

# Rapport de phase 2

## Phase 2 : Analyse et conception

Objectif : définir comment doit être construit le logiciel

### Besoins priorités + tests d'acceptation

Nous avons décidé de choisir les principales fonctionnalités de l'application pour avoir quelque chose de fonctionnel, et adapté pour un délai de développement assez court (environ 2 semaines).

- **Ajouter items**

- - **Test d'acceptation 1:**

- Connexion en tant qu'admin
- Affichage de la page d'accueil
- Clic sur le bouton pour ajouter un élément
- Remplissage de tous les champs
- Validation et observation du bon ajout de l'élément dans la liste

- - **Test d'acceptation 2 :**

- Connexion en tant qu'admin
- Affichage de la page d'accueil
- Clic sur le bouton pour ajouter un élément
- Remplir quelques champs sauf certains
- Essayer de valider, observer les champs non remplis en rouge
- Correction des champs
- Validation et observation du bon ajout

- **Faire une demande de réservation**

- - **Test d'acceptation 1 :**

- Connexion en tant qu'utilisateur
- Affichage de la page d'accueil
- Utiliser la barre de recherche pour trouver l'item à réserver
- Clic pour réserver l'item
- Indiquer une plage de temps
- Observer la bonne demande de réservation

- - **Test d'acceptation 2 :**

- (Créer au préalable un item réservé sur certaines périodes)
- Connexion en tant qu'utilisateur
- Affichage de la page d'accueil
- Utiliser la barre de recherche pour trouver l'item à réserver
- Clic pour réserver l'item
- Indiquer une plage de temps très grande (assez pour qu'il y ait déjà une réservation pendant cette plage)
- Observer le refus car il y a des réservations sur cette plage
- Annuler la réservation, et clic sur le planning
- Fermer le popup et re réserver l'item avec une plage valide

- Observer la bonne demande de réservation
- **Traiter demande de réservation : (besoin priorisé car l’item n’est pas vraiment réservé sans la confirmation)**

Rappel Wireframe:



