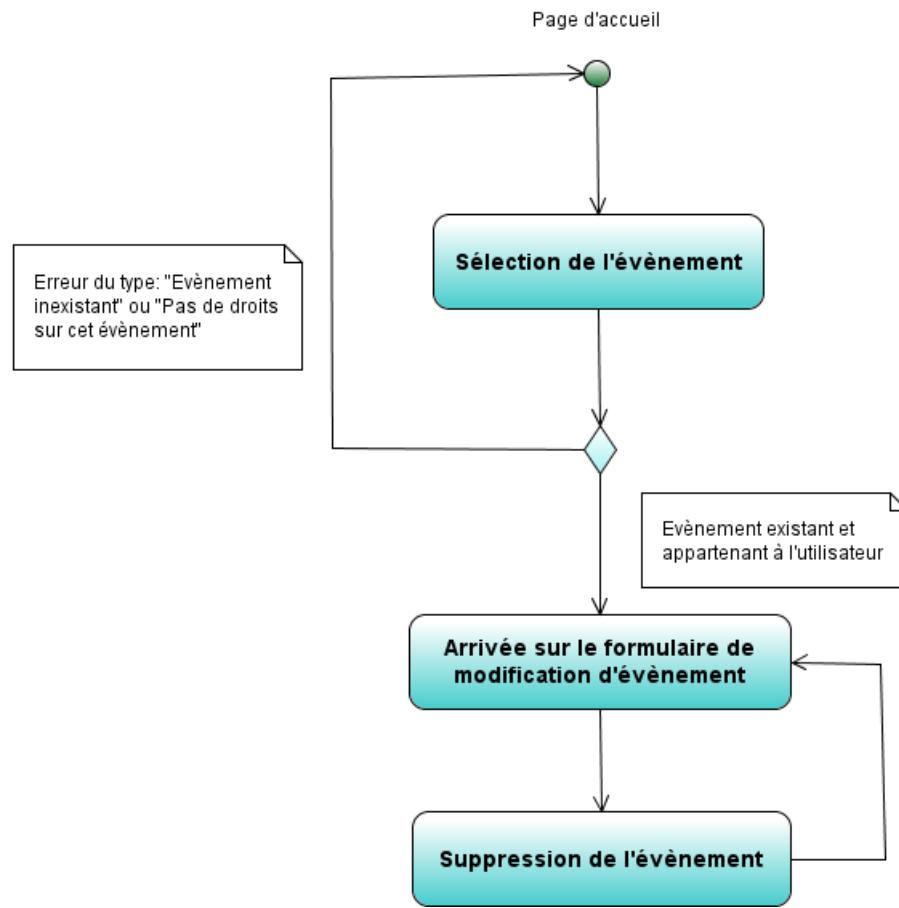
### 3.4.6 Suppression d'agenda

La suppression d'un agenda s'effectue de la façon suivante :

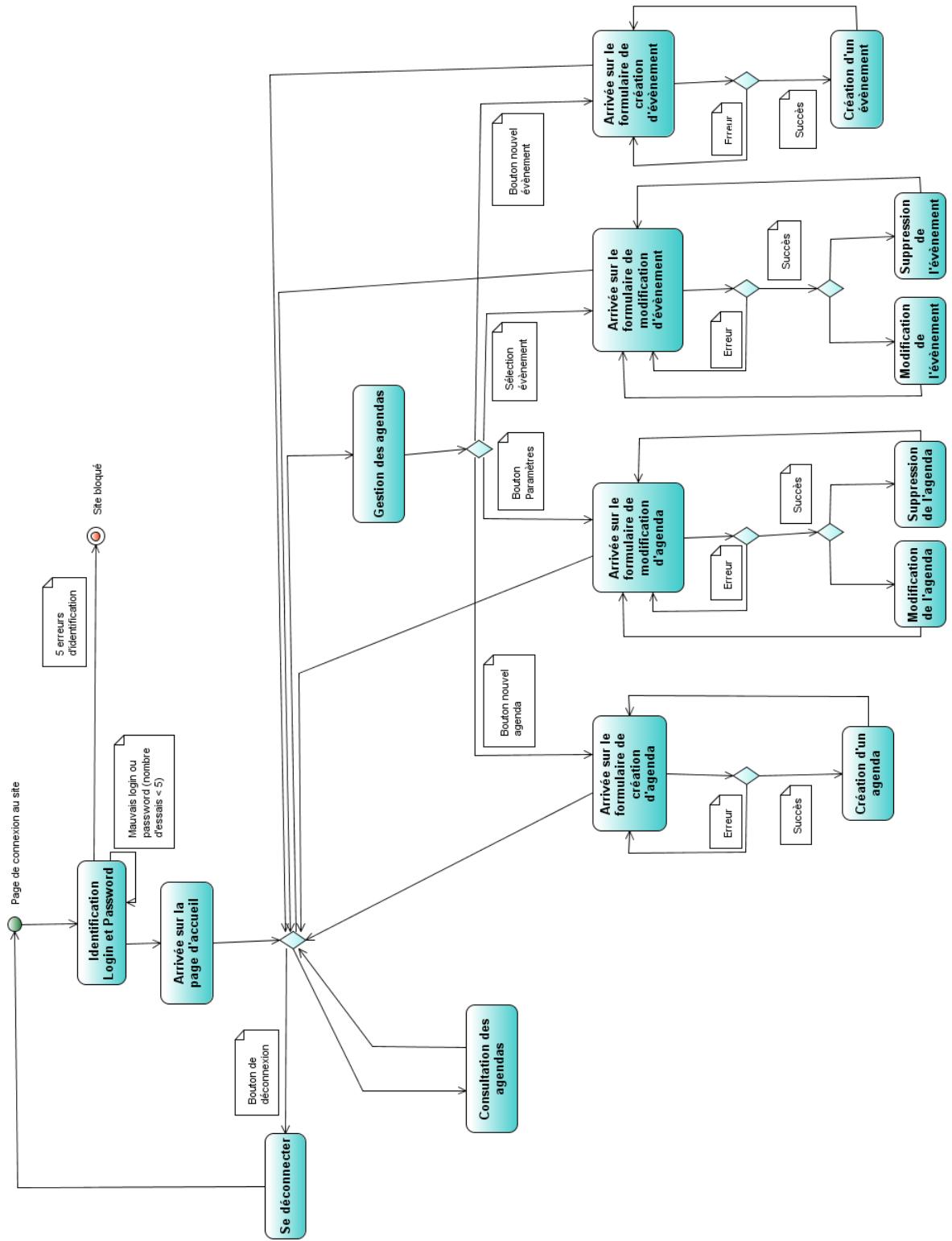


### 3.4.7 Suppression d'événement

Lorsqu'un utilisateur souhaite supprimer un événement, cela se passe de la manière suivante :



### 3.4.8 Diagramme d'activité global



## 3.5 Diagramme de séquences

Nous allons maintenant voir les différents diagrammes de séquences qui permettent de décrire les interactions entre les objets de notre projet.

### 3.5.1 Authentification

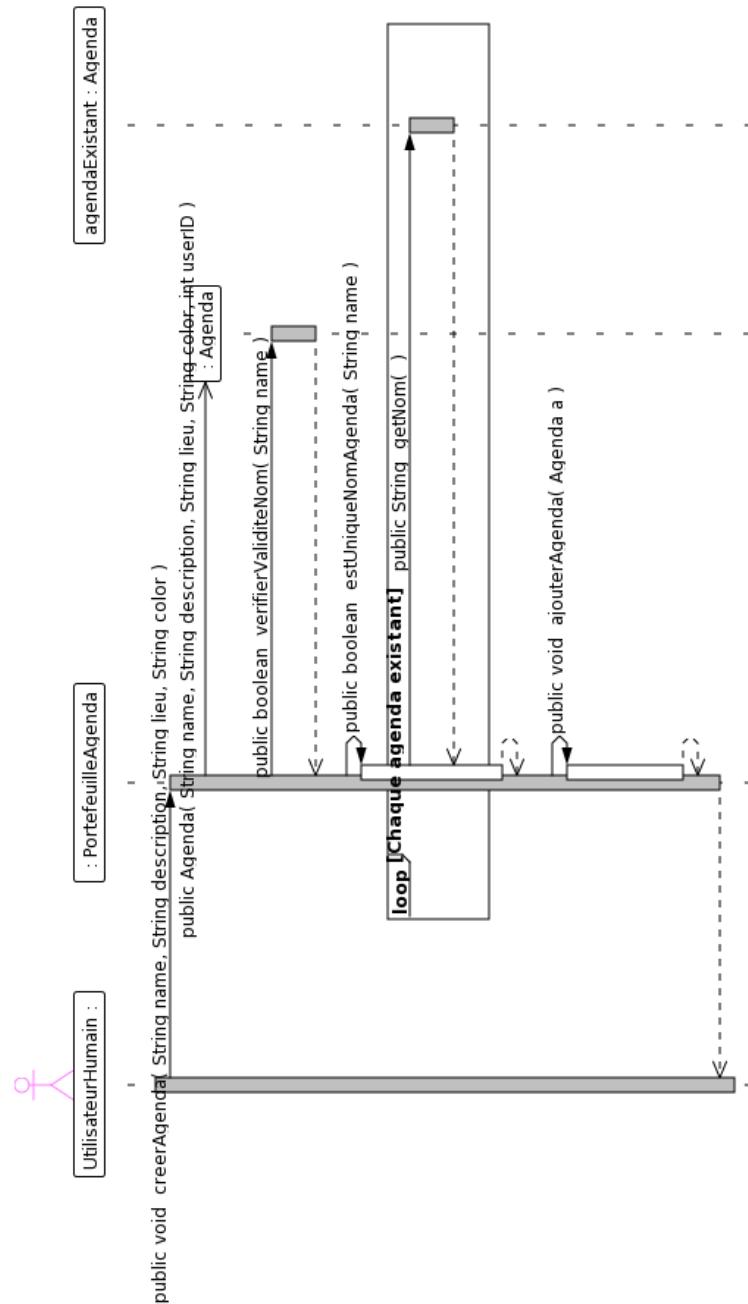
Ce premier diagramme définit l'authentification. Celle-ci est finalement très simple puisque qu'il s'agit d'une interrogation d'un objet UtilisateurDAO (DAO pour Data Access Object) qui permet de s'abstraire de la façon dont sont stockées les données référençant l'utilisateur. Dans le cas où l'utilisateur n'existe pas ou que le mot de passe ne correspond pas, un signal est envoyé par UtilisateurDAO.



