

Mise en place d'un site de réservation de salles

-

**Rapport sur notre démarche d'analyse
pour la conception du site web avec
l'architecture Objet**

Rapport réalisé par :

Katia TALBI
Tahrat Anis
HAMMI Thilleli

Maîtrise d'ouvrage:

Université Paris-Saclay

Responsable de la matière:

Wolff Burkhart

Sommaire :

I. Objectifs de la conception :	3
Types de langages de programmation :	3
Langages clients	3
Langages serveurs	3
Types de fonctions :	4
II. La conception en UML :	4
1- Diagramme de classe :	5
2- Diagrammes de cas d'utilisation :	6
Diagramme cas d'utilisation : authentification:	7
Diagramme cas d'utilisation :Création de compte :	7
Diagramme cas d'utilisation :Demande de réservation:	8
Diagramme cas d'utilisation :Gestion Salle:	9
Diagramme cas d'utilisation : Gestion Système:	10
3- Diagrammes de séquences :	10
sequence-diagram : Utilisateur qui se connecte sur notre site web	11
sequence-diagram : Étudiant qui s'enregistre sur notre site web et qui devient membre	12
sequence-diagram : Professeur qui s'enregistre sur notre site web devenant membre	13
sequence-diagram : Creation d'un Compte Association au niveau du serveur	14
sequence-diagram : d'un utilisateur qui formule une demande de reservation	14
4- Diagrammes d'états :	15
Le diagramme d'état concernant les deux états possibles d'une salle :	15
Le diagramme d'état concernant les deux états possibles d'une réservation :	15
5- Diagrammes d'objets :	16
Professeur :	16
Une demande de réservation élémentaire:	16
Une demande de réservation élémentaire avec une alternative :	17
Deux demandes de réservation élémentaires dont l'une avec une alternative :	18
Association	19
Création d'un nouveau compte association	19
Gestionnaire	20
Demande de réservation exceptionnelle par le gestionnaire:	20
Création de nouvelle salle par le gestionnaire:	21
6- Diagramme d'activité: déroulement d'une demande :	22
II. La conception en détail	22
1- Problèmes + patterns de conception:	22
Pseudo-code :	22
Bibliothèques :	25

2- Attributs d'objets et sémantique:	26
Utilisateur:	26
Gestionnaire:	28
Association:	30
Etudiant:	31
Professeur :	32
Pôle:	33
Réservation:	33
Demande Elémentaire:	34
Equipement:	36
Salle:	37
Disponibilité :	38
Créneau:	38
Planning :	38
EDT:	38
Systeme :	39
3- Base de données:	41
Stockage persistant :	41
IV. Interfaces web:	42
Page principale :	42
Compte Utilisateur :	42
Inscription étudiant:	44
Inscription professeur/association:	45

I.Objectifs de la conception :

Dans cette phase de conception, nous allons apporter plus de détails et de précision à la solution que nous avons donné dans la phase d'analyse précédente et nous allons clarifier les aspects techniques, tels que l'installation des différentes parties logicielles à installer sur du matériel ainsi que le fonctionnement interne de notre futur site web.

Pour obtenir un système réellement évolutif nous avons ces objectifs de base :

- La modularité : pour faciliter la réutilisation des packages et des classes.
- La maintenabilité : en créant des packages et des classes d'une dimension humaine.
- La simplicité : parce que c'est un gage de réussite.

Pour se faire, nous avons chercher les meilleurs outils, en ce qui concerne la programmation, qui est la partie dans laquelle nous réaliseront le site web à l'aide de langages de programmation, de systèmes de gestion de bases de données, etc. Il va y avoir deux types de machines connectées à notre site web: le client et le serveur.

a. Types de langages de programmation :

Pour construire notre site web, nous allons faire recours au JavaScript principalement car nous voulons un outil très fluide et très rapide qui communique beaucoup avec le serveur, JavaScript avec Node.js est le mieux placé par rapport a PHP ou Python par exemple. en plus de ça nous allons avoir :

- Langages clients

langages clients (langages FrontEnd lus par la machine du client) suivants:

- HTML
- CSS
- JAVASCRIPT

- Langages serveurs

Ainsi que les langages serveurs (décrivant comment le site doit se comporter) :

- Node.js : nous permet d'utiliser JAVASCRIPT sur le serveur :
JavaScript avait toujours été utilisé du côté du client (du côté du

visiteur qui navigue sur notre site). Le navigateur web du visiteur (Firefox, Chrome, IE...) exécute le code JavaScript et effectue des actions sur la page web. Node.js offre un environnement côté serveur qui permet aussi d'utiliser le langage JavaScript pour générer des pages web.

- JQuery : nous allons pouvoir intégrer quelques notions de jQuery qui vont nous permettre de faire évoluer notre serveur HTTP vers un serveur plus générique. Car cet outil est devenu incontournable dans la manipulation de la structure HTML

b. Types de fonctions :

Deux type de fonctions: les fonctions côté serveur, et les fonctions côté client:

- Les fonctions que nous allons présenter dans notre diagramme de classes dont nous allons parler plus loin dans notre conception, sont les fonctions back-end, que nous allons utiliser du côté serveur. Elles vont gérer tous les calculs faits sur les informations collectées au niveau front-end.
- Les autres fonctions, du côté client, sont celles qui vont permettre de collecter les informations clients et leurs demandes.

II. La conception en UML :

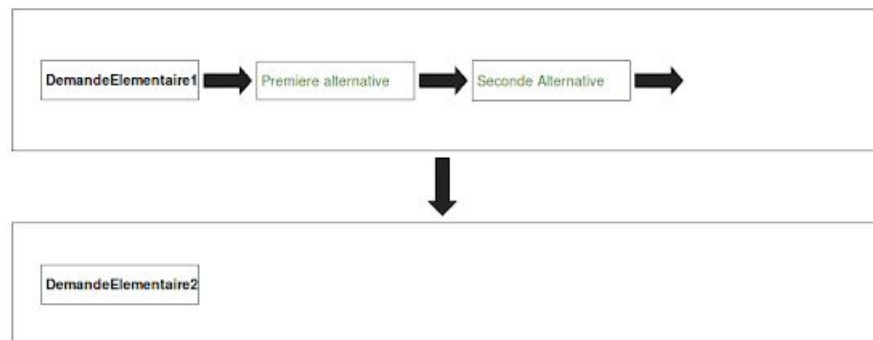
Dans cette partie nous utilisons l'UML "Unified Modeling Language" ou Langage de Modélisation Unifié en français, un langage visuel constitué d'un ensemble de diagrammes pour représenter le site web que nous allons développer : son fonctionnement, sa mise en route, les actions susceptibles d'être effectuées par le logiciel, etc.

Une réservation est un ensemble de demandes élémentaires d'un même utilisateur qui peut être satisfaite par le système. Elle contient les demandes élémentaires dans



une liste dans laquelle chaque noeud contient une liste de DemandeElementaire, qui représente les cas suivants :

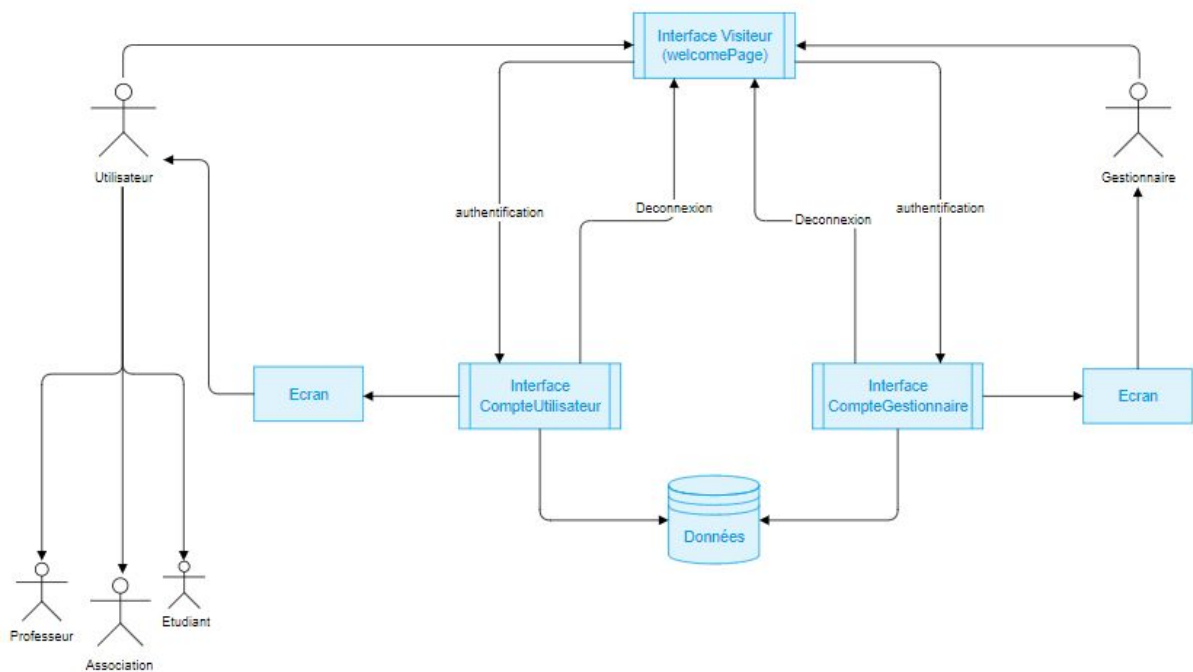
- Une demande élémentaire et ses alternatives appartiennent à la même liste de la liste. Dans quel cas la réservation est satisfaite si la demande ou l'une de ses alternatives l'est aussi.
- Deux demandes différentes faites dans la même réservation appartiennent à deux listes différentes, dans quel cas la réservation est satisfaite si les deux demandes sont satisfaites. C'est à dire, si la première demande ou l'une de ses alternatives est satisfaite, et la deuxième demande ou l'une de ses alternatives est satisfaite.



- Un utilisateur (sauf le gestionnaire), peut donc faire plusieurs demandes avec plusieurs alternatives pour toute demande, tant qu'il ne dépasse pas son nombre maximum de réservations autorisé.
- Le Planning représente le planning d'une salle sur la période d'une semaine, où chaque jour est représenté dans le tableau Jours de taille 5 : du lundi (première case du tableau) au vendredi (dernière case). Et pour chaque jour il y'a la liste des créneaux où la salle est réservée.
- L'EDT c'est l'emploi du temps de toutes les salles. Il regroupe donc les plannings des salles dans edtDeSalles, qui est une liste de Planning.
- Tout utilisateur (sauf le gestionnaire) peut annuler une de ses demandes de réservation tant qu'elle n'est pas encore traitée par le système.