*Page de gestion des notifications pour un utilisateur (administrateur)*

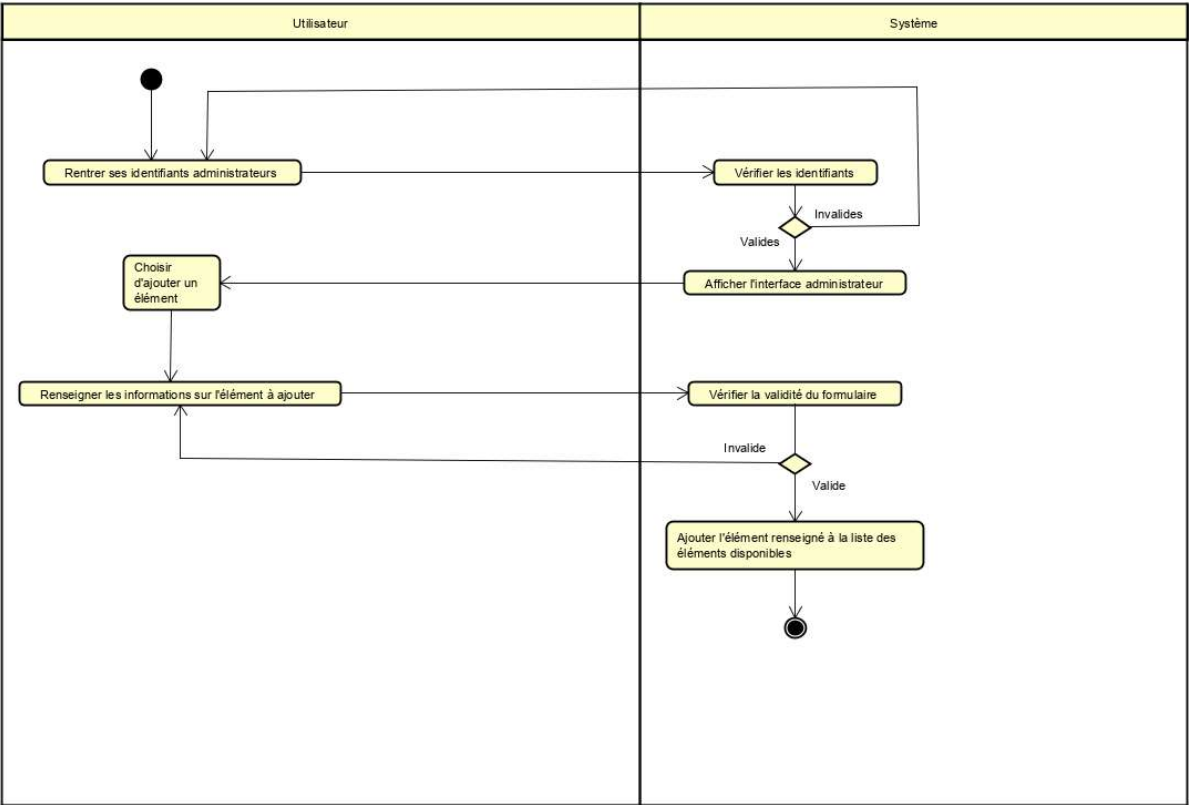
- - **Test d’acceptation 1 :**
    - (au préalable, faire une demande de réservation avec un autre utilisateur)
    - Se connecter en tant qu’administrateur
    - Affichage de la page d’accueil
    - Vérification de la présence de la notification
    - Clic sur la cloche pour afficher le/les notifications
    - Vérification de la demande de réservation en attente de validation, et des bonnes informations renseignées auparavant (préalable)
    - Clic sur Valider
    - Vérification des bons changements liés à la validation (cf. wireframe phase 1)
    - Se connecter en tant qu’utilisateur (celui qui a voulu réserver)
    - Vérifier dans les notifications que l’item a été accepté
  - **Test d’acceptation 2 :**
    - (au préalable, faire une demande de réservation avec un autre utilisateur)
    - Se connecter en tant qu’administrateur
    - Affichage de la page d’accueil
    - Vérification de la présence de la notification
    - Clic sur la cloche pour afficher le/les notifications
    - Vérification de la demande de réservation en attente de validation, et des bonnes informations renseignées auparavant (préalable)
    - Clic sur Refuser
    - Vérification des bons changements liés au refus (cf. wireframe phase 1)
    - Se connecter en tant qu’utilisateur (celui qui a voulu réserver)
    - Vérifier dans les notifications que l’item a été refusé