### 3.5.2 Creer un agenda

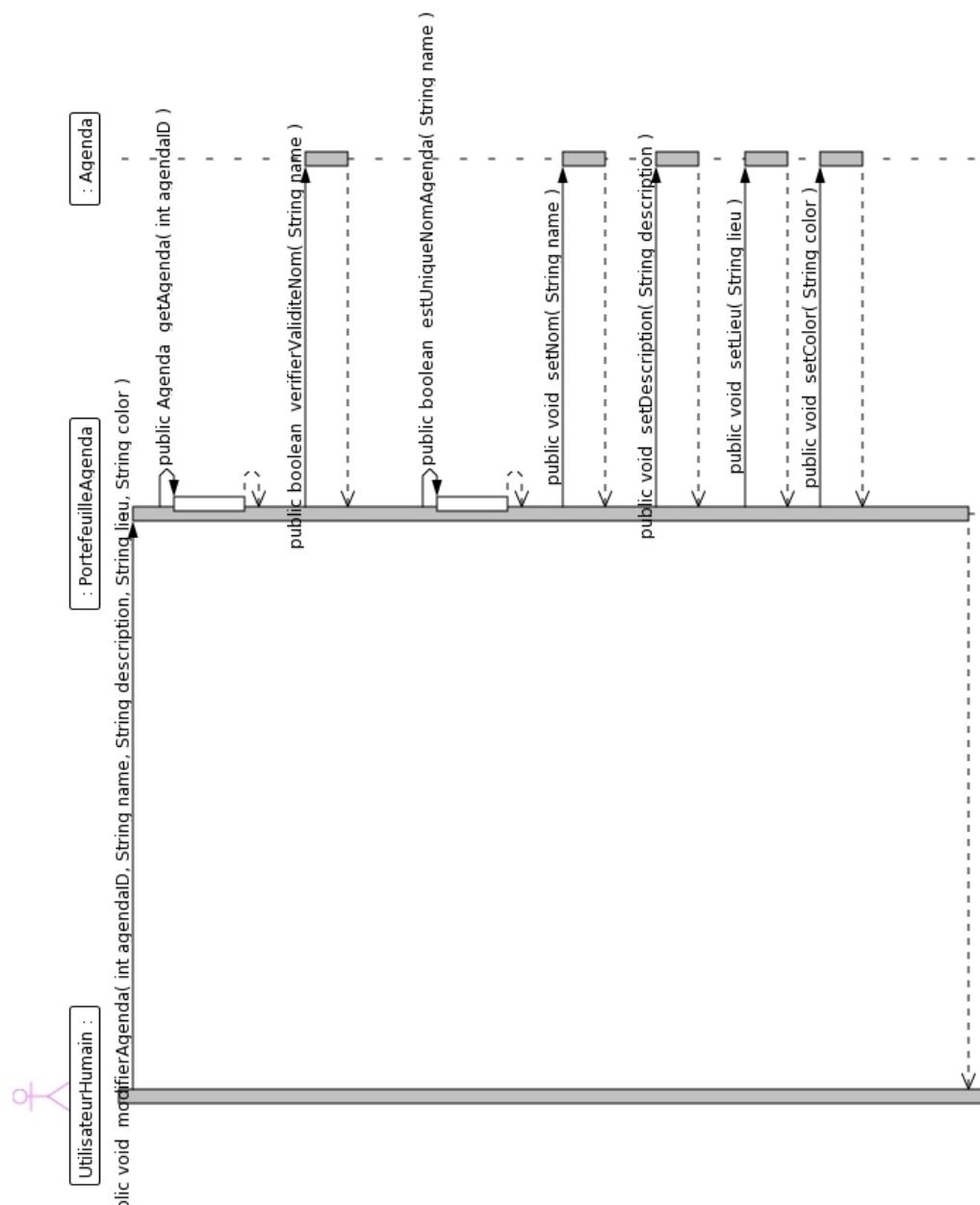
Un des points clé de notre projet est la création d'un agenda. Ce diagramme de séquence explique le déroulement de cette opération. Comme nous pouvons le voir ci-dessous, la création de l'agenda commence par l'ajout dans le portefeuille d'agenda de l'utilisateur qui vérifiera que le nom de l'agenda est valide et qu'il n'est pas déjà attribué. Si toutes les conditions sont bien remplies, l'agenda est alors ajouté au portefeuille.

Bien entendu la persistance de cette modification sera assurée par un objet PortefeuilleAgendaDAO qui appellera également des objets AgendaDAO et EventementDAO.



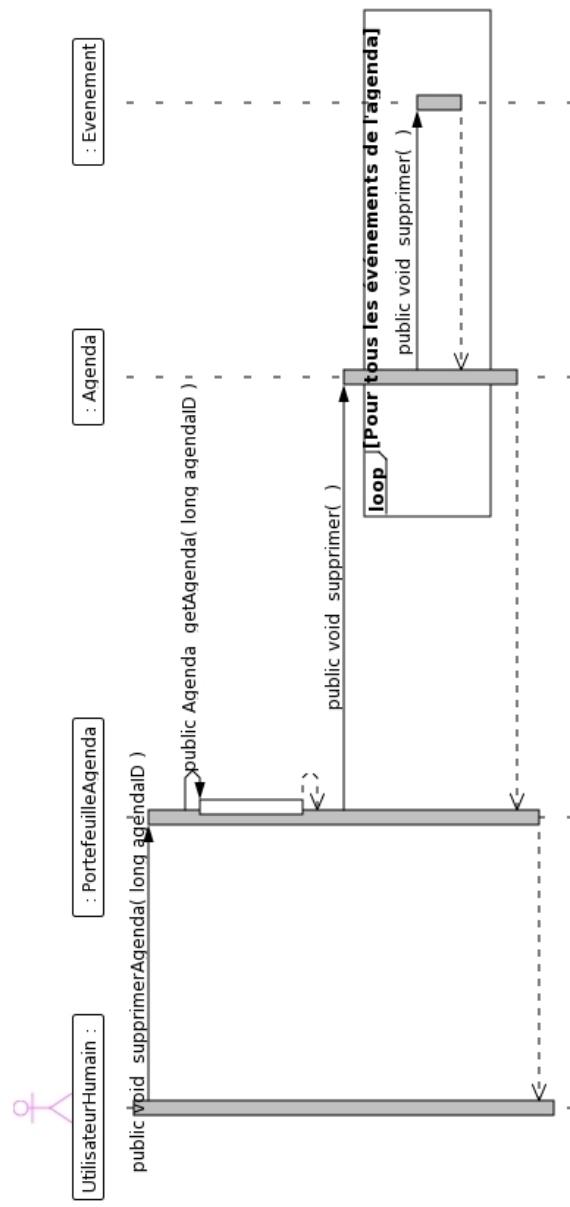
### 3.5.3 Modifier un agenda

Après avoir ajouté un agenda, l'utilisateur doit toujours être capable de le modifier. Ce diagramme explique le déroulement de cette opération. On constatera qu'il s'agit de récupérer l'agenda correspondant dans le portefeuille d'agenda puis de le modifier en vérifiant bien sur que les nouvelles données correspondent aux conditions décrites dans les scénarios.