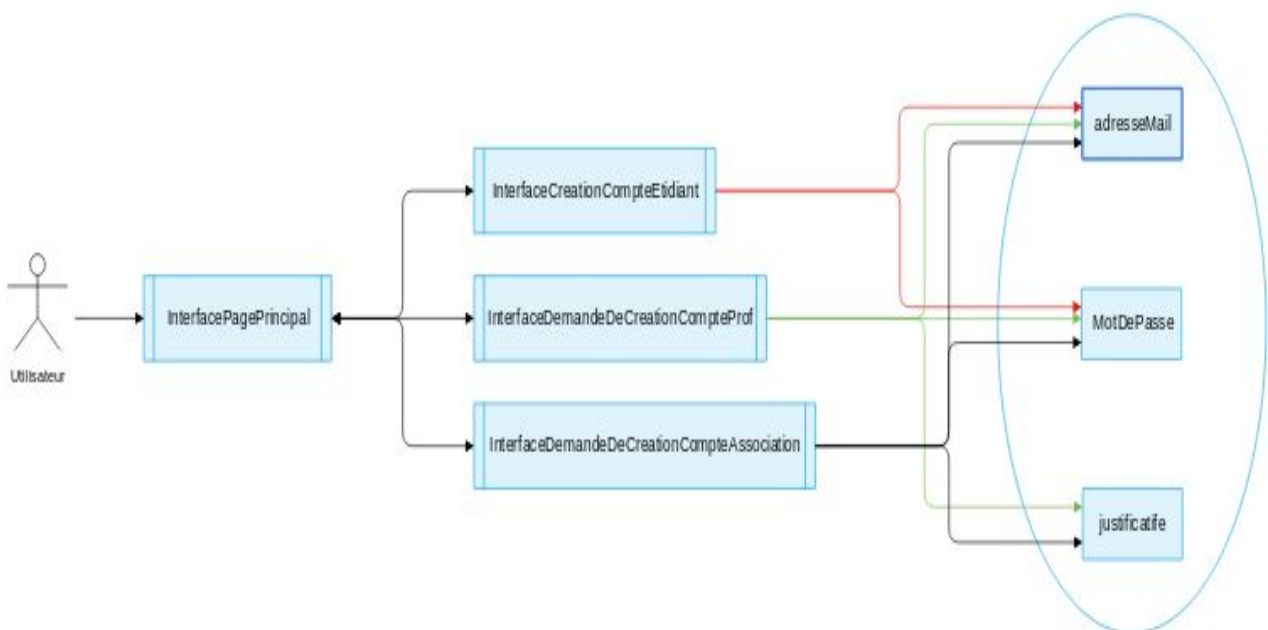
2- Diagrammes de cas d'utilisation :

a. Diagramme cas d'utilisation : authentification:



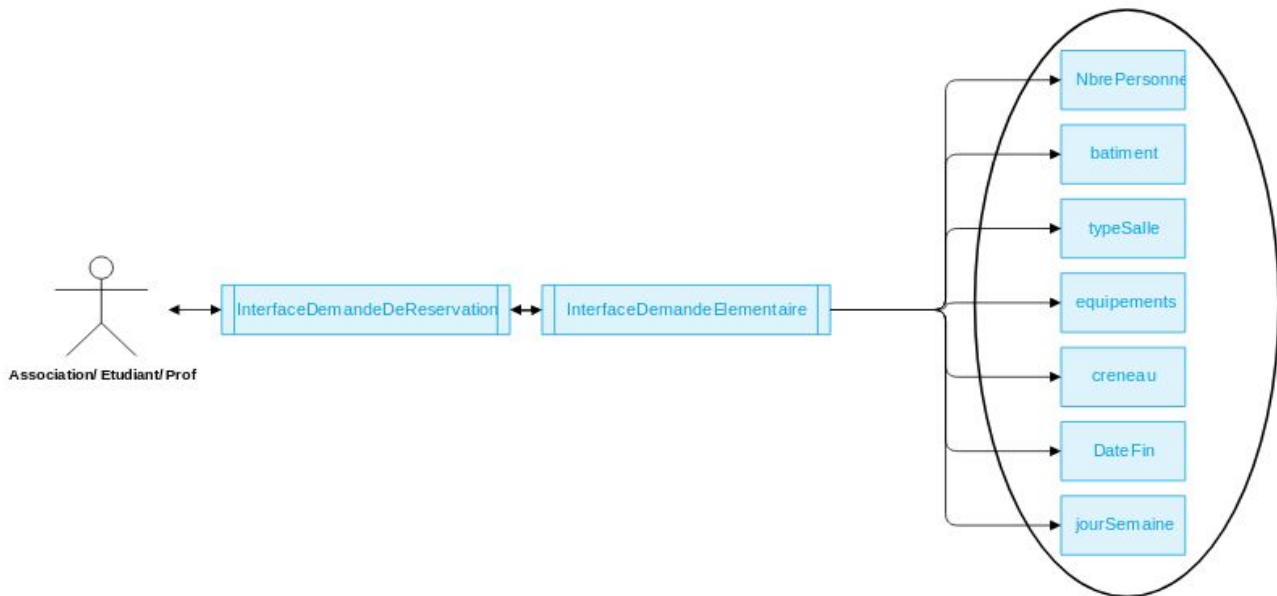
- Lors de la connexion (authentification), les utilisateurs étudiant, professeur et association, seront dirigés vers une interface différente de celle d'un gestionnaire qui représentent respectivement les comptes étudiant, professeur, association et le compte gestionnaires.

b. Diagramme cas d'utilisation :Création de compte :



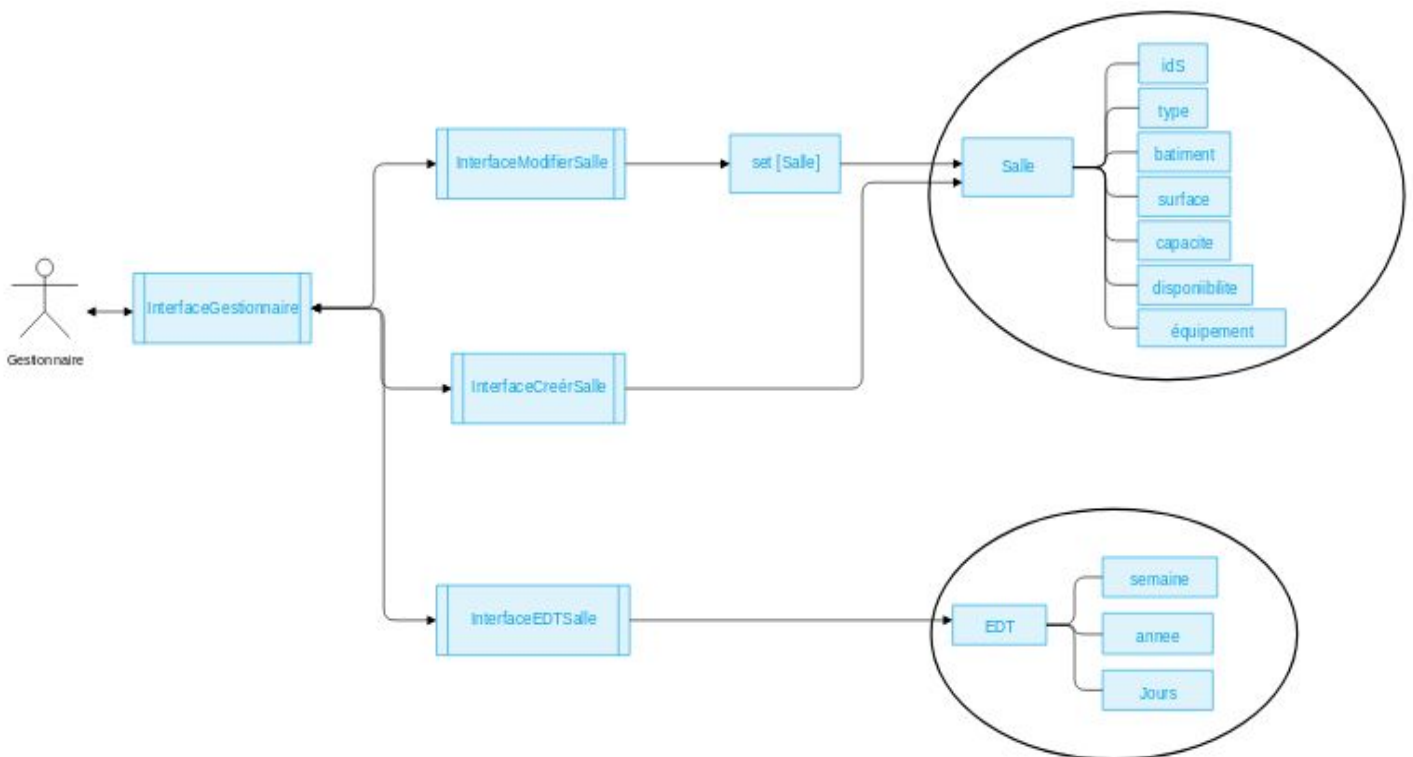
- Lors de l'inscription le professeur et association seront envoyées vers une interface d'inscription différente de celle de l'étudiant ou il doivent fournir un justificatif pour leurs inscription.

c. Diagramme cas d'utilisation :Demande de réservation:



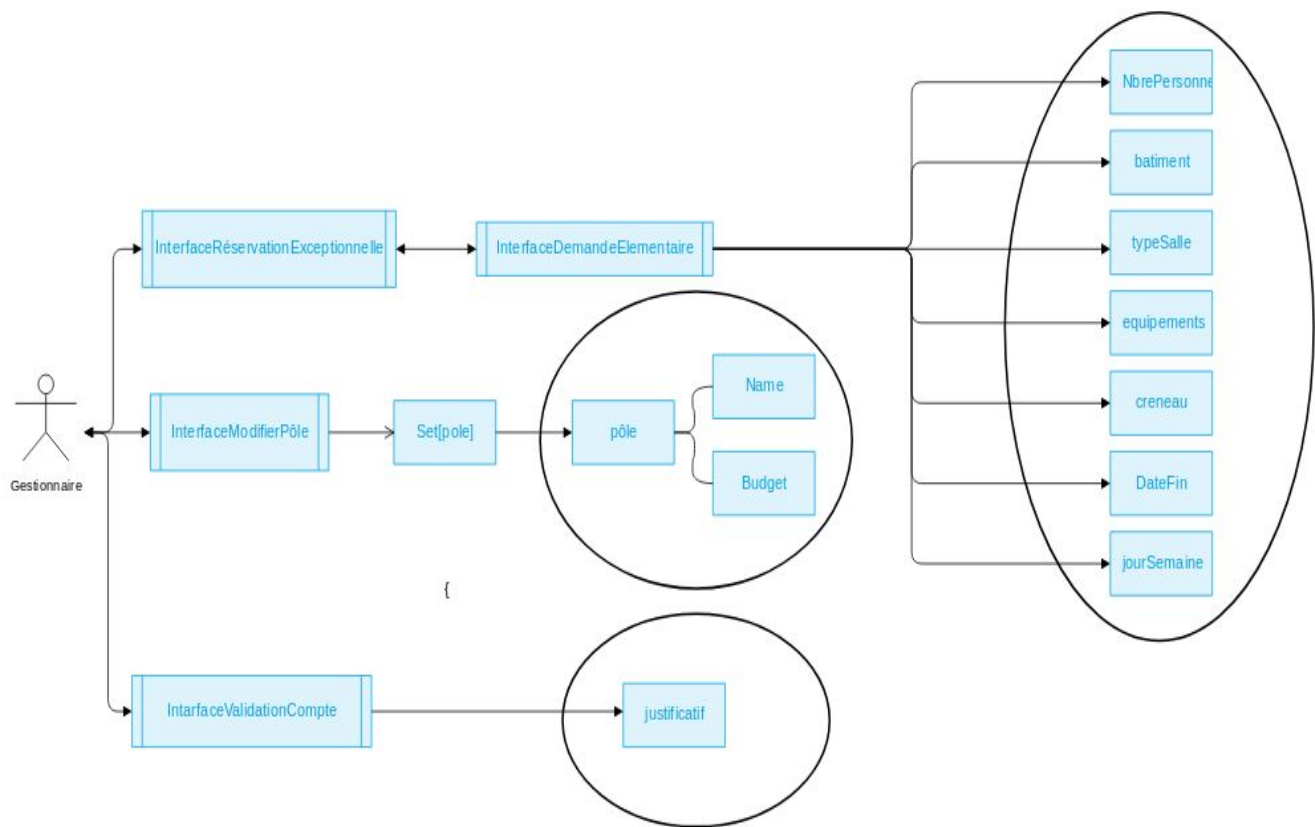
- Le professeur et l'étudiant peuvent directement faire une demande de réservation alors que l'association sa demande doit être vérifiée par le gestionnaire sauf lorsqu'il s'agit d'une salle de travail.
- Un étudiant peut effectuer une demande de réservation uniquement pour une salle de travail.

d. Diagramme cas d'utilisation :Gestion Salle:



- La gestion de salle est faite par le gestionnaire et consiste    pouvoir
 - Modifier une salle (modifier ses sp  cificit  s, ses   quipements, le nombre de personnes max, son type etc).
 - Changer l'EDT (emploi du temps) d'une salle ou cr  er une nouvelle salle (en sp  cifiant les   quipements, le nombres de personnes max, etc).
 - Rendre une salle disponible ou indisponible.