## Spécifications

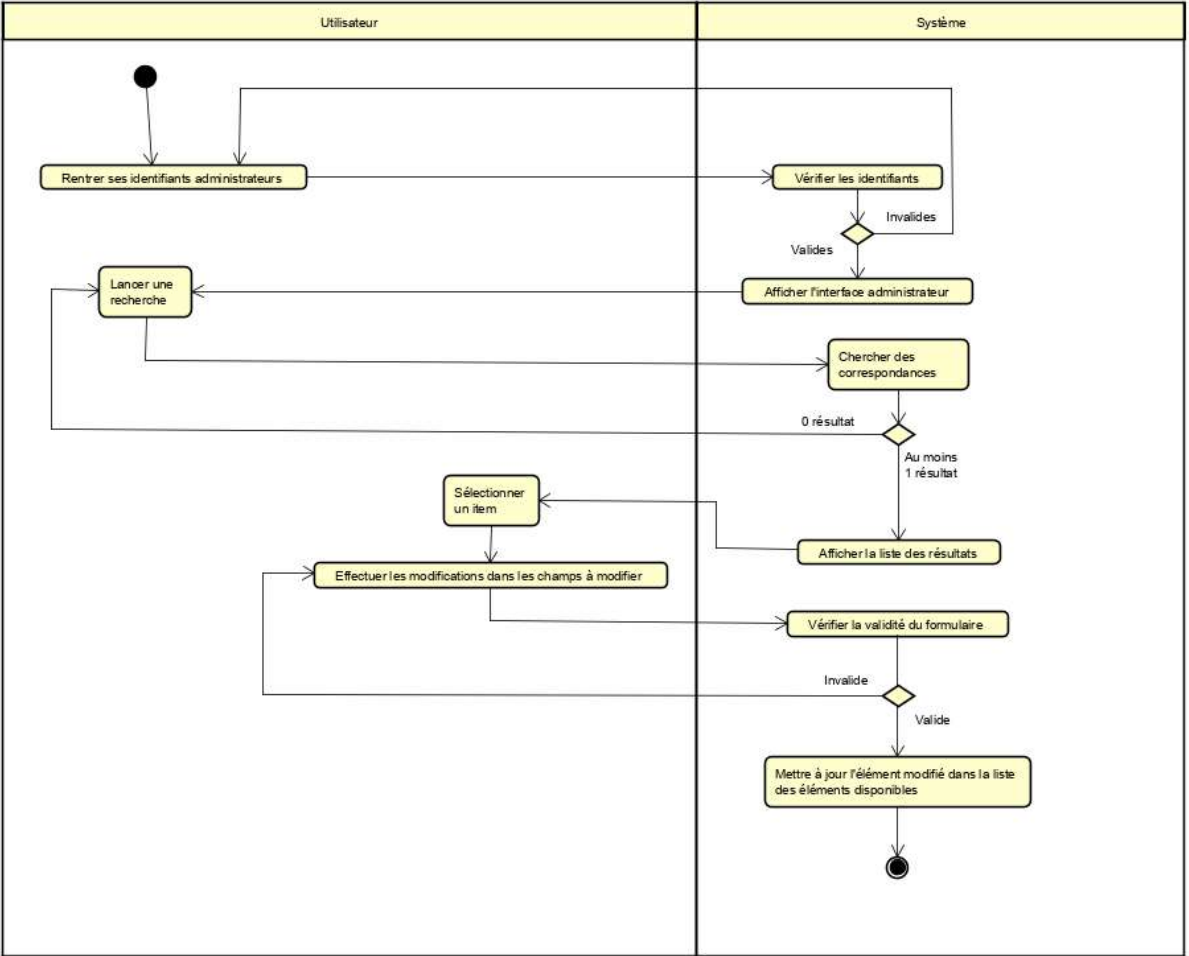
- **Architecture**

Diagrammes d'activité :

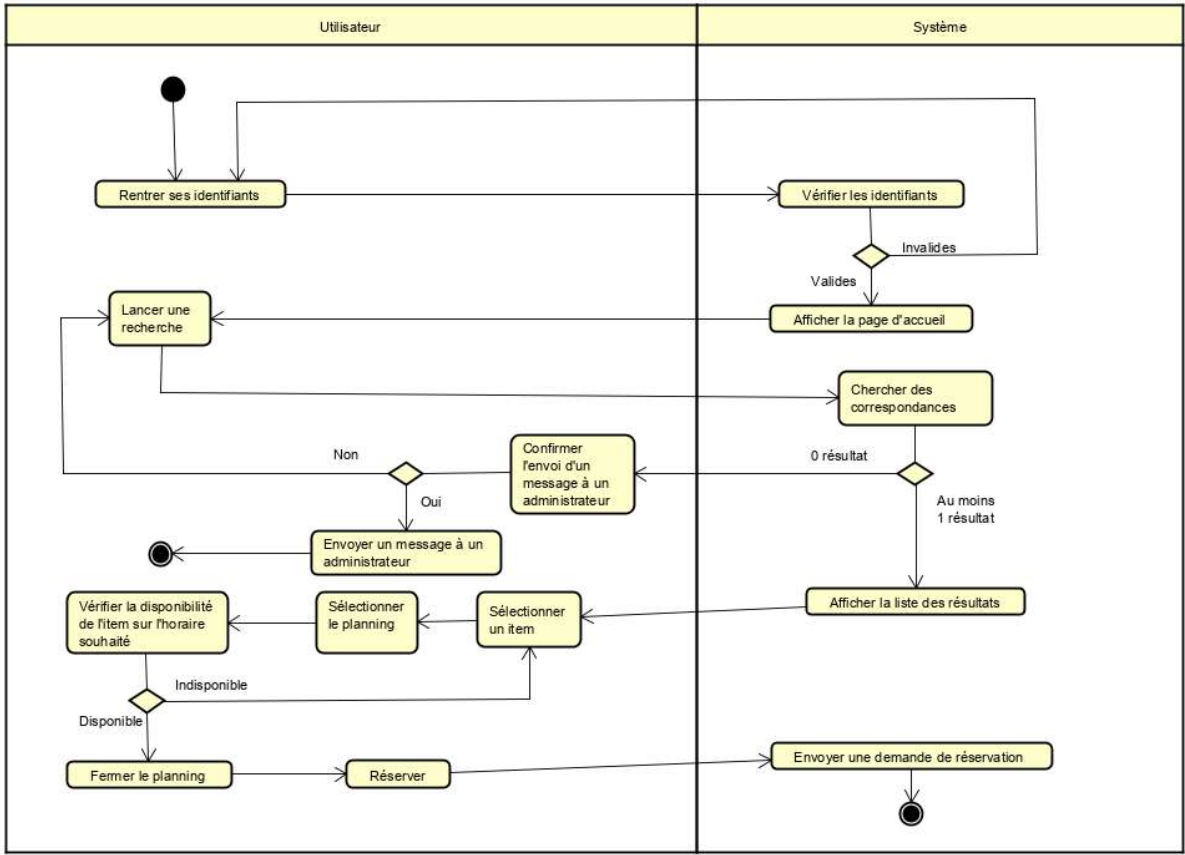
Ajouter un item :