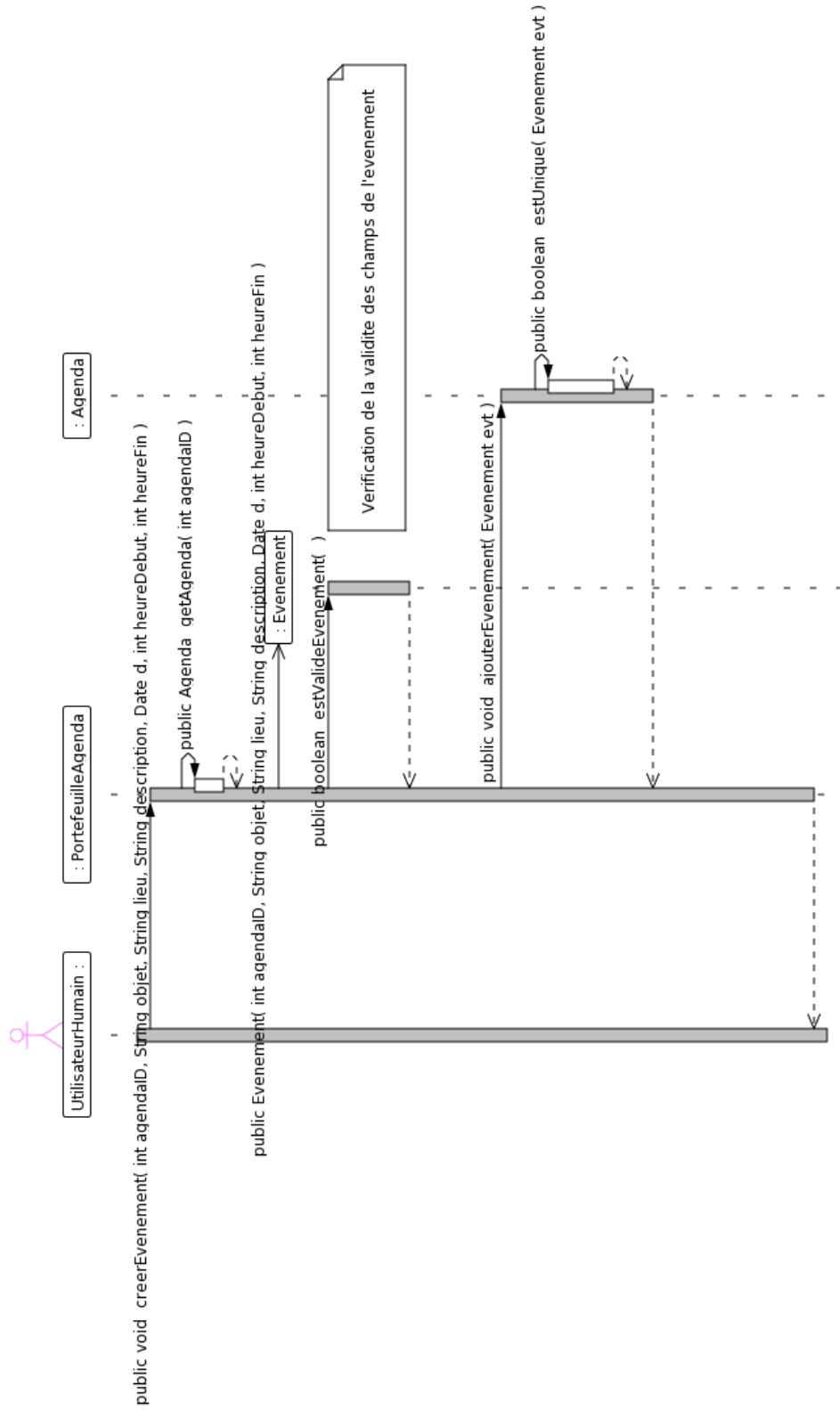
### 3.5.4 Supprimer un agenda

Un utilisateur peut également supprimer un agenda. Ce diagramme de séquence décrit le déroulement de cette opération.



### 3.5.5 Création d'un événement

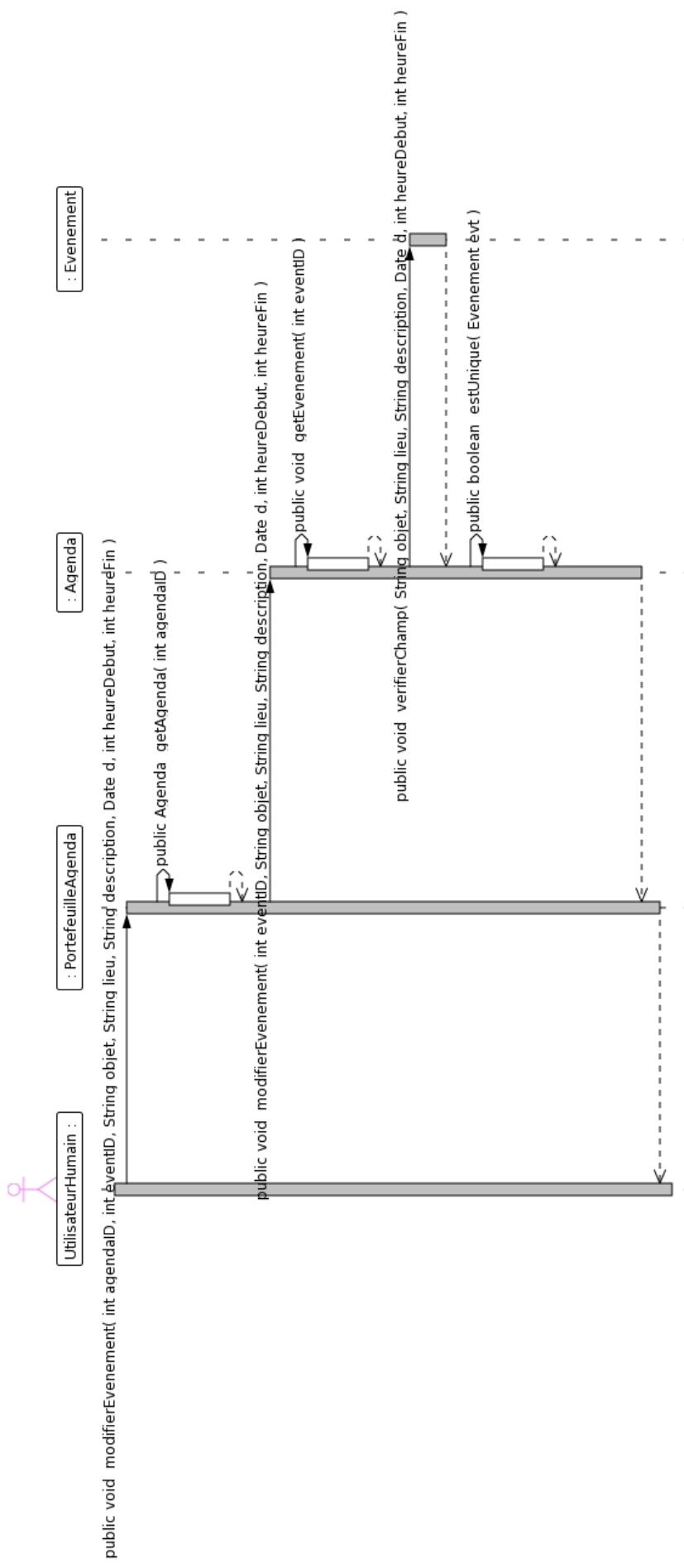
Une fois que l'utilisateur possède un ou plusieurs agendas, il peut ajouter des événements. L'ajout d'événement est décrit par le diagramme de séquence suivant :



Il faut souligner que dans un même agenda, il ne peut pas y avoir plusieurs événements sur un même créneau horaire. Nous avons choisi cette option pour des raisons de simplicité d'une part et parce que l'utilisateur peut très bien créer un autre événement simultané dans un autre agenda.