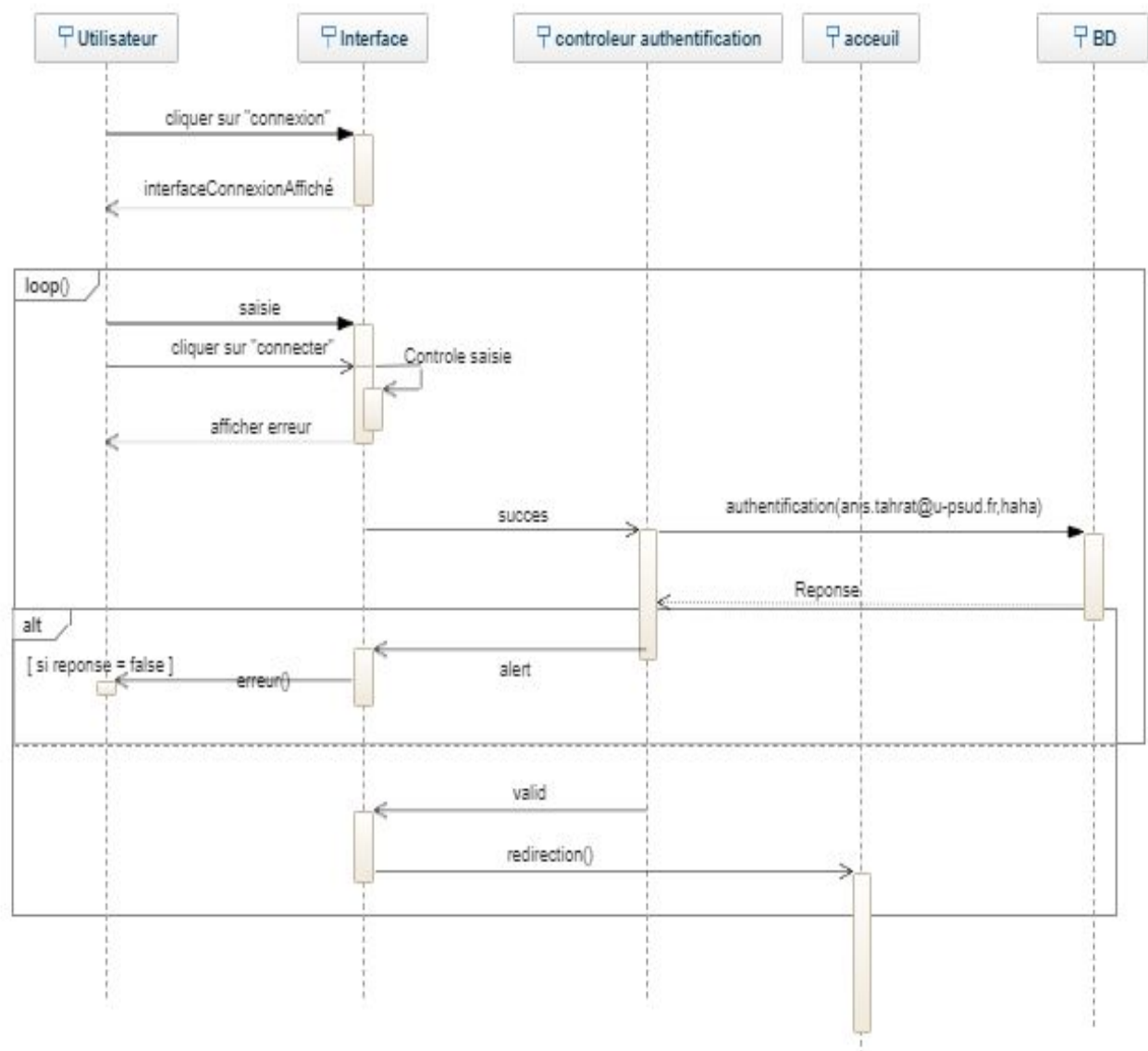
e. Diagramme cas d'utilisation : Gestion Système:



- Seul le gestionnaire peut accéder à la gestion du système qui représente tout ce qui concerne les opérations de :
 - Réservation de salle exceptionnelle : en cas d'événement exceptionnel empêchant l'utilisation de la salle (comme le dysfonctionnement du matériel fournis) cela nécessite une intervention manuelle du gestionnaire afin de réserver une nouvelle salle.
 - Modification de pôle (modifier le côta horaire concernant un pôle donné),.
 - Ainsi que les requête concernant les comptes des utilisateurs comme la création.

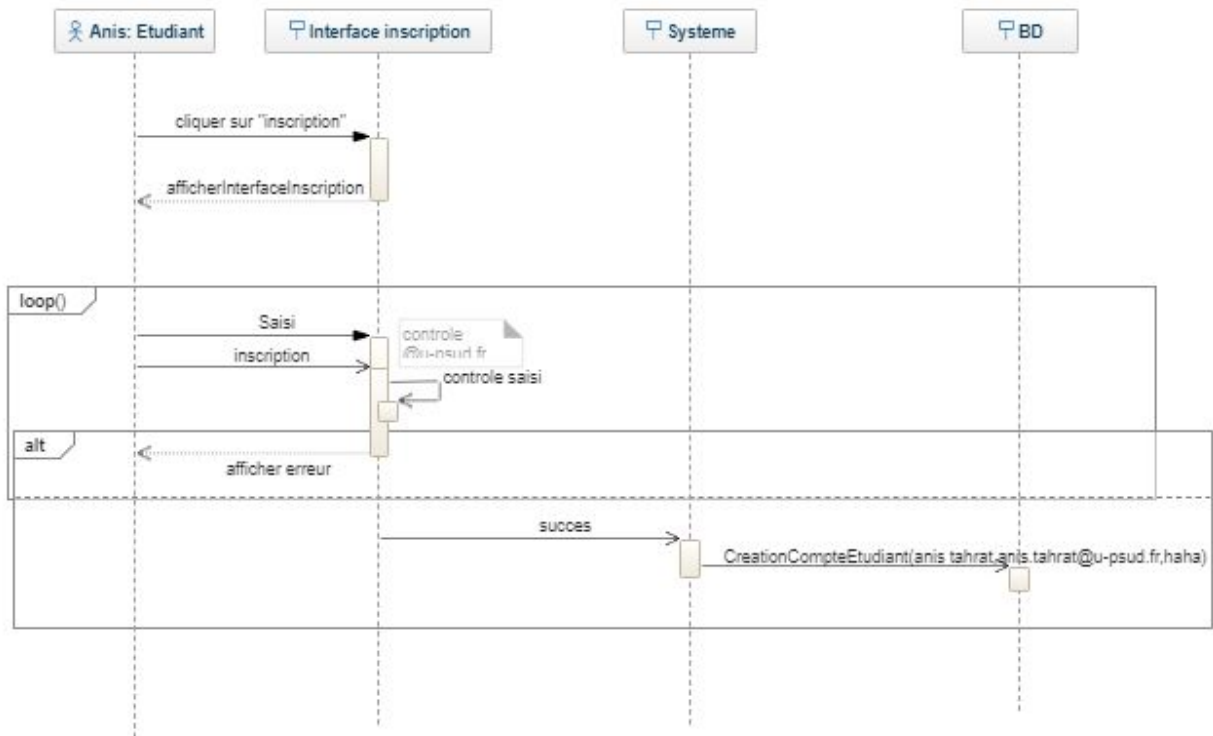
3- Diagrammes de séquences :

- **sequence-diagram : Utilisateur qui se connecte sur notre site web**



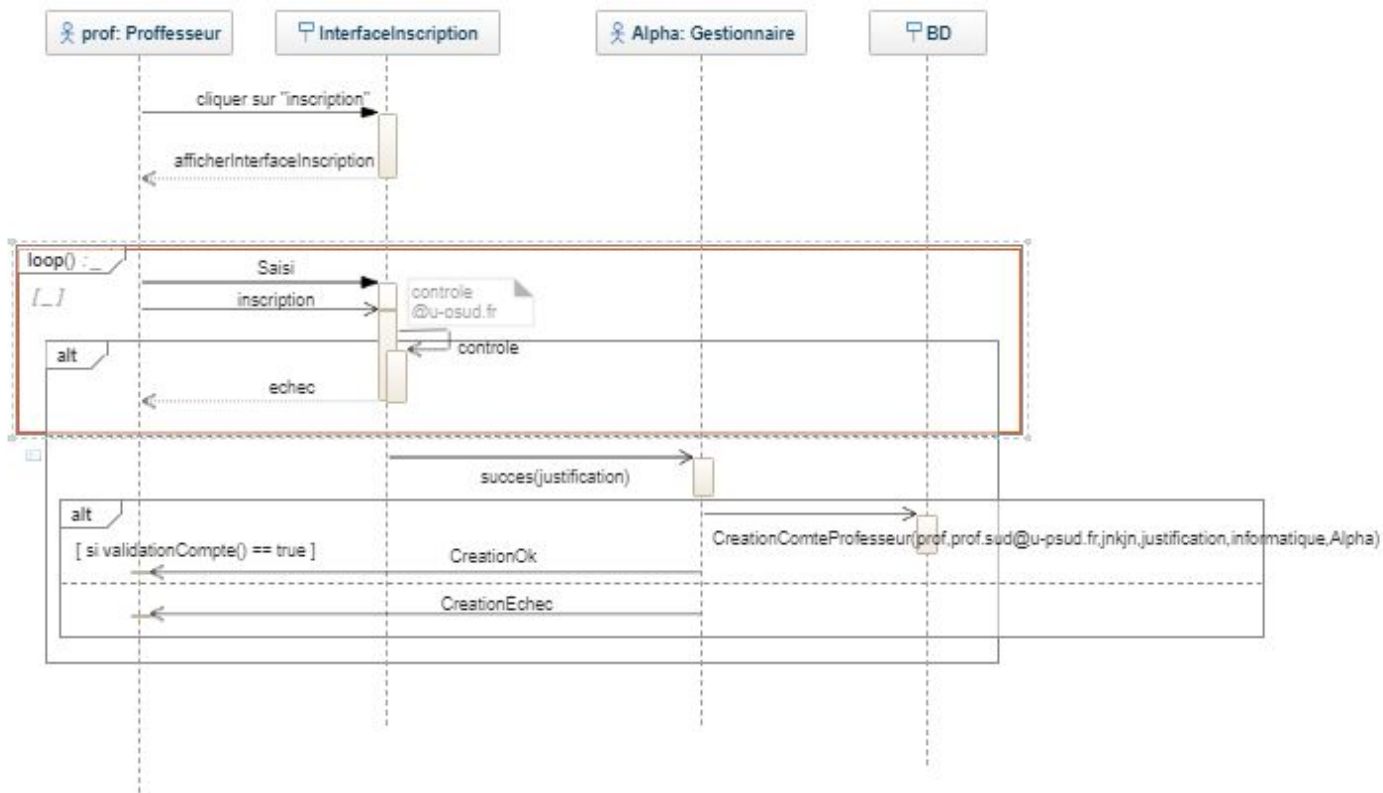
Tant que l'étudiant n'a pas remplis les informations correctement ou bien le compte n'est pas présent dans la base de donnée, il se verra refuser l'authentification et sera renvoyé sur l'interfaceConnexion avec un message d'erreur, sinon, il sera directement redirigé en page d'accueil et pourra réserver des salles.

- **sequence-diagram : Étudiant qui s'enregistre sur notre site web et qui devient membre**



Tant que l'étudiant n'a pas remplis les informations correctement, il se verra refuser l'inscription et sera renvoyé sur l'interfaceInscription, sinon, le systeme se chargera de la création du compte via la fonction CreationCompteEtudiant(...)