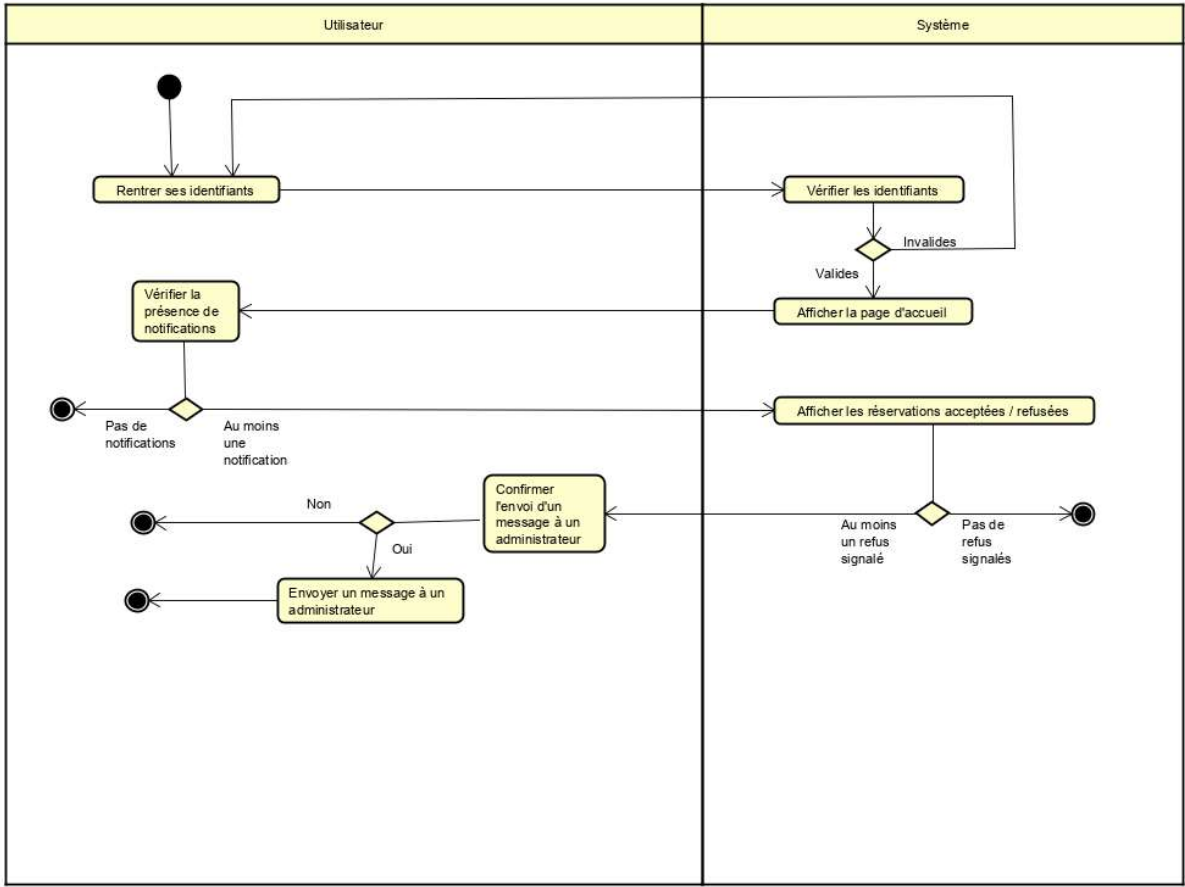
Modifier item (n'est pas dans les besoins priorisés) :