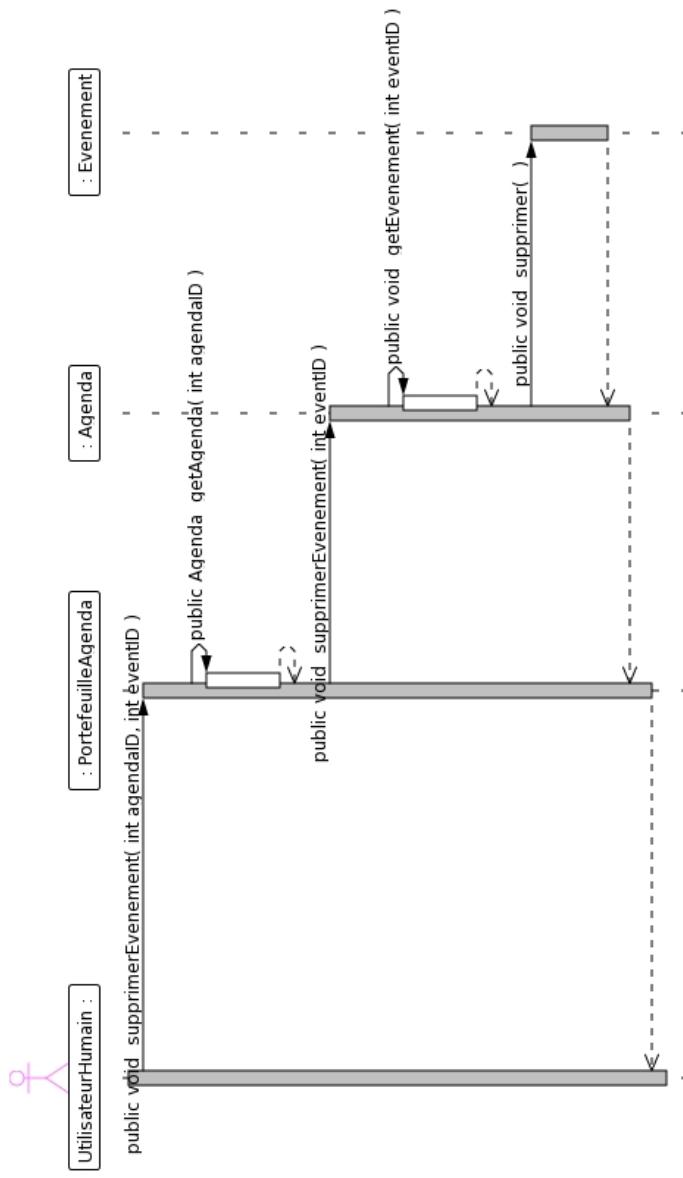
### 3.5.6 Modifier un événement

L'utilisateur peut bien évidemment modifier un événement qu'il a créé. Cette opération s'effectue selon le diagramme de séquence suivant :



### 3.5.7 Supprimer un événement

Finalement, un utilisateur peut supprimer un événement. Cette opération s'effectue selon le diagramme de séquence suivant :



## 3.6 Diagramme de classes

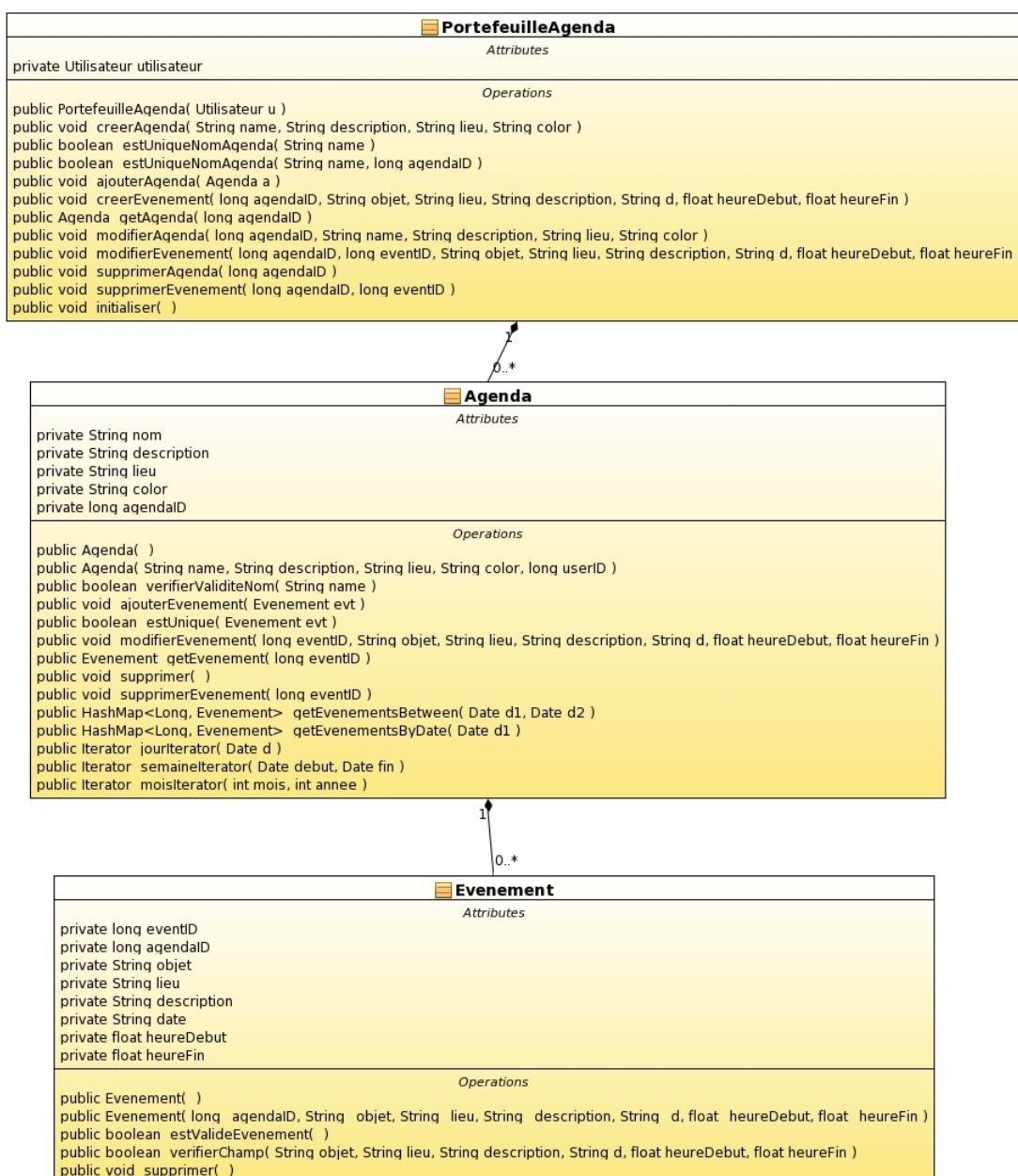
Nous allons maintenant voir la décomposition en différentes classes que nous avons choisie.

### 3.6.1 Gestionnaire d'agendas

Les premières classes sont directement en rapport avec le gestionnaire d'agenda. Il s'agit de nos classes métiers.

- PortefeuilleAgenda : associé à un utilisateur, il regroupe les agendas et permet des opérations telles que l'ajout, la modification et la suppression d'agendas et d'événements
- Agenda : associé à un utilisateur, il regroupe les différents événements ajoutés. Il permet l'ajout, la modification et la suppression d'événements. Un agenda est caractérisé par un nom, une description, une couleur et un lieu.
- Evenement : caractérisé par un objet, un lieu, une description et une date de début et de fin

Toutes ces classes sont regroupées dans le paquetage "Gestion agenda".

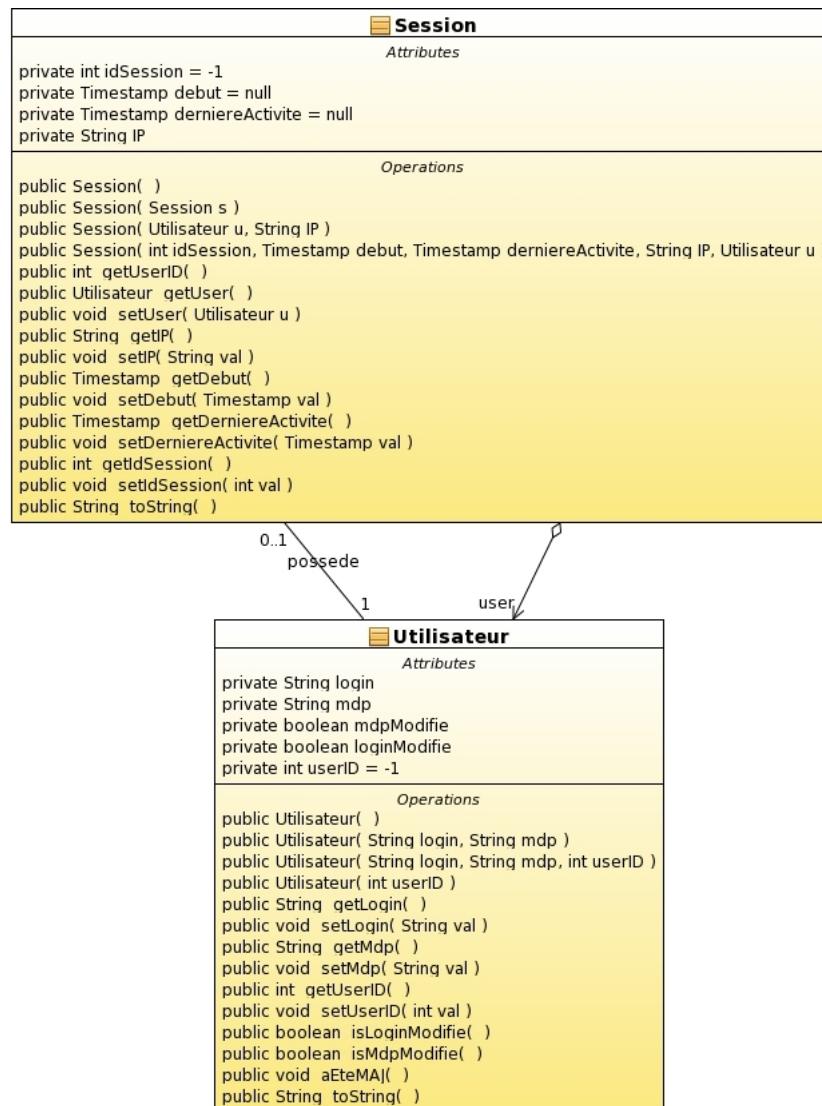


### 3.6.2 Authentification

Pour accéder à ses agendas, l'utilisateur doit s'authentifier à l'aide d'un couple pseudonyme (ou login) / mot de passe. Cette authentification est rendue persistante grâce à un système de Session qui intègre des options de sécurité (une session ne peut exister plus d'un certain temps, une session inactive trop longtemps n'est plus valide). D'où les deux classes :

- Utilisateur : caractérisé par un login, mot de passe et un identifiant unique
- Session : caractérisé par un identifiant unique, une date de début, une date de dernière activité, une adresse IP et bien évidemment un utilisateur

Ces classes sont regroupées dans le paquetage "Authentification" et sont réutilisables pour n'importe quel autre projet.