- **sequence-diagram : Professeur qui s'enregistre sur notre site web devenant membre**

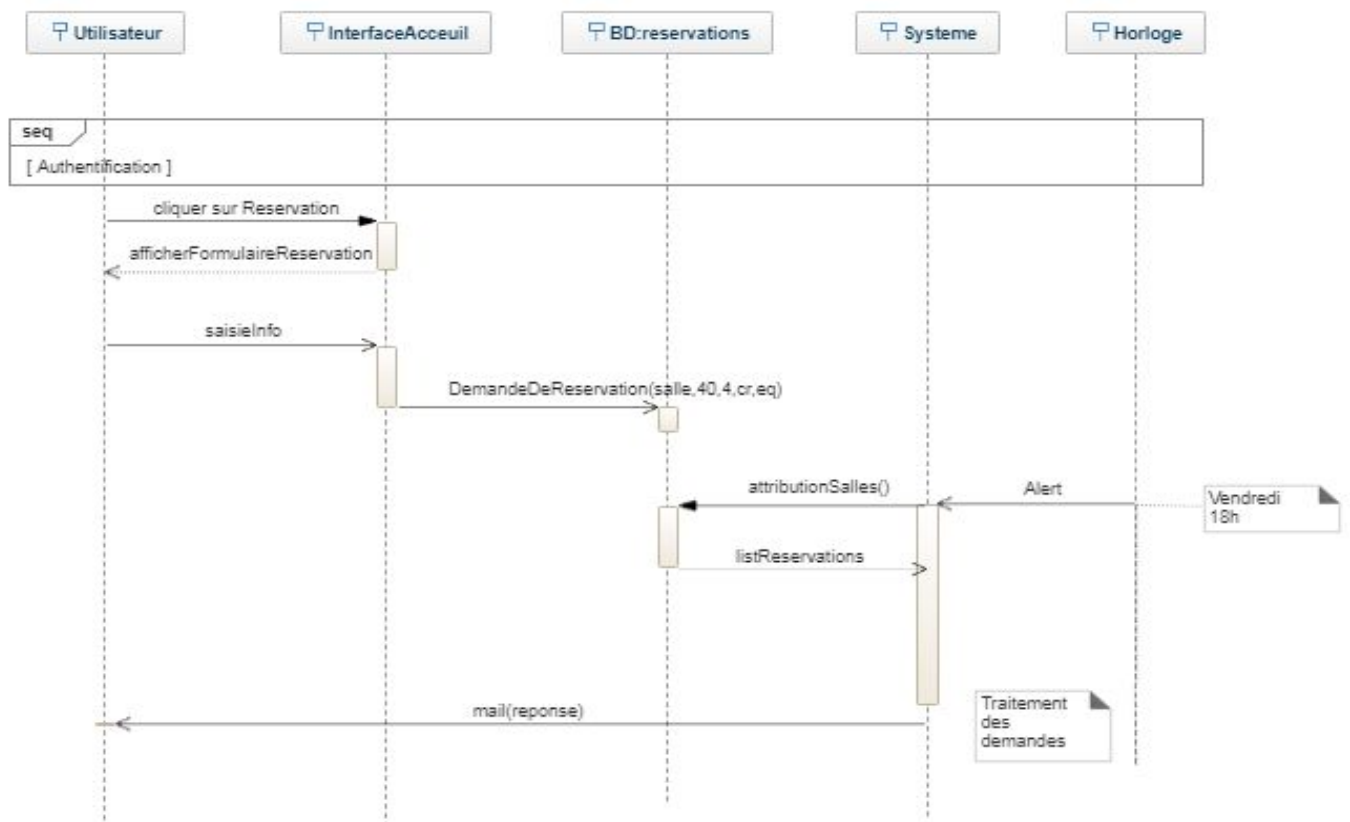


Tant que le professeur n'a pas remplis les informations correctement, il se verra refuser l'inscription et sera renvoyé sur l'interfaceInscription, sinon, sa justification sera envoyé à un gestionnaire qui se chargera de vérifier la crédibilité du professeur. Si sa décision est favorable, le système se chargera de la création du compte via la fonction `CreationCompteProfesseur(...)` sinon son inscription est refusé. Dans les deux cas l'utilisateur sera informé par mail.

- **sequence-diagram : Creation d'un Compte Association au niveau du serveur**



- **sequence-diagram : d'un utilisateur qui formule une demande de reservation**



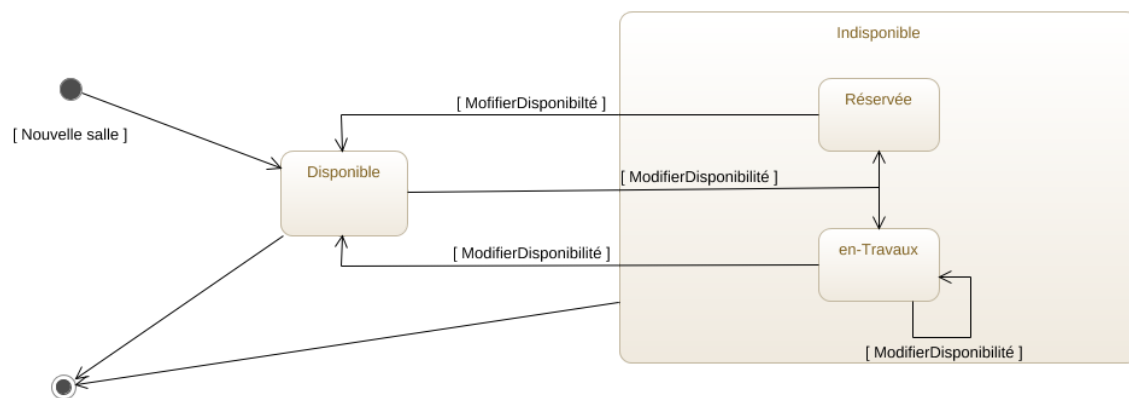
Ce diagramme regroupe les deux scénarios où l'utilisateur formule une demande de réservation et le système traite toutes ses dernières et renvoi la réponse.

L'utilisateur se connecte en premier lieu. Souhaitant réserver une salle, il ouvre le formulaire de demande et saisi les informations nécessaires à l'opération. Enfin, l'utilisateur envoie le formulaire qui permet de créer la réservation et la stocker dans la base de données.

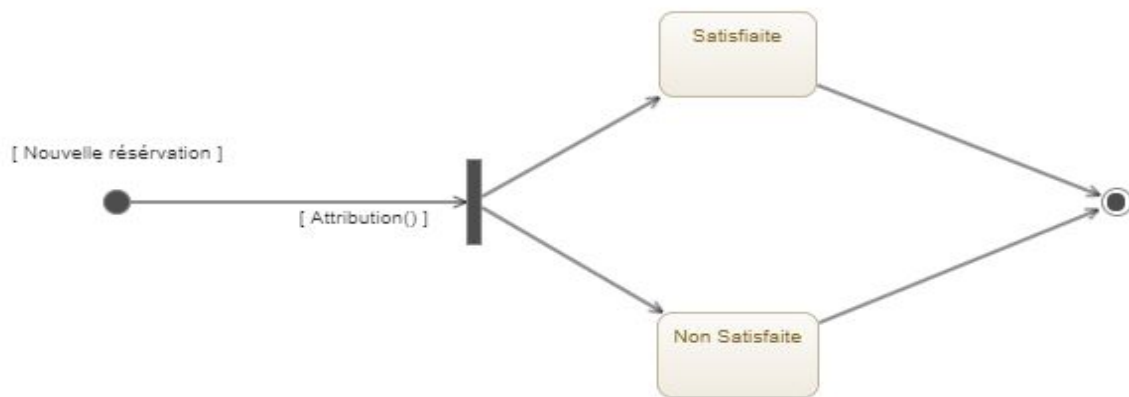
Ceci étant fait, chaque vendredi 18h, le système traitera toutes les demandes de réservation en essayant de couvrir le plus possible les salles.

4- Diagrammes d'états :

- Le diagramme d'état concernant les deux états possibles d'une salle :
 - disponible : salle présente dans notre base.
 - indisponible : salle réservée ou endommagée.



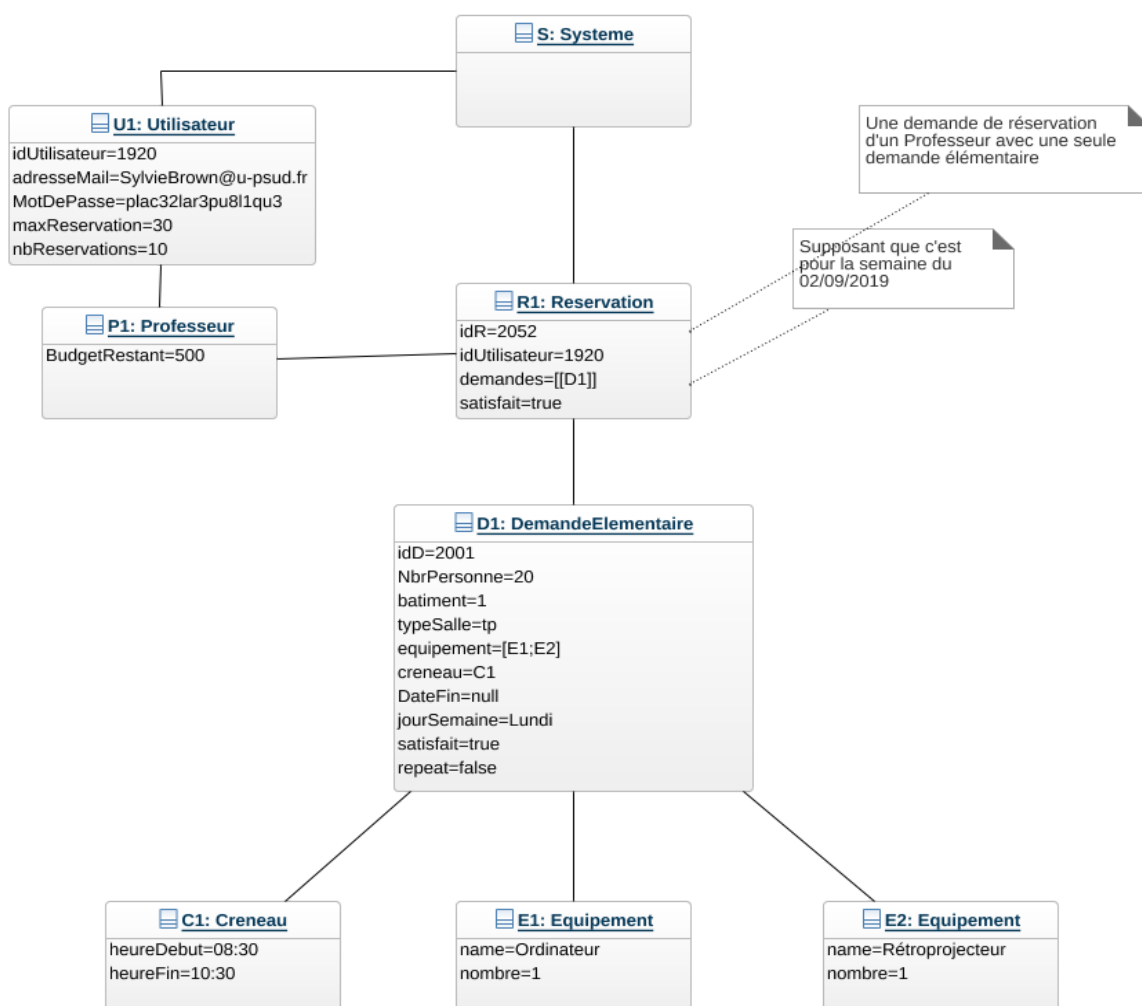
- Le diagramme d'état concernant les deux états possibles d'une réservation :
 - Satisfaite : La réservation a été satisfaite par notre algorithme, elle reste dans notre base de donnée avant le prochain traitement
 - Non Satisfaite : La réservation est supprimée



5- Diagrammes d'objets :

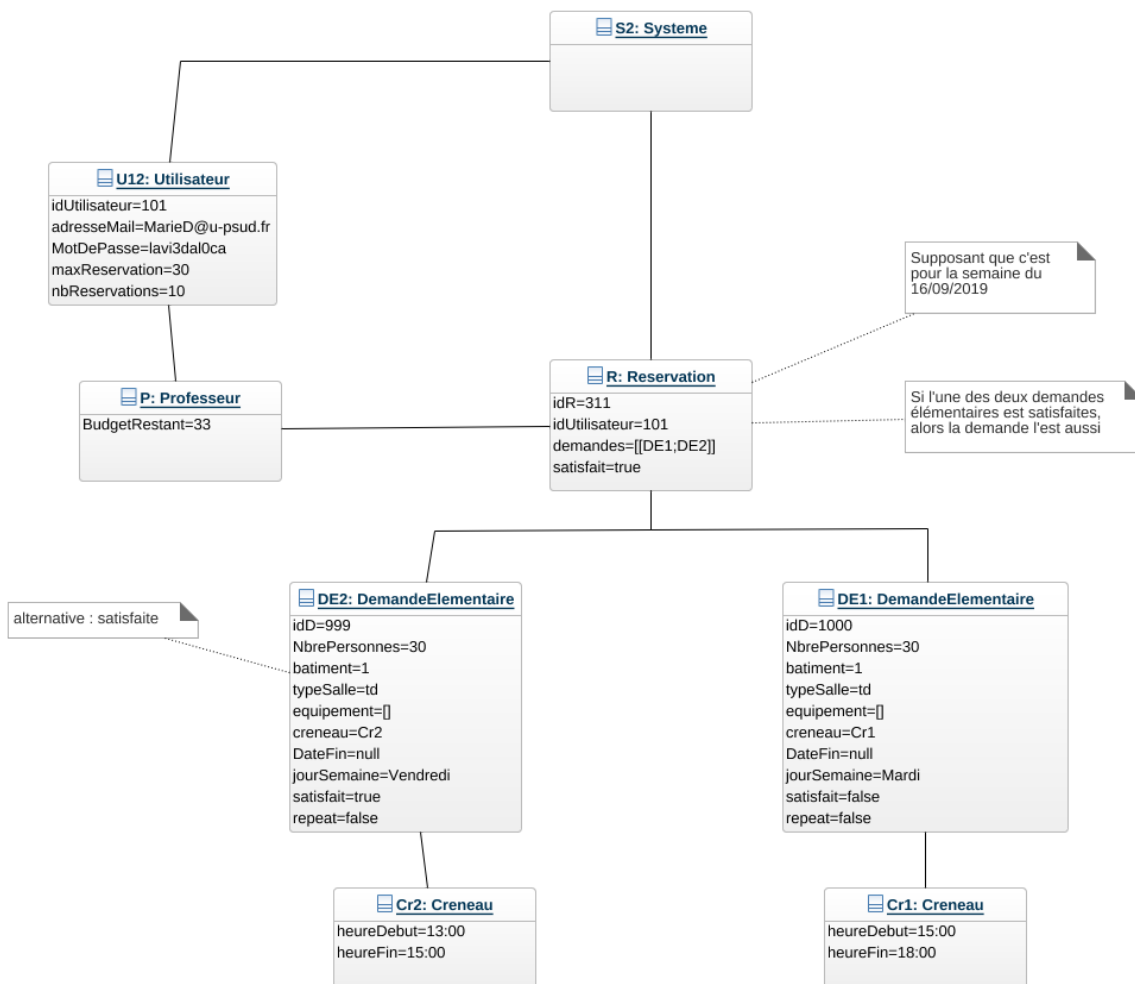
- Professeur :