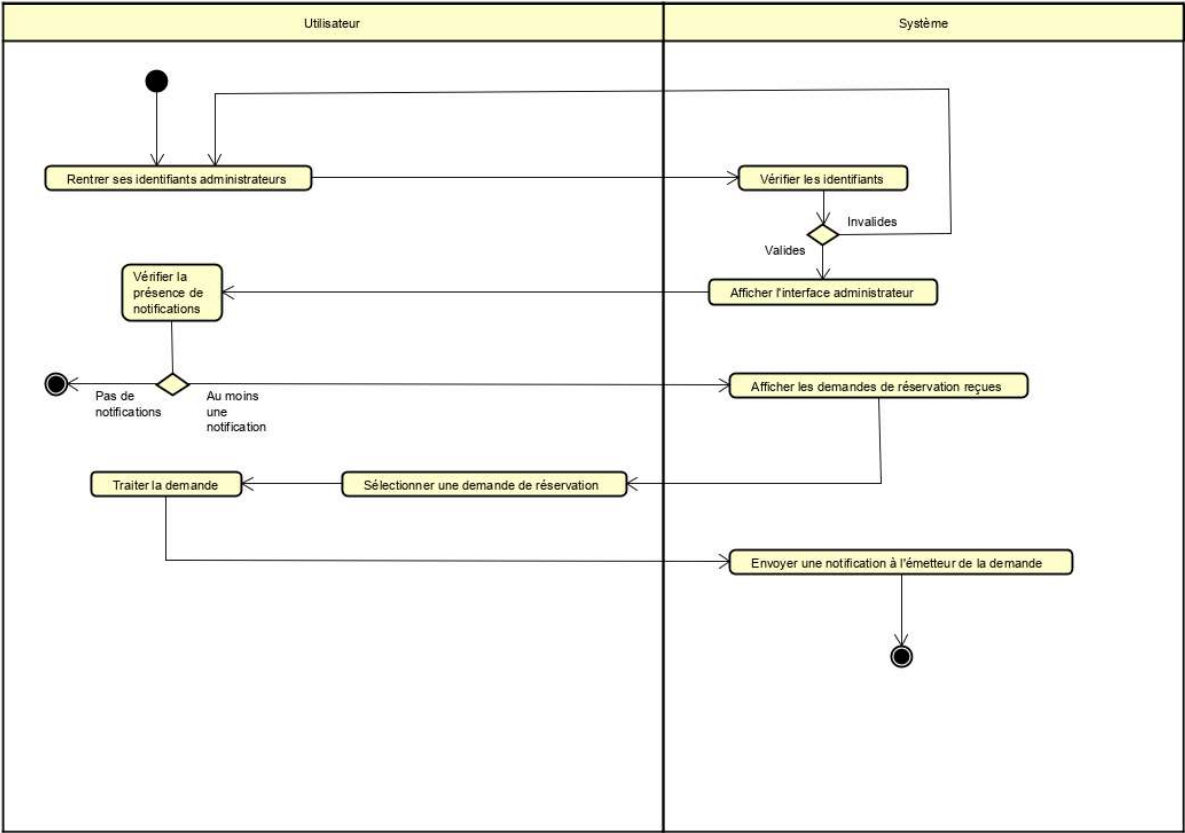
Demander réservation :



Consulter notifications :



Traiter demande réservation :



définition de l'architecture générale

La partie client du site sera développée en html, css, javascript. Nous recommandons l'utilisation de framework comme bootstrap et jquery, ou même angular, vuejs ou reactjs.