### 3.6.3 Service DAO

Enfin, pour assurer la persistance des objets nous avons créé des interfaces DAO qui permettent le mapping entre les objets manipulés par le programme et un système de stockage (comme une base de données, des fichiers XML). Il y a une classe DAO pour chaque classe que nous avons vu précédemment.

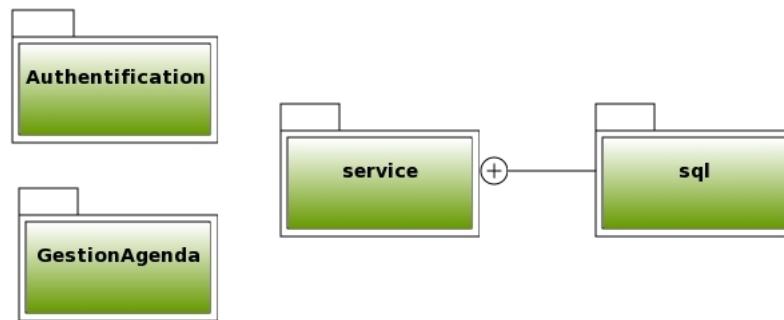
Ces interfaces sont regroupées dans le paquetage "Service".

Dans le cadre de notre projet, nous avons implémenté ces différentes interfaces pour réaliser un mapping objet/mySQL.

<pre>&lt;&lt;interface&gt;&gt; <b>SessionDAO</b> Attributes Operations public void cleanUp( ) public void insert( Session s ) public void update( Session s ) public void delete( Session s ) public Session findByUser( Utilisateur u, String IP )</pre>	<pre>&lt;&lt;interface&gt;&gt; <b>UtilisateurDAO</b> Attributes Operations public Utilisateur findByUserMdp( String login, String mdp ) public Utilisateur findById( int userID ) public boolean loginUtilise( String login ) public void insert( Utilisateur u ) public void update( Utilisateur u ) public void delete( Utilisateur u )</pre>
<pre>&lt;&lt;interface&gt;&gt; <b>PortefeuilleAgendaDAO</b> Attributes Operations public void save( PortefeuilleAgenda pa ) public void saveAgenda( Agenda a ) public void deleteAgenda( Agenda a ) public void updateAgenda( Agenda a ) public void deleteAll( PortefeuilleAgenda pa ) public void updateAll( PortefeuilleAgenda pa ) public PortefeuilleAgenda findByUser( Utilisateur user )</pre>	<pre>&lt;&lt;interface&gt;&gt; <b>AgendaDAO</b> Attributes Operations public void insert( Agenda a ) public void update( Agenda a ) public void delete( Agenda a ) public HashMap&lt;Long, Agenda&gt; findAll( ) public Agenda findByPrimaryKey( long agendaID ) public HashMap&lt;Long, Agenda&gt; findByUser( Utilisateur u )</pre>
<pre>&lt;&lt;interface&gt;&gt; <b>EvenementDAO</b> Attributes Operations public void insert( Evenement e ) public void update( Evenement e ) public void delete( Evenement e ) public HashMap&lt;Long, Evenement&gt; findAll( ) public Evenement findByPrimaryKey( long eventID ) public HashMap&lt;Long, Evenement&gt; findAfter( Date d ) public HashMap&lt;Long, Evenement&gt; findBefore( Date d ) public HashMap&lt;Long, Evenement&gt; findBetween( Date d1, Date d2 ) public HashMap&lt;Long, Evenement&gt; findByAgenda( long agendaID )</pre>	

### 3.6.4 Les différents paquetages

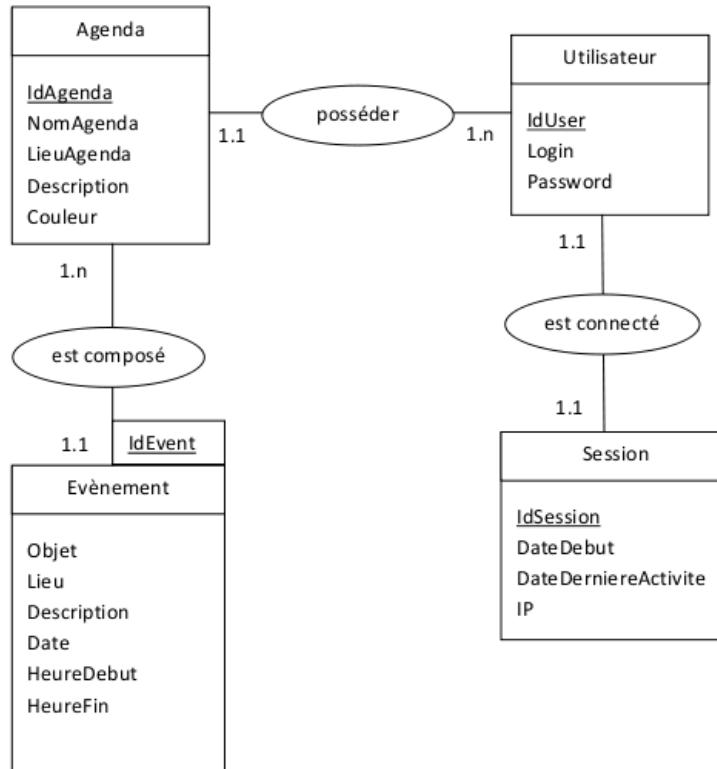
Voici les différents paquetages utilisés lors de notre projet. Le contenu des paquetages GestionAgenda, Authentification, et service a été décrit précédemment. Le paquetage sql contenu dans le paquetage service est l'implémentation des interfaces DAO pour le mapping objet/-mySQL.



## 3.7 Modèle E/A et relationnel de la base de données

### 3.7.1 Modèle entité/association

Pour modéliser notre base de données, nous nous sommes appuyés sur le modèle entité/association suivant :



Comme nous pouvons le voir, un utilisateur est connecté à une Session. En fait, cette session se comporte comme une persistance de l'authentification. Un utilisateur possède plusieurs agendas qui contiennent eux-mêmes des événements.

Nous allons maintenant passer à la transformation vers le modèle relationnel.

### 3.7.2 Modèle relationnel

#### Pour chaque entité, on crée une relation

Agenda(**IdAgenda**,NomAgenda,LieuAgenda,Description,Couleur)  
 Utilisateur(**IdUser**,Login,Password)  
 Session(**IdSession**,DateDebut,DateDerniereActivite,IP)

#### Entité faible

Evènement est une entité faible d'Agenda. On crée donc un mécanisme de clé étrangère pour référencer l'entité forte dans l'entité faible.

Evènement(**IdAgenda**,**IdEvent**,Objet,Lieu,Description,Date,HeureDebut,HeureFin)

#### Autres associations binaires

L'identifiant de l'utilisateur devient un attribut de la table Agenda.  
 L'identifiant de l'utilisateur devient un attribut de la table Session.