- Une demande de réservation élémentaire:



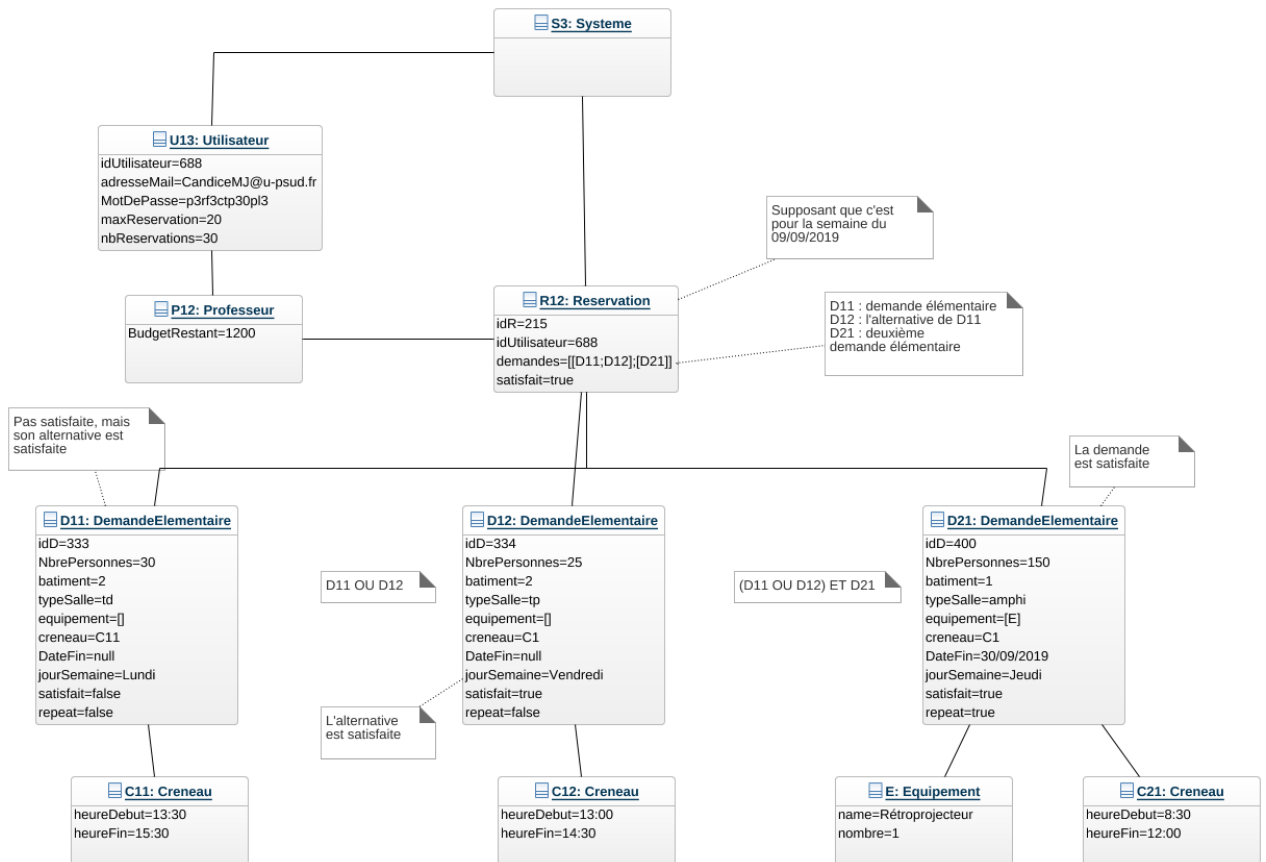
Le professeur P1 fait une demande de réservation R1 qui contient une seule demande élémentaire D1 pour le créneau horaire C1. P1 dans sa demande, souhaite avoir dans sa salle deux types d'équipements : E1 et E2. Sa demande a été traité par le système. Et a bien été satisfaite.

- Une demande de réservation élémentaire avec une alternative :



Le professeur P fait une demande de réservation R qui contient une seule demande élémentaire DE1 pour laquelle il souhaite avoir la demande alternative DE2. La demande DE1 pour le créneau horaire Cr1 et son alternative pour le créneau horaire Cr2. P dans sa demande, souhaite avoir une salle td pouvant contenir 30 personnes pour le mardi **OU** une salle td pouvant contenir 30 personnes le Vendredi. Après traitement de la demande R, le système n'a pas satisfait la demande DE1, mais il a pu satisfaire son alternative DE2. Donc la demande R du professeur P a bien été satisfaite.

- Deux demandes de réservation élémentaires dont l'une avec une alternative :



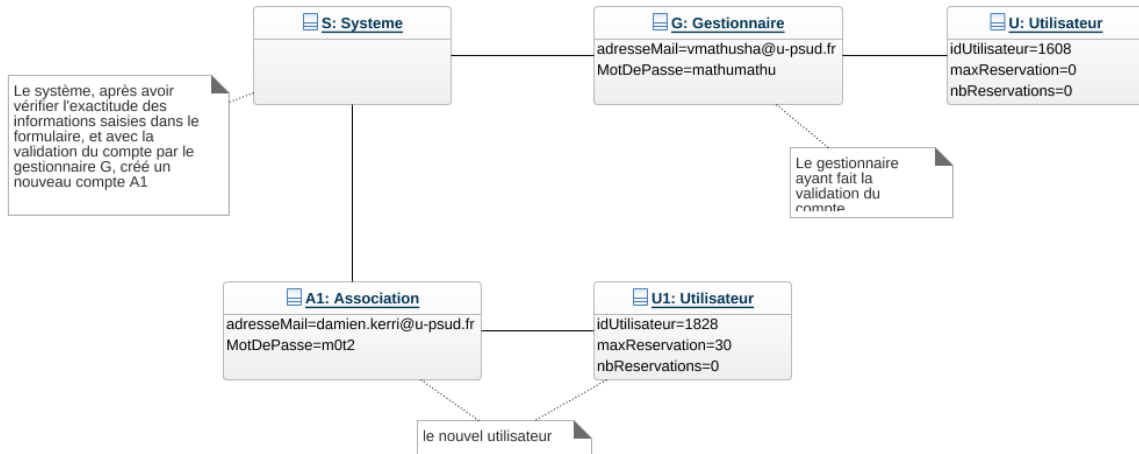
Le professeur P12 fait une demande de réservation R12 qui contient une demande élémentaire D11 et une autre demande élémentaire D21.

P dans sa demande de réservation souhaite trouver une salle td de 30 personnes le Lundi **OU** sinon une salle tp le Vendredi de 25 personnes, **ET** en plus de ça un amphi de 150 personnes pour tous les Jeudis jusqu'au 30/09/2019.

Après traitement de la demande R12, le système a satisfait la demande alternative de D11 qui est D12, et la demande D21. Donc la demande R du professeur P a bien été satisfaite.

- Association

- Création d'un nouveau compte association

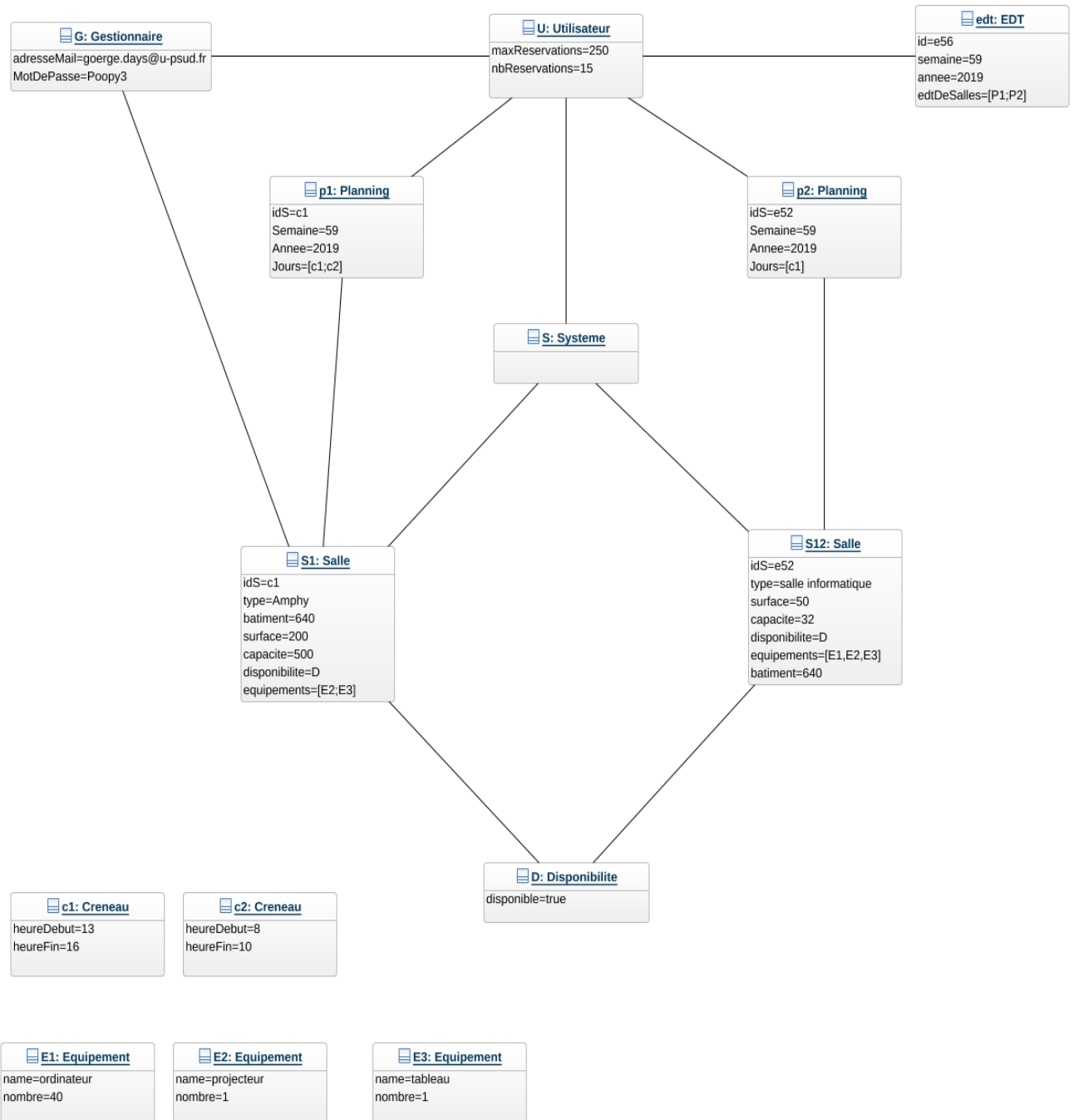


Un visiteur a demandé la création d'un compte gestionnaire, le système reçoit la demande, sous forme de formulaire dont il teste l'exactitude des informations rentrées et envoie la pièce (image) de justification, que le visiteur aurait joint afin d'expliquer sa demande d'ouverture d'un compte association (le but de cette association, les membres, etc), au gestionnaire G pour qu'il valide ou pas la création du compte.