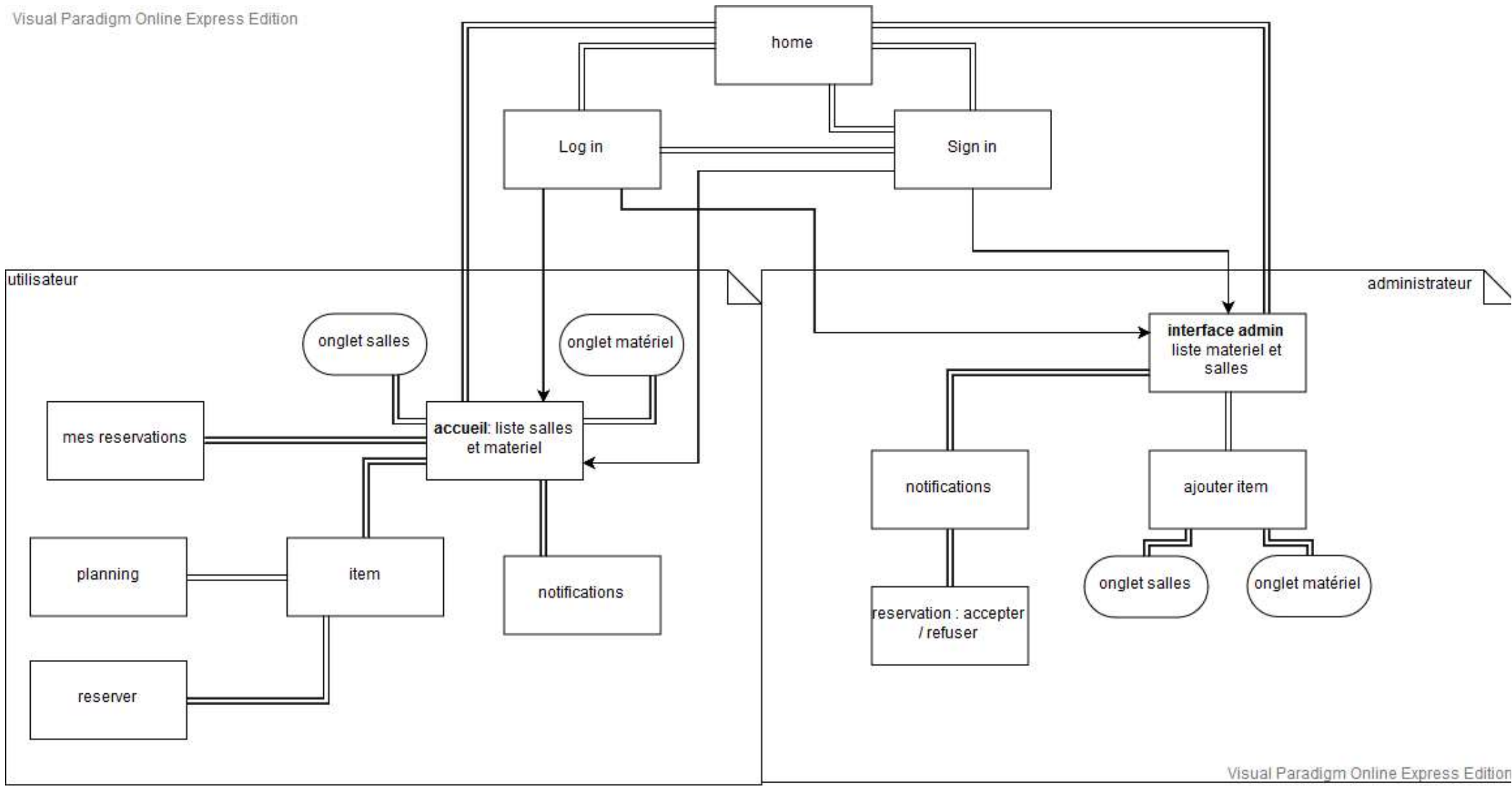
La partie serveur du site sera développée en javascript avec le framework expressJS.

Il y aura 2 modules : un pour l'utilisateur