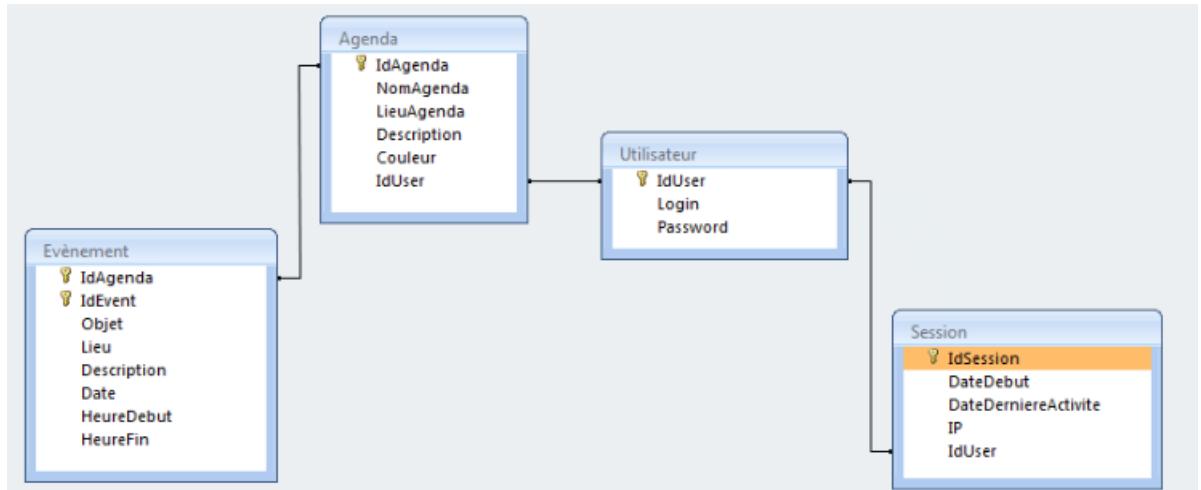
On obtient ainsi le modèle relationnel suivant :

Agenda(IdAgenda,NomAgenda,LieuAgenda,Description,Couleur,IdUser)

Utilisateur(IdUser,Login,Password)

Session(IdSession,DateDebut,DateDerniereActivite,IP,IdUser)

Évènement(IdAgenda,IdEvent,Objet,Lieu,Description,Date,HeureDebut,HeureFin)



## 4 Exemple de fonctionnement

### 4.1 Identification

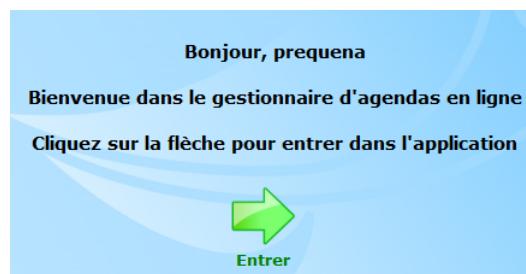
Lorsque l'utilisateur veut se connecter à l'application, il est dirigé vers la page d'identification du gestionnaire d'agendas.

L'identification se fait grâce à un nom d'utilisateur et un mot de passe stockés dans la base de données de l'application. L'utilisateur doit donc remplir les champs « Nom d'utilisateur » et « Mot de passe » avec ses identifiants et cliquer ensuite sur le bouton « Se connecter ».



#### 4.1.1 Authentification réussie

Si un utilisateur correspondant à ce couple login/mot de passe est enregistré dans la base et si aucune session n'est déjà ouverte pour cet utilisateur, l'authentification est réussie et l'utilisateur arrive sur la page de bienvenue :



Il doit ensuite cliquer sur le bouton « Entrer » pour finalement entrer dans le gestionnaire d'agendas.

#### 4.1.2 Erreurs d'authentification

Il existe plusieurs types d'erreurs d'authentification :

- Couple login/mot de passe inexistant
- 5 échecs de connexion
- Session déjà ouverte pour cet utilisateur

### Couple login/mot de passe inexistant

Si aucun utilisateur de la base ne correspond au nom d'utilisateur et au mot de passe, entrés dans les champs d'identification, un message d'erreur est affiché et l'utilisateur est redirigé vers la page d'authentification.



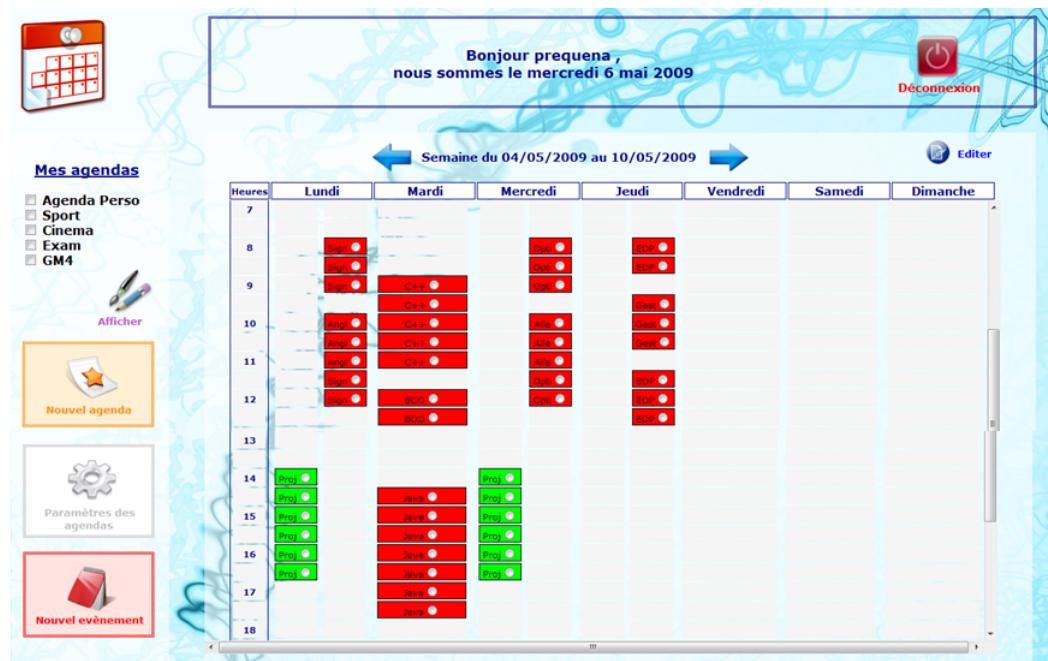
### 5 échecs de connexion

Au bout de 5 tentatives de connexion infructueuses, l'accès à l'application est bloqué. L'utilisateur ne peut plus accéder à la page de connexion pour des raisons de sécurité. Cette mesure évite que certaines personnes puissent tomber par hasard sur un couple login/mot de passe valide, en essayant de multiples valeurs et ainsi pirater le compte d'un autre utilisateur.



## 4.2 Page d'accueil : structure et fonctionnement

Une fois l'authentification réussie, l'utilisateur arrive sur la page d'accueil de l'application.



Cette page est composée de 3 parties :

### 4.2.1 Le menu horizontal

Ce menu est constitué du logo de l'application, et d'un cadre possédant :

- un message d'accueil pour l'utilisateur connecté
- une indication de la date du jour
- un bouton permettant de se déconnecter de l'application



### 4.2.2 Le menu vertical de gauche

Ce menu est composé de 2 parties :

- Une liste des agendas de l'utilisateur



Ce dernier peut choisir ceux qu'il souhaite voir apparaître dans son calendrier en les sélectionnant et en cliquant sur le bouton « Afficher ».

- Des boutons d'accès aux différentes pages de gestion des agendas et des évènements



Le bouton « Nouvel agenda » permet d'accéder au formulaire de création d'un agenda.  
Le bouton « Paramètres des agendas » permet d'accéder au formulaire de modification ou de suppression d'un agenda.

Le bouton « Nouvel évènement » permet d'accéder au formulaire de création d'un évènement.

#### 4.2.3 Le calendrier d'affichage des agendas

Ce calendrier permet à l'utilisateur connecté de consulter ces agendas par semaine.

Lorsque l'utilisateur se connecte à l'application, la semaine par défaut est la semaine correspondant à la date du jour. Mais il lui est ensuite possible de changer en utilisant les flèches bleues situées de part et d'autre de la semaine courante, qui permettent de passer à la semaine précédente ou à la semaine suivante.

Dans le calendrier, les agendas sont affichés sont ceux sélectionnés dans la liste « Mes agendas » du menu vertical de gauche (cf. paragraphe précédent). Pour plus de visibilité, chaque agenda a une couleur caractéristique. Par exemple, dans le screenshot ci-dessous, l'agenda « Cinéma » est caractérisé par la couleur bleu clair, et l'agenda « GM4 » est caractérisé par la couleur rouge.

Heures	Lundi	Mardi	Mercredi	Jeudi	Vendredi	Samedi	Dimanche
13							
14	Proj.		Proj.				
15	Proj.	Java	Proj.				
16	Proj.	Java	Proj.				
17	Proj.	Java	Proj.				
18							
19			Dans.		Cine		
20			Dans.		Cine		
21			Dans.		Cine		
22			Dans.		Cine		
23			Dans.		Cine		

Si l'utilisateur choisit de ne sélectionner que l'agenda « GM4 »,

#### Mes agendas

- Agenda Perso
- Sport
- Cinema
- Exam
- GM4



Afficher

son calendrier aura donc l'allure suivante :

Heures	Lundi	Mardi	Mercredi	Jeudi	Vendredi	Samedi	Dimanche
8	Sign.		Opt.	EDP.			
9	Sign.	C++.	Opt.	EDP.			
10	Angl.	C++.	Alle.	Gest.			
11	Angl.	C++.	Alle.	Gest.			
12	Sign.	BDD.	Opt.	EDP.			
13		BDD.	Opt.	EDP.			
14							
15		Java.					
16		Java.					
17		Java.					
18		Java.					
19		Java.					
20		Java.					
21		Java.					
22		Java.					
23		Java.					