G valide la création du compte, et le système crée donc un compte association A1 qu'il ajoute à l'ensemble des utilisateurs se trouvant dans la base de données.

- Gestionnaire

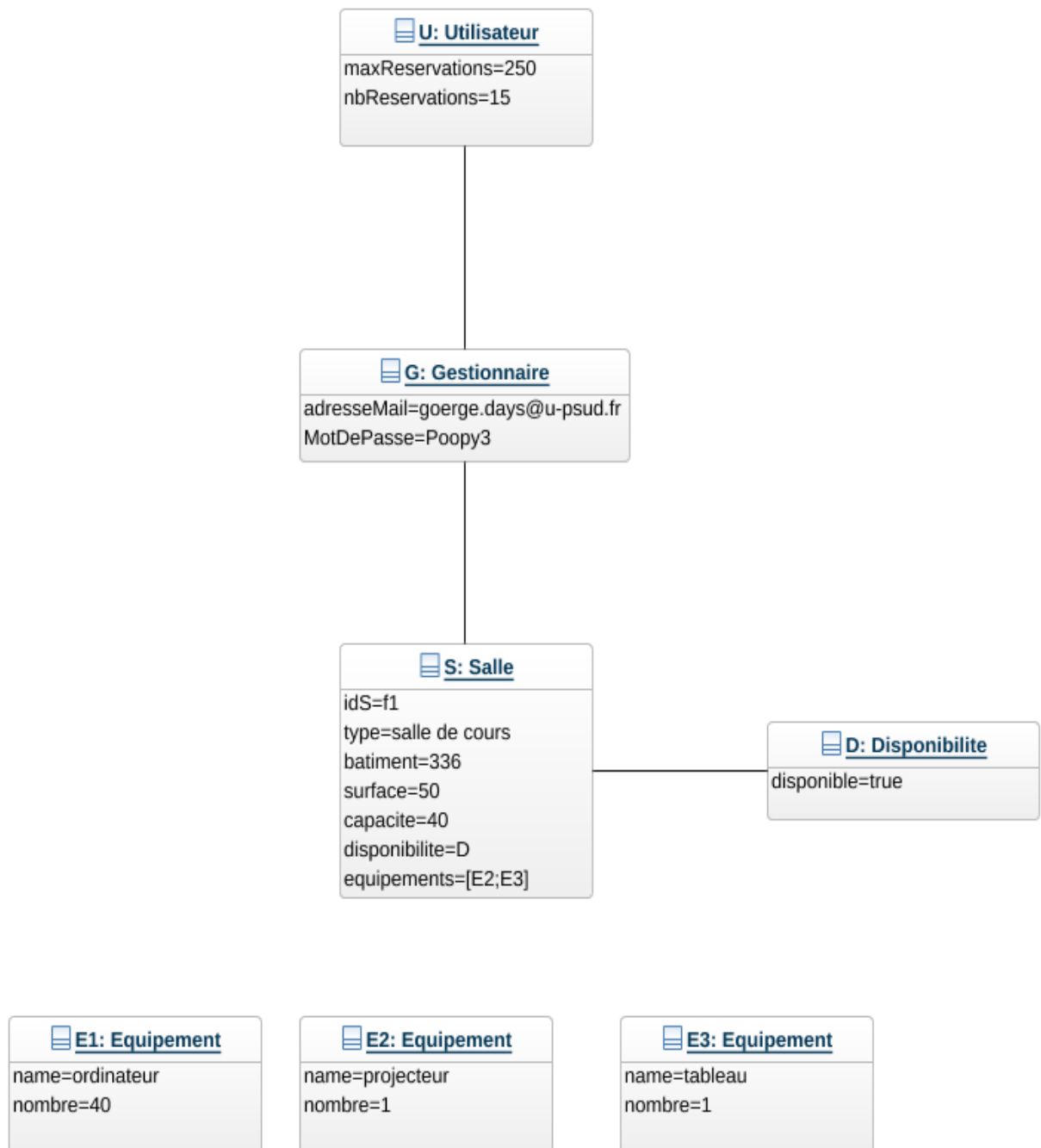
- Demande de réservation exceptionnelle par le gestionnaire:



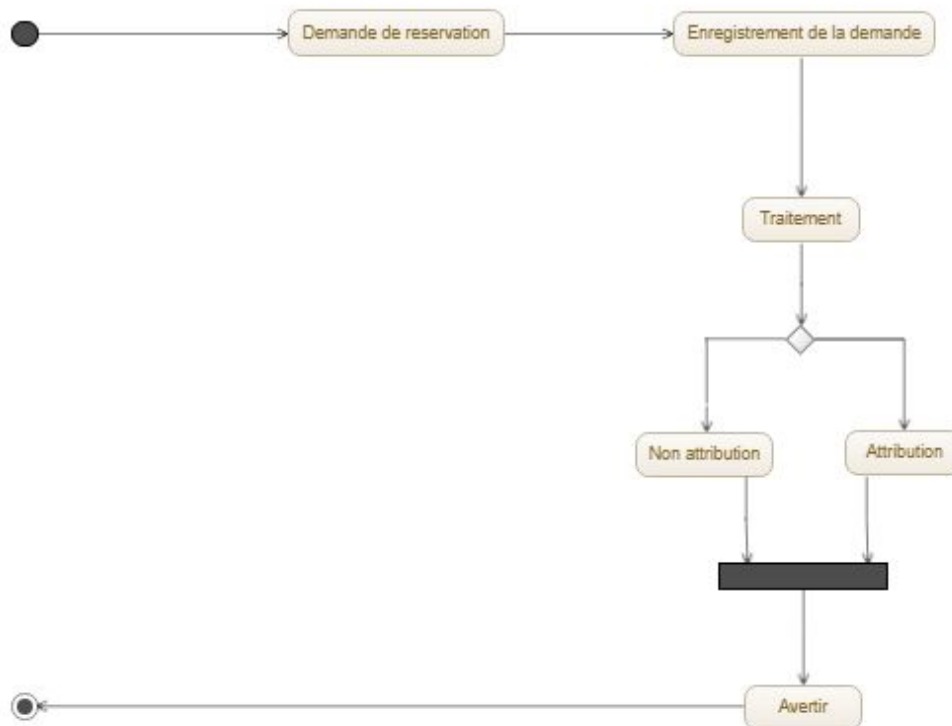
A l'aide de la fonction `DemandeDeReservationExceptionnelle` le gestionnaire peut réserver une salle, exceptionnellement afin de l'attribuer à un utilisateur ayant réservé une salle qui est devenue indisponible à cause d'un événement quelconque (endommagement des équipements, travaux, etc)

La nouvelle salle réservée est donc attribuée à l'utilisateur en question et est ajoutée à l'emploi du temps .

- Création de nouvelle salle par le gestionnaire:



6- Diagramme d'activité: déroulement d'une demande :



III. La conception en détail

1. Problèmes + patterns de conception:

1.1. Pseudo-code :

- Fonction qui teste l'email de l'utilisateur :

```
-bool check_user (mail : String) {  
    mailTab = mail.split('@')  
    if (mailTab[1] = "u-psud.fr") then true  
    else false  
}
```
- Fonction qui crée un compte étudiant :

```
-Etudiant CreationCompteEtudiant(mail: string, mdr: string ){  
    Etudiant newE = new Etudiant( idUt+1, mail, mdr, maxRes, 0)  
    add(newE, baseDeDonnee)  
}
```

- Fonction qui crée un compte gestionnaire :
 -Gestionnaire CreationCompteGestionnaire(mail: string, mdr: string, g: Gestionnaire){
 Gestionnaire newG = new Gestionnaire(idU+1, mail, mdr, 0, 0)
 if (g.ValidationCompte(newG)
 then add(newG, baseDeDonnee)
 }
- Fonction qui attribue les salles aux réservations (le scheduling)
 pour rappel: la liste de liste des demandes élémentaires d'une même réservation est représentée comme suit :
 - la liste principale contient des listes telles que : 1ère liste de demandes ET 2ème liste de demandes 2 ET ... ET N'ième liste de demandes doivent être satisfaites pour que la réservation soit satisfaite.
 - toute liste contenue dans la liste principale contient des demandes élémentaires telles que : la demande1 OU la demande2 OU ... OU la demandeN doit être satisfaite pour que la liste soit satisfaite.

```
-void attributionSalles( ) {
    p = Planning
    listR = 'Select * from Reservation ' //récupérer toutes les réservations de tous les
    utilisateurs dans une liste, où chaque ligne contient un objet de la classe
    Reservation

    foreach( Reservation r : listR ) { //parcourir les réservations
        foreach( <list<list<DemandeElementaire>> d : r.demandes){ //d étant la liste
        principale d'une même réservation
            try {
                satisfaireReservation(d,p) //essayer de placer d dans p
            }catch (not possible){
                "la réservation r.idR n'est pas satisfaite"
            } //s'il n'est pas possible de placer d
        }
    }
}
```

- Fonction qui vérifie si la réservation d'un utilisateur est satisfaite
 void satisfaireReservation(d : <list<list<DemandeElementaire>>, p : Planning){
 <list<DemandeElementaire>> listD; // une demande elementaire avec ses
 alternatives
 listD = choose(d) //choisir une liste
 }


```

if( satisfaireListe(listD, p) ) {
    delete listD from d;
    satisfaireReservation(d) //se rappeler récursivement sur le reste de la liste
}else{
    throw not possible //une liste n'est pas satisfaite alors la réservation n'est
pas satisfaite ( .
}
}

```

- Fonction qui cherche à satisfaire une liste de demandes qui contient une demande élémentaire et ses alternatives)

```

bool satisfaireListe ( <list<DemnadeElemntaire>> listD, p){
    count = 0;
    while (notAllVisited (listD) ){
        x = choose(d,listD) //choisit une demande de la liste
        if (place x into p) {
            x.satisfait = true;
            return true;
        }else{
            retrake x from p
            satisfaireListe(listD) //rappeller récursivement sur le reste de la liste
        }
    }
}

```

- Fonction qui permet d'annuler une réservation :

```

-void resAnnulee(r: Reservation) {
    delete (r, baseDeDonnees)
}

```

- Fonction qui vérifie les informations entrées dans le formulaire de demande de réservation :

```

-bool checkFormulaireReservation() {
    if(!(tous les champs sont remplies) ) then return false
    else
        if(hFin - hDebut < 30 ) then return false //30 min au minimum
        if(repete) then //repete a partir de la semaine prochaine
            if(dateFin < dateActuelle + 7) then return false
            if(alternative ) then //on fait de meme pour l'alternative
                if (!alternative.checkFormulaireReservation()) then
                    return false
        return true
}

```

- Fonction de connexion (authentification) d'un membre :

```
-void authentication() {
    var email= req.email
    var password = req.password; connection.query('SELECT * FROM Utilisateur
    WHERE email = ?', [email], [resultat]
    Verifier(resultat)
}
```

```
Verifier(res){
    if (error) {
        "error occurred"
    }else {
        "la solution est results"
        if(results.length >0){
            if(results[0].password == password){
                "success"
            }else{
                "Email et mot de passe incompatibles"
            }
        }else{
            "l'email n'existe pas"
        }
    }
}
```

- Fonction qui permet de changer son mot de passe

```
void ModifierMDP (old : String, new : String){
    replace(old, new);
}
```

- Fonction qui permet à un utilisateur d'afficher l'ensemble de ses réservations utilisateur

```
void check_EDT(){
    show EDT() where idUtilisateur = this.idUtilisateur;
}
```

1.2. Bibliothèques :

Node.js	Initialiser des objets correspondant aux librairies avec la fonction require, par exemple, la librairie http qui gère le protocole HTTP, définir un serveur, lancer le serveur, etc.
Moment.js	Pour la gestion des dates (ex : additionner une date avec une période)
Decimal.js	Pour tout ce qui est du traitement des nombres : conversion, transposition, interprétation, gestion des virgules.
Underscore.js	Pour la manipulation des données(tableau, itérateurs, map, etc.)
jQuery	Simplifie de nombreuses tâches en réduisant la quantité de code que nous avons besoin d'écrire.
server.js (qui fait partie de Express)	pour l'authentification et la manipulation de la base MySQL.

2. Attributs d'objets et sémantique:

- Visibilité des attributs d'une classe :

Tout attribut est privé dans sa classe et ne peut être utilisé par une deuxième classe que si cette dernière a accès à un objet du type de la première classe.

2.1. Utilisateur:

Attribut	Type	Définition	Représentation graphique
idUtilisateur	integer, ça sera une variable globale dans cette classe.	L'identifiant unique de l'utilisateur.	/
adresseMail	string	L'adresse mail de l'utilisateur.	Un champ de texte à remplir .
MotDePasse	string	le mot de passe de l'utilisateur.	Un champ de texte à remplir .

maxReservation	Integer	Le nombre maximum de demandes de réservation qu'un utilisateur peut faire. Qui aura la valeur 0 par défaut pour les gestionnaires.	Il vous reste X demandes possibles.
nbReservations	integer	le nombre de reservation effectuée par l'utilisateur. Qui aura la valeur 0 par défaut pour les gestionnaires.	Vous avez effectué X demandes de reservations.

Fonction	Rôle	Entrée	Sortie
public void ModifierMDP P (mdp : String, mdp1 : String)	Permet à l'utilisateur de modifier son mot de passe.	-mdp : L'ancien mot de passe. -mdp1 : Le nouveau mot de passe.	/
public void AnnuleReservation (id : integer)	Permet d'annuler une demande de réservation.	id : l'identifiant de la réservation .	/
public void deconnexion ()	Pour fermer la session .	/	/
public void check_EDT ()	Permet à l'utilisateur de consulter son emplois du temps.	/	/

public int getMaxRes	Retourne le nombre maximum de demandes de réservations que l'utilisateur peut faire pendant toute la semaine	/	Integer
---------------------------------	--	---	---------

- L'utilisateur (étudiant, professeur, association) peut accéder qu'à son emplois du temps . Il ne peut pas voir qui a réservé une salle.
- Quand un utilisateur(professeur, étudiant, association) veut modifier son mot de passe, il aura sur le site un bouton , quand il cliquera dessus il va voir afficher deux champs de texte :
 - 1) le premier à remplir avec l'ancien mot de passe.
 - 2) le deuxième a remplir avec le nouveau mot de passe.
 - 3) Puis il cliquera sur le bouton valider pour valider ses modifications.
- Un utilisateur (professeur, étudiant, association) peut annuler une réservation avant que les plannings soient faits en utilisant le bouton annuler réservation où un champ de texte s'affiche et il pourra mettre l'id de la réservation qu'il veut annuler.
- Tout utilisateur peut consulter son planning grâce à un bouton mon Planning. Il sera redirigé vers une page ou le planning sera affiché.

2.2. Gestionnaire:

Fonction	Rôle	Entrée	Sortie
private Salle CreationSalle(id : Integer, Salle : String, bat : Integer, s : Integer, equip : <list>Equipement , C : Integer, dispo : Disponibilite)	Création d'une nouvelle salle .	- Integer :L'identifiant unique de la salle. - String :le type de salle. - Integer :le bâtiment ou elle se situe. - Integer :sa surface. - <list>Equipement : équipement disponible dans la salle (exp: ordinateur, projecteur,...)	Salle

		<p>-Integer:sa capacité (le nombre maximum de personne qu'elle peut accueillir)</p> <p>-Disponibilite:sa disponibilité.</p>	
<p>private void DemandeDeReservationExceptionnelle(S : String, nbPers : integer, Bat : integer, j : Semaine, hpreaire : Creneau)</p>	<p>Réservation d'une salle exceptionnellement.</p>	<p>-String:Type de la salle a réserver.</p> <p>-integer:nombre minimum de personne qu'elle peut accueillir.</p> <p>-integer:le numéro du bâtiment ou on la veut .</p> <p>-Semaine:Le jour de la réservation.</p> <p>-Creneau:heure de début et heure de fin de la réservation.</p> <p>-<list>Equipement: une liste d'équipements dont l'utilisateur a besoin (exp: ordinateur, projecteur,...)</p>	/
<p>private void modifierBudgetPole(P : Professeur)</p>	<p>Décrémenter le nombre d'heure réservé du budget pôle du professeur.</p>	<p>un professeur.</p>	/
<p>private void check_EdtGestionnaire(S : Salle)</p>	<p>Consulter l'emploi du temps d'une salle .</p>	<p>La salle a laquelle on veut voir son emploi du temps.</p>	/

public Boolean ValidationCompte()	Valider la création de compte (gestionnaire, professeur, association).	/	Boolean
private void modifier_salle(S : Salle)	Modifier les différentes caractéristiques d'une salle.	- La salle à modifier.	/

- Le gestionnaire peut accéder à tous les emplois du temps des salles et voir tous ceux qui les ont réservé .

2.3. Association:

Possède tout ce que la classe utilisateur a comme attributs et méthodes et la procédure suivante :

Fonction	Rôle	Entrée	Sortie
private void demandeDeReservation(String,integer,integer,semaine,Creneau,<list>Equipement)	Pour faire une demande de réservation de salle .	-String: type de la salle (amphitheatre, salle de cours , selle de td...). -integer: le nombre minimum de place dans la salle. -integer: le numéro du bâtiment ou l'utilisateur veut sa salle. -semaine: le jour de la reservation dans la semaine. -Creneau: heure de début et heure	/

		de fin de la réservation. -<list>Equipemen t: une liste d'équipements dont l'utilisateur a besoin (exp: ordinateur, projecteur,...)	
--	--	---	--

2.4. Etudiant:

Possède tout ce que la classe utilisateur a comme attributs et méthodes et la procédure suivante:

Fonction	Rôle	Entrée	Sortie
private void demandeDeReservation(S : String, NbPers : integer, Bat : integer, J : semaine, Creneau, Equip : <list>Equipement)	Pour faire une demande de réservation de salle .	- String : type de la salle (amphitheatre, salle de cours , selle de td...). - integer : le nombre minimum de place dans la salle. - integer :le numéro du bâtiment ou l'utilisateur veut sa salle. - semaine : le jour de la reservation dans la semaine. - Creneau : heure de début et heure de fin de la réservation. -<list>Equipem	/

		ent: une liste d'équipements dont l'utilisateur a besoin (exp: ordinateur, projecteur,..)	
--	--	--	--

2.5. Professeur :

Cette classe possède tout ce que la classe utilisateur a comme attributs et méthodes par héritage.

Attribut	Type	Définition	Représentation graphique
BudgetRestant	integer	Le nombre d'heure restant du budget horaire réservé pour chaque professeur selon pôle.	une phrase écrite: Il vous reste x heures de votre budget horaires.

Fonction	Rôle	Entrée	Sortie
public void AppartietAPole()	Retourne le pôle du professeur.	/	Pole
private void demandeDeReservation(S : String, nbPers : integer, Bat : integer, J: semaine, horaire :	Pour faire une demande de réservation de salle.	-String: type de la salle (amphitheatre, salle de cours , selle de td...). -integer: le nombre minimum de place dans la	/

Creneau,<list>Equipement)		<p>salle.</p> <p>-integer:le numéro du bâtiment ou l'utilisateur veut sa salle.</p> <p>-semaine: le jour de la reservation dans la semaine.</p> <p>-Creneau: heure de début et heure de fin de la réservation.</p> <p>-<list>Equipement: une liste d'équipements dont l'utilisateur a besoin (exp: ordinateur, projecteur,..)</p>	
--	--	---	--

2.6. Pôle:

Attribut	Type	Définition	Représentation graphique
name	String	Nom du pôle exemple: informatique, mathématique, chimie...	/
Budget	integer	Total des horaires.	/

2.7. Réservation:

Attribut	Type	Définition	Représentation graphique
----------	------	------------	--------------------------

idR	Integer	L'identifiant unique de la réservation.	/
idUtilisateur	String	L'identifiant unique de l'utilisateur ayant fait la demande.	
demandes	LinkedList < LinkedList < DemandeElementaire > >	<p>La liste des demandes d'un utilisateur donné telles que :</p> <ul style="list-style-type: none"> • Une demande de reservation dans laquelle l'utilisateur a sélectionné plusieurs jours (lundi, mardi) forme plusieurs demandeElementaire (D11 pour lundi et D2 pour mardi), et sont donc dans deux listes différentes : [[D11] ;[D2]]. • Pour chaque jour choisi (ex D11), l'utilisateur se verra proposer une alternative (ex D12) qui sera ajoutée à la meme liste que la demande élémentaire pour ce jour là. [[D11;D12];[D2]]. 	/

Remarque :

Avec l'attribut demandes on a pu représenter les demandes élémentaires avec leurs alternatives. Pour les séquences, deux demandes D1 et D2 faites séparément par le même utilisateur forment une séquence, mais sont dans deux réservations différentes.

2.8. Demande Elémentaire:

Attribut	Type	Définition	Représentation graphique
idD	Integer	Entier identifiant une demande de manière unique	/
NbrePersonnes	Integer	Le nombre de personnes que doit pouvoir contenir la salle.	Champ texte à remplir.

batiment	Integer	Le numéro du bâtiment souhaité.	Liste déroulante avec les numéros des bâtiments.
typeSalle	String	Type de la salle souhaitée, peut-être une salle tp, td, cours, amphi, etc.	Liste avec tous les types de salles. Un seul peut être sélectionné.
equipements	LinkedList<Equipement >	Liste des équipements souhaités. (En plus des équipements de la salle demandée).	Liste des équipements (ordinateurs, tableaux etc). Choix multiple.
creneau	Creneau	La plage horaire de la demande. >= 30 minutes	2 champs à remplir : heure début, heure de fin.
DateFin	Date	Indique la date de fin de renouvellement de la demande de réservation d'un utilisateur. Peut-être nulle.	Espace à remplir si jamais l'utilisateur souhaite que sa demande soit renouveler automatiquement toutes les semaines jusqu'à cette dateFin.
JourSemaine	Boolean[5]	De type tableau de booléens de taille 5. Il contiendra 5 booléens correspondants aux jours de semaine du lundi au vendredi.	Un tableau de cases à cocher contenant les jours de semaine du lundi au vendredi que l'utilisateur devra cocher afin d'indiquer les jours où il souhaite faire sa demande de réservation.
satisfait	Boolean	Vrai si la demande a été satisfaite, faux sinon.	/

Fonction	Rôle	Entrée	Sortie
bool getEtat ()	Retourne vrai si la demande est satisfaite, faux sinon.	/	Boolean

Une demande de réservation se passe comme suit :

L'utilisateur Kenny, connecté, peut faire une réservation R1 en remplissant un formulaire :

- Entre le nombre de personnes.

- Choisit le bâtiment (dans une liste déroulante contenant les numéros des bâtiments).
- Choisit le type de salle (liste déroulante avec le nom de la salle : tp, td, amphithéâtre, etc, et ses équipements : x tables, y ordinateurs etc).
- Choisit les équipements supplémentaires à rajouter dans la salle (en cochant des cases : ordinateurs, rétroprojecteurs etc).
- Indique le créneau : heure de début et heure de fin de la réservation. (30 min au minimum).
- Coche le bouton d'itération s'il souhaite que sa demande soit renouvelée automatiquement toutes les semaines.
 - Si c'est le cas, il devra indiquer une date de fin d'itération.
- Choisit les jours de semaines pendant lesquels il souhaite faire sa réservation : une demande de réservation peut être faite pour un ou plusieurs jours d'une semaine X.
 - Ex : Kenny veut faire une demande R pour le Lundi, mardi et jeudi de la semaine du 02/09/2019, il coche alors les cases correspondantes à ces jours. Dans ce cas là, notre système considère que Kenny a fait trois demandes élémentaires R1 R2 et R3 pour chacun des trois jours.
- Il peut choisir de faire une demande de réservation alternative, si jamais sa demande élémentaire ne peut être satisfaite. Il remplira alors, tous les champs de la même manière que sa demande précédente.
 - si Kenny décide de faire une alternative, il doit choisir le jour pour lequel il souhaite faire cette alternative. Il pourra faire une ou plusieurs alternative soit pour la demande du Lundi, ou celle du Mardi ou bien celle du Jeudi.
- Soumet le formulaire.

La réservation est créée, avec état non satisfaite. Elle est ajoutée à la table de réservation. Elle a un idR XXXX, l'id utilisateur ainsi que la liste de ses demandes R1 R2 et R3 et Les alternatives s'il y en a.

Les demandes de réservations sont aussi initialement insatisfaites, elles sont ajoutées à la table de DemandeElementaire.