Pour modifier un évènement, l'utilisateur doit le sélectionner en cochant le bouton radio correspondant et ensuite cliquer sur le bouton « Editer », situé en haut à droite du calendrier.

### 4.3 Créer un agenda

Pour créer un agenda, l'utilisateur doit cliquer sur le bouton « Nouvel Agenda » du menu vertical de gauche. Il est ensuite dirigé sur le formulaire suivant :

**Création d'un nouvel agenda**

**Nom :** Cours maths

**Lieu :**

**Description :** Cours de maths particulier pour Louise et Justine

**Couleur :**

Créer

Un agenda est caractérisé par les 4 informations suivantes :

- un nom
- un lieu d'application
- une description
- une couleur d'affichage

L'utilisateur doit donc renseigner ce formulaire de création et ensuite cliquer sur le bouton « Créeer » pour le valider.

#### 4.3.1 Crédation réussie

Si l'utilisateur a rempli correctement le formulaire, i.e. s'il a au moins rempli le champ « Nom » avec une valeur valide, la création est effectuée et un message d'information est affiché.

L'agenda a été créé.

#### 4.3.2 Erreurs de création

Si l'utilisateur n'a pas rempli le champ « Nom » du formulaire, la création est invalidée et un message d'erreur est affiché.

**ERREUR : Le champ Nom n'a pas été renseigné.**

Si l'utilisateur possède un autre agenda portant le même nom que celui inscrit dans le champ « Nom » du formulaire, la création est invalidée et un message d'erreur est affiché.

**ERREUR : Un agenda porte déjà ce nom.**

L'utilisateur peut à tout moment retourner sur la page d'accueil en cliquant sur le bouton « Retour à l'agenda ».

## 4.4 Modifier les paramètres d'un agenda

Pour modifier les paramètres d'un agenda, l'utilisateur doit cliquer sur le bouton « Paramètres des agendas » du menu vertical de gauche. Il est ensuite dirigé sur le formulaire suivant :

**Paramètres des agendas**

Agenda : GM4

Nom : GM4

Lieu : INSA de Rouen

Description : Cours de GM4

Couleur :

L'agenda a été modifié.

**Modifier** **Supprimer**

Il doit tout d'abord sélectionner l'agenda qu'il désire modifier en le sélectionnant dans la liste déroulante « Agenda » et en cliquant ensuite sur la flèche verte. Les caractéristiques de l'agenda sélectionné sont ainsi affichées dans les différents champs du formulaire.

### 4.4.1 Suppression de l'agenda

Pour supprimer l'agenda sélectionné, l'utilisateur doit cliquer sur le bouton « Supprimer » situé en bas à droite du formulaire. Cette action a pour conséquence de supprimer l'agenda de la base de données, ainsi que tous les événements relatifs à cet agenda.

**Paramètres des agendas**

Agenda : Agenda Perso

Nom :

Lieu :

Description :

Couleur :

L'agenda a été supprimé.

**Modifier** **Supprimer**

### 4.4.2 Modification des caractéristiques de l'agenda

Pour modifier les caractéristiques de l'agenda sélectionné l'utilisateur doit renseigner les champs du formulaire avec les informations voulues et cliquer ensuite sur le bouton « Modifier ». Si le

formulaire a été correctement rempli, la modification est validée et un message d'information est affiché.

L'agenda a été modifié.

**NB :** La modification est invalidée dans le cas d'erreurs de même type que lors de la création d'un agenda, à savoir le champ « Nom » non rempli, ou un agenda portant déjà le nom inscrit dans le champ « Nom ».

## 4.5 Crée un événement

Pour créer un nouvel événement, l'utilisateur doit cliquer sur le bouton « Nouvel Evènement » du menu vertical de gauche. Il est ensuite dirigé sur le formulaire suivant :

The screenshot shows a web-based application for creating events. The main title is "Création d'un évènement". The form fields are as follows:

- Objet : Entretien Job Ete
- Date : 2009-05-11 Format: AAAA-MM-JJ
- Lieu : Cleres
- Heure de début : 14 : 0
- Heure de fin : 15 : 0
- Agenda : Agenda Perso
- Description : (empty text area)

On the left side, there is a "Retour à l'agenda" link with a small icon of a document with a pencil. On the right side, there is a "Créer" button with a green checkmark icon.

Un évènement est caractérisé par les 7 informations suivantes :

- L'agenda auquel il appartient
- Un objet
- Une date d'application
- Un lieu d'application
- Une heure de début
- Une heure de fin
- Une description

L'utilisateur doit donc renseigner ce formulaire de création et ensuite cliquer sur le bouton « Créer » pour le valider.

### 4.5.1 Crédit réussie

Si l'utilisateur a rempli correctement le formulaire, i.e. s'il a au moins rempli les champs « Agenda », « Objet », « Date », « Heure de début » et « Heure de fin » avec des valeurs valides, la création est effectuée et un message d'information est affiché.

L'évènement a été créé.

#### 4.5.2 Erreurs de création

Le premier type d'erreur de création correspond à un mauvais remplissage des champs par l'utilisateur, à savoir :

- Champ « Objet » non rempli
- Date invalide
- Heures de début de fin invalides : heure de début supérieure à l'heure de fin

Dans l'un de ces 3 cas, la création est invalidée et un message d'erreur est affiché.

ERREUR : Champs mal renseignés.

Le second type d'erreur de création correspond à l'existence dans l'agenda sélectionné, d'un évènement simultané, i.e. possédant une date et des heures de début et de fin identiques. Dans ce cas, la création est invalidée et un message d'erreur est affiché.

ERREUR : Evènement simultané existant.

### 4.6 Modifier les paramètres d'un événement

Pour modifier les paramètres d'un évènement, l'utilisateur doit le sélectionner dans le calendrier de la page d'accueil et cliquer sur le bouton « Editer ». Il est ensuite dirigé sur le formulaire suivant :

**Modification d'un évènement**

**Objet :** C++

**Date :** 2009-05-05 Format: AAAA-MM-JJ

**Lieu :** INSA

**Heure de début :** 9  0

**Heure de fin :** 11  30

**Description :**

**Retour à l'agenda**

**Modifier** **Supprimer**

Les caractéristiques de l'évènement sélectionné sont affichées dans les différents champs du formulaire.

**NB :** L'utilisateur ne peut pas déplacer son évènement d'un agenda à l'autre. C'est pour cette raison que le champ « Agenda » n'apparaît pas dans ce formulaire. Pour déplacer un évènement vers un autre agenda, l'utilisateur doit le supprimer du premier agenda, et ensuite le recréer dans un autre agenda.

#### 4.6.1 Suppresion de l'événement

Pour supprimer l'évènement sélectionné, l'utilisateur doit cliquer sur le bouton « Supprimer » situé en bas à droite du formulaire. Cette action a pour conséquence de supprimer l'évènement de la base de données.

L'évènement a été supprimé.

#### 4.6.2 Modification des caractéristiques de l'événement

Pour modifier les caractéristiques de l'évènement sélectionné, l'utilisateur doit renseigner les champs du formulaire avec les informations voulues et cliquer ensuite sur le bouton « Modifier ». Si le formulaire a été correctement rempli, la modification est validée et un message d'information est affiché.

L'évènement a été modifié.

**NB :** La modification est invalidée dans le cas d'erreurs de même type que lors de la création d'un évènement, à savoir un ou plusieurs champs mal renseignés, ou un évènement simultané existant.

## 5 Pour aller plus loin : exemple d'utilisation du RMI

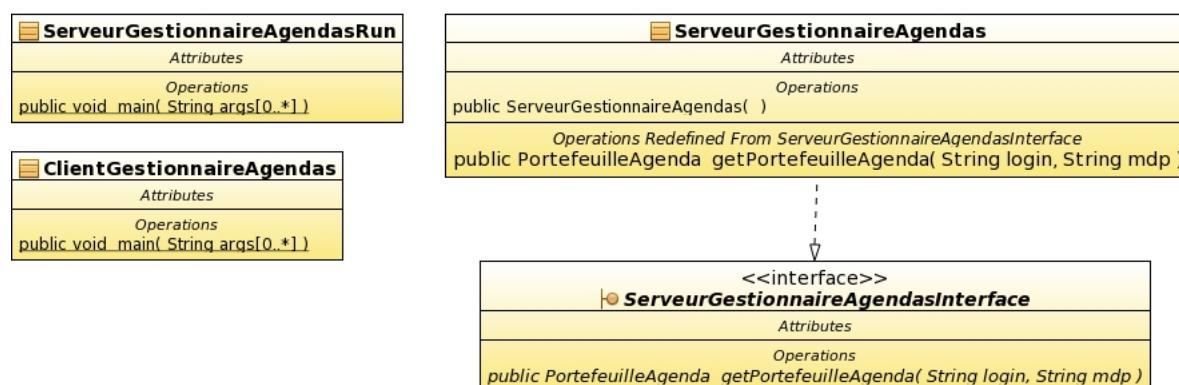
### 5.1 Exemple envisagé

Afin de nous assurer de la validité de notre modélisation, nous avons voulu changer notre point de vue sur le gestionnaire d'agenda. En plus de l'accès par une interface web en JSP/xHTML, nous nous sommes dits qu'il serait intéressant que l'utilisateur puisse accéder à ces agendas depuis un programme. C'est comme ça que nous est venu l'idée d'implémenter un petit exemple en RMI. Ce petit exemple n'existe que pour démontrer la faisabilité de la chose et non pas pour implémenter un logiciel complet.

Notre exemple est composé des éléments suivants :

- Un serveur distant : il se charge de mettre à disposition un objet distant qui permet d'accéder aux agendas d'un utilisateur quand on lui précise un login et un mot de passe. Une fois qu'on lui donne un login et un mot de passe valide, il se connecte à la base de données mySQL (à l'aide des objets du paquetage service.sql) pour récupérer le portefeuille d'agenda de l'utilisateur.
- un client : qui se connecte et récupère un portefeuille d'agenda d'un utilisateur

Voici les différentes classes utilisées :



### 5.2 Retranscription du fonctionnement

Le petit exemple que nous avons programmé va faire les deux choses suivantes :

- Il va essayer de récupérer un portefeuille d'agenda avec un couple login/mot de passe invalide
- Il va essayer de récupérer un portefeuille d'agenda avec un couple login/mot de passe valide

Voici la sortie du programme :

```

9 mai 2009 16:18:14 remote.ClientGestionnaireAgendas main
GRAVE: null
Exception.UtilisateurInexistantException
|   at service.sql.UtilisateurSQL.findByUserMdp(UtilisateurSQL.java:81)
|   at remote.ServeurGestionnaireAgendas.getPortefeuilleAgenda(ServeurGestionnaireAgendas.java:32)
|   at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
|   at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:57)
|   at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)
|   at java.lang.reflect.Method.invoke(Method.java:616)
|   at sun.rmi.server.UnicastServerRef.dispatch(UnicastServerRef.java:322)
|   at sun.rmi.transport.Transport$1.run(Transport.java:177)
|   at java.security.AccessController.doPrivileged(Native Method)
|   at sun.rmi.transport.Transport.serviceCall(Transport.java:173)
|   at sun.rmi.transport.tcp.TCPTTransport.handleMessages(TCPTTransport.java:553)
|   at sun.rmi.transport.tcp.TCPTTransport$ConnectionHandler.run0(TCPTTransport.java:808)
|   at sun.rmi.transport.tcp.TCPTTransport$ConnectionHandler.run(TCPTTransport.java:667)
|   at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1110)
|   at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:603)
|   at java.lang.Thread.run(Thread.java:636)
|   at sun.rmi.transport.StreamRemoteCall.exceptionReceivedFromServer(StreamRemoteCall.java:273)
|   at sun.rmi.transport.StreamRemoteCall.executeCall(StreamRemoteCall.java:251)
|   at sun.rmi.server.UnicastRef.invoke(UnicastRef.java:160)
|   at java.rmi.server.RemoteObjectInvocationHandler.invokeRemoteMethod(RemoteObjectInvocationHandler.java:195)
|   at java.rmi.server.RemoteObjectInvocationHandler.invoke(RemoteObjectInvocationHandler.java:149)
|   at $Proxy0.getPortefeuilleAgenda(Unknown Source)
|   at remote.ClientGestionnaireAgendas.main(ClientGestionnaireAgendas.java:35)

PORTEFEUILLE AGENDA :
Utilisateur : prequena
-----
Agenda : Agenda Perso
        (objet)Lol ; (lieu)uuuuu ; (description)ytdrttsy
        (objet)uuuuu ; (lieu)iiiiii ; (description)kkk
        (objet)Demenagement ; (lieu)Cours Clemenceau ; (description)blabla

Agenda : Sport
Agenda : Cinema
Agenda : Cours de maths
Agenda : Exam
Agenda : ad

```

Nous allons maintenant expliquer cette sortie.

Dans la première partie, nous pouvons constater qu'une exception de type "UtilisateurInexistantException" est lancée.

```

GRAVE: null
Exception.UtilisateurInexistantException
|   at service.sql.UtilisateurSQL.findByUserMdp(UtilisateurSQL.java:81)
|   at remote.ServeurGestionnaireAgendas.getPortefeuilleAgenda(ServeurGestionnaireAgendas.java:32)
|   at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
|   at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:57)
|   at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)
|   at java.lang.reflect.Method.invoke(Method.java:616)
|   at sun.rmi.server.UnicastServerRef.dispatch(UnicastServerRef.java:322)
|   at sun.rmi.transport.Transport$1.run(Transport.java:177)
|   at java.security.AccessController.doPrivileged(Native Method)
|   at sun.rmi.transport.Transport.serviceCall(Transport.java:173)
|   at sun.rmi.transport.tcp.TCPTTransport.handleMessages(TCPTTransport.java:553)
|   at sun.rmi.transport.tcp.TCPTTransport$ConnectionHandler.run0(TCPTTransport.java:808)
|   at sun.rmi.transport.tcp.TCPTTransport$ConnectionHandler.run(TCPTTransport.java:667)
|   at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1110)
|   at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:603)
|   at java.lang.Thread.run(Thread.java:636)
|   at sun.rmi.transport.StreamRemoteCall.exceptionReceivedFromServer(StreamRemoteCall.java:273)
|   at sun.rmi.transport.StreamRemoteCall.executeCall(StreamRemoteCall.java:251)
|   at sun.rmi.server.UnicastRef.invoke(UnicastRef.java:160)
|   at java.rmi.server.RemoteObjectInvocationHandler.invokeRemoteMethod(RemoteObjectInvocationHandler.java:195)
|   at java.rmi.server.RemoteObjectInvocationHandler.invoke(RemoteObjectInvocationHandler.java:149)
|   at $Proxy0.getPortefeuilleAgenda(Unknown Source)
|   at remote.ClientGestionnaireAgendas.main(ClientGestionnaireAgendas.java:35)

```

Effectivement, nous essayons de récupérer un portefeuille d'agenda d'un utilisateur soit inexistant, soit le mot de passe donné est invalide.

Ensuite nous essayons de récupérer le portefeuille d'agenda d'un utilisateur existant avec le bon mot de passe. Nous l'obtenons et l'affichons dans le terminal :

```

PORTEFEUILLE AGENDA :
Utilisateur : prequena
-----
Agenda : Agenda Perso
        (objet)Lol ; (lieu)uuuuu ; (description)ytdrttsy
        (objet)uuuuu ; (lieu)iiiiii ; (description)kkk
        (objet)Demenagement ; (lieu)Cours Clemenceau ; (description)blabla

Agenda : Sport
Agenda : Cinema
Agenda : Cours de maths
Agenda : Exam
Agenda : ad

```

### **5.3 Conclusion que nous pouvons tirer de cet exemple**

Cet exemple simpliste démontre qu'il serait vraiment facile de développer une application distante pour consulter et modifier ses agendas. L'utilisateur aurait alors le choix d'utiliser l'interface web, une application fenêtée ou bien sur les deux.

## 6 Conclusion

Ce projet a été l'occasion de découvrir de nouvelles technologies en Java. Effectivement, nous avions déjà eu l'occasion dans le passé d'utiliser les socket en Java notamment grâce à l'ancienne UV I4 pendant notre premier cycle. C'est la raison pour laquelle nous nous sommes orientés vers le JSP et les servlet que nous ne connaissions pas. Il faut avouer que ce projet a été une excellente occasion de nous plonger dans le côté "web" de Java.

Nous avons pu découvrir la puissance offerte par les JSP qui permettent de coupler à la fois du développement Java classique et un langage script. Cela nous a permis de développer notre partie métier avec du Java "classique" et la partie "entrée/sortie" avec le JSP et l'xHTML. Ensuite, nous avons pu découvrir les outils qui existent en Java pour interagir avec une base de données MYSQL.

Finalement, nous avons pu découvrir la puissance de Java pour les applications distribuées avec notre exemple qui fait intervenir le Java RMI. En effet, après avoir développé notre application web de gestion d'agendas, nous avons pu constater qu'il était vraiment simple de réutiliser ce que nous avions déjà développé pour une application distante.

Finalement, ce projet a été une bonne occasion d'appliquer ce que nous avons vu en cours de Java distribué et de génie logiciel. Nous avons eu un bon aperçu des possibilités du Java côté serveur qui complète ainsi notre vision de ce langage.