2.9. Equipement:

Attribut	Type	Définition	Représentation graphique
name	String	nom de l'équipement	un équipement est représenté

			dans une liste de tous les équipement par son nom
nombre	Integer	nombre d'équipements présent dans la salle	/

2.10. Salle:

Attribut	Type	Définition	Représentation graphique
Ids	Integer	identifiant de la salle	/
type	String	type de la salle : Salle de travail, Amphithéâtre, Salle de cours , Salle informatique...	Une liste déroulante de tous les types de salles
batiment	Integer	le bâtiment où se trouve la salle	Une liste déroulante de tous les numéros de batiment
surface	Integer	la surface de la salle	chiffre à préciser
capacite	Integer	le nombre de personne que la salle peut contenir	chiffre à préciser
disponibilite	Disponibilite	si c'est égal à true donc la salle est disponible sinon indisponible	/
equipements	LinkedList< Equipement >	la liste de tous les équipements d'une salle	/

Fonction	Rôle	Entrée	Sortie
void modifierDisponibilite(dispo : Disponibilite)	modifie l'attribue disponibilite en dispo	dispo : une Disponibilité	/

L'emploi du temps se fait pour la semaine prochaine seulement. :

- Toutes les demandes de réservation pour une semaine donnée sont faites la semaine d'avant.
- Le traitement des demandes aura lieu la fin de la semaine où les demandes ont été faites.

2.11. Disponibilité :

Attribut	Type	Définition	Représentation graphique
disponible	boolean	si c'est égal à true donc la salle est disponible sinon indisponible	/
raison	String	raison pour laquelle la salle n'est pas disponible	/

2.12. Créneau:

Attribut	Type	Définition	Représentation graphique
heureDebut	Integer	heure du début	/
heureFin	Integer	heure de fin	/

2.13. Planning :

Attribut	Type	Définition	Représentation graphique
idP	Integer	identificateur du planning d'une salle	/
semaine	Integer	Semaine du planning	/
annee	Integer	Année du planning	/
jours	LinkedList<Creneau> [5]	Un tableau de 5 listes de créneaux(l'indice 0 représente la liste des créneaux de lundi par	/

		exemple)	
--	--	----------	--

Le planning représente l'emploi du temps de chaque salle.

2.14. EDT:

Attribut	Type	Définition	Représentation graphique
idE	Integer	identifiant de l'edt	/
semaine	Integer	Semaine de l'EDT	/
annee	Integer	Annee de l'EDT	/
edtDeSalles	LinkedList<Planning>	Liste des planning de toutes les salles	/

- l'EDT représente l'emploi du temps de toutes les salles, il est accessible par tous les utilisateurs. Or chacun ne peut voir que ses propres réservations. Sauf le gestionnaire, qui lui, peut voir l'ensemble du EDT.

2.15. Systeme :

Fonction	Rôle	Entrée	Sortie
public void checkUser(String)	Contrôler l'email de l'utilisateur si il contient la motion "@u-psud.fr"	-String : Email de l'utilisateur	Boolean
public Etudiant CreationCompteEtudiant(String,String,String)	Création d'un nouveau compte étudiant	-String : Nom et Prenom -String : Email -String : Mot de passe	
public Professeur CreationCompteProfesseur(String,String,String,image,Pôle,Gestionnaire)	Création d'un nouveau compte professeur	-String : Nom et Prenom -String : Email -String : Mot de passe	

		-Image: Justificatif sous forme d'image -Pole: Le pôle auquel appartient le prof -Gestionnaire : le gestionnaire qui a permis la création du compte	
public Association CreationCompteAssociation(String,String,String,image,Ge stionnaire)	Création d'un nouveau compte association	-String Nom et Prenom -String Email -String Mot de passe -Image Justificatif sous forme d'image -Gestionnaire : le gestionnaire qui a permis la création du compte	
public Gestionnaire CreationCompteGestionnaire (String,String,String,Gestionn aire)	Création d'un nouveau compte gestionnaire	-String Nom et Prenom -String Email -String Mot de passe -Gestionnaire : le gestionnaire qui a permis la création du compte	
public boolean Authentification(String,String)	permet à l'utilisateur de s'authentifier, si il est déjà inscrit il sera redirigé vers l'accueil, sinon, une erreur sera déclenché.	-String Email -String Mot de passe	
public boolean checkFormulaireDemandeRe servation()	contrôle la saisi du formulaire de demande de réservation qui conditionne l'envoi des données de la réservation vers la base de données	/	
public void attributionSalles()	l'algorithme qui se chargera, à une heure	/	

	précise et bien définie, de traiter toutes les réservations contenues dans notre base de données et d'en attribuer des salles à la limite du possible. Enfin, toutes les réservations traitées seront supprimées à l'exception de celles qui se réitèrent.		
public boolean resAnnulee(Reservation)	annulation d'une réservation	-Reservation : la réservation à annuler	Boolean

3. Base de données:

Nous avons de nombreuses données à traiter et pour les organiser correctement, nous avons pensé aux bases de données qui forment un outil bien adapté.

Il existe plusieurs types de base de données. Celle que nous allons utiliser est de type relationnel qui est le modèle le plus répandu actuellement. Plus précisément MySQL. Un Système de Gestion de Bases de Données Relationnelles (abrégé SGBDR), c'est-à-dire un logiciel qui permet de gérer des bases de données, et donc de gérer de grosses quantités d'informations. Ce type de base de données repose sur les théories ensemblistes et l'algèbre relationnel. Les données sont organisées en tables possédant des relations entre elles grâce à des clés primaires et étrangères. Les opérations sur la base sont réalisées grâce à des requêtes SQL. Nous avons choisi MySQL pour la gestion des données car il s'agit d'un système libre et populaire qui assurera la pérennité du site grâce à sa robustesse et sa communauté.

- Stockage persistant :

Toute application web nécessite de stocker des données de façon permanente sur le serveur. Nous avons précisé précédemment notre choix d'utilisation de base de données MySQL. Et afin de stocker de façon permanente les objets persistants, nous utiliserons MySQL pour Node.js.

Les objets persistants sont les suivants :

- Utilisateur : Nous devons garder les informations sur les utilisateurs membres (leurs adresses mail, leurs identifiants, etc) de façon permanente.
- Réservation : comme toutes les demandes de réservation pour semaine donnée sont faites la semaines d'avant, elles (les demandes élémentaires) sont stockées dans notre base jusqu'à leur traitement. Après quoi, les demandes non satisfaites seront supprimées. Ensuite, après la création de l'emploi du temps pour la semaine d'après, toutes les demandes non récurrentes seront supprimées de notre base. Sauf celles qui sont renouvelables. Les demandes renouvelables Elles restent persistantes jusqu'à la date de fin d'itération.
- L'EDT : l'emploi du temps est stocké dans la base pour la durée d'une semaine. Il est mis à jour toutes les fin de semaines.
- Salles & équipements : Nous sauvegardons également les salles et les équipements.

III. Interfaces web:

- Page principale :


Un visiteur de notre site web tombe sur la page de connexion/inscription suivante:

The image shows a web interface for Paris-Sud University. At the top, there is a logo with a blue tag icon and the text 'PARIS-SUD UNIVERSITY'. Below the logo, a horizontal line separates it from the main content area. The main content area has a heading 'Welcome to our website!'. Below this, there are two columns. The left column is titled 'Connexion ici' and contains two input fields: 'Email Address' and 'Password', followed by a 'Connexion' button. The right column is titled 'Inscription ici' and contains a blue dropdown menu labeled 'Je suis un (e) :'. A line from the dropdown menu points to a callout box that shows the available options: 'Professeur', 'Etudiant (e)', and 'Association'.

- Compte Utilisateur :


- **Étudiant, Professeur et Association :**

Après sa connexion, l'étudiant ou le professeur ou l'association vera son profil sur notre site s'afficher comme suit :

		Planning de la semaine du 19 Avril 2019				
<input type="text" value="Username"/>		Lun	Mar	Mer	Jeu	Ven
<input type="button" value="Mon Planning"/>		08h				
<input type="button" value="Modifier mon mot de passe"/>		09h	8h30 - 10h bât. 1 Salle 22			
<input type="button" value="Demande de réservation"/>		10h		10h - 11h30 bât 2 Salle 01		
<input type="button" value="Annuler une demande de rése"/>		11h				
		12h				
		13h				
		14h				
		15h			15h - 18h bât 1 Salle 101	
		16h				
		17h				
		18h				
		19h				
		20h		20h - 21h bât 4 Salle 43		
<input type="button" value="Deconnexion"/>		21h				

- **Gestionnaire :**

Après sa connexion, le gestionnaire vera son profil sur notre site s'afficher comme suit:



Planning des salles

Modifier mon mot de passe

+

Demande de réservation exco

✓

Valider un compte

↺

Modifier une Salle

+

Créer une Salle

≡

Pôles

⏻

Deconnexion

Planning de la semaine du 19 Avril 2019

	Lun	Mar	Mer	Jeu	Ven
001					
002					
003					
004					
005					
006					
007					
008					
009					
010					
011					
012					
013					
014					
015					
016					
017					
018					
019					
020					

Activ
Accès

- Inscription étudiant:

Lorsqu'un étudiant décide de créer un compte il sera dirigé vers une page comme la suivante:



Créer votre compte

Pseudo * :

E-Mail * :

Confirmez E-Mail * :

Mot de passe * :

Confirmez le mot de passe * :

Inscription

Formulaire inscription étudiant

- Inscription professeur/association:

Un professeur ou une association peut créer un compte en remplissant le formulaire suivant :



The form is titled 'Créer votre compte' and is for 'PARIS-SUD UNIVERSITY'. It includes fields for 'Pseudo *', 'E-Mail *', 'Confirmez E-Mail *', 'Mot de passe *', 'Confirmez le mot de passe *', and 'Piece jointe *'. The 'Piece jointe *' field has a 'Parcourir' button and a text input field. An 'Inscription' button is at the bottom.



Créer votre compte

Pseudo * :

E-Mail * :

Confirmez E-Mail * :

Mot de passe * :

Confirmez le mot de passe * :

Piece jointe * :

Formulaire inscription professeur/association

La pièce jointe étant une justification de l'identité pour les professeurs (comme preuve qu'il est bien professeur). Et une demande explicative pour les associations.

- **Demande de réservation:**

Lorsqu'un utilisateur choisit de faire une demande de réservation à partir de son profil, il est amené à remplir un formulaire comme suit:

Formulaire de réservation

Nombre de personnes

Bâtiment

Type de salle

Équipement

[Ajouter équipement](#)

Créneau

Répéter ☐ Oui ☒ Non

Date fin

Voulez vous une alternative ? [Oui](#)

Nb de réservations restantes 12

Formulaire de demande de réservation

En cochant la case répéter, la demande de l'utilisateur va être renouvelé chaque semaine jusqu'à la date de fin.

En cliquant sur "oui", l'utilisateur va formuler une autre demande comme alternative.

En envoyant le formulaire, la demande sera enregistré sur la base de donnée et l'utilisateur sera redirigé vers l'accueil.

Pour formuler une autre demande l'utilisateur n'a qu'à refaire la même démarche.

Conclusion

Du point de vue de notre cahier de charge, ce projet est objectivement réussi. Tous les objectifs principaux ont été atteints : nous disposons de plusieurs méthodes, implémentant successivement les différentes fonctionnalités, et nous les avons implémenté de manière à pouvoir exploiter nos connaissances ainsi que les points forts des langages de programmation (côté client et côté serveur) que nous avons utilisé. Nous avons pu aussi nous intéresser aux objectifs secondaires. En particulier le renouvellement automatique des demandes a pu être effectué et a été l'occasion de nous familiariser avec la gestion des bases de données en MySQL avec Node.js. Cependant, il reste encore beaucoup de choses qu'il pourrait être intéressant d'essayer. Notamment, nous aurions bien voulu implémenter un algorithme travaillant sur des plannings placés dans les noeuds d'un arbre, ce qui nous aurait permis de faire un backtracking sur un arbre dont chaque noeud a un planning meilleur que son père, au lieu de faire un backtracking uniquement sur les demandes de chaque liste. Il est à noter que le gain de temps d'un tel algorithme ne sera sans doute pas immédiat, car passer à la position suivante (qui est l'opération la plus répétée de l'algorithme de backtracking) nécessite maintenant de générer un nouveau planning, ce qui est un peu coûteux. Il aurait aussi été logique de continuer de développer l'interface graphique pour nous permettre d'effectuer les opérations de mise à jour de l'emploi du temps et l'accès à l'ensemble des utilisateurs et des réservations (uniquement pour le gestionnaire) par son intermédiaire. Enfin, il serait également raisonnable de ne pas utiliser beaucoup de copier/coller de la base de données vers des objets JavaScript comme nous avons déjà un accès direct à la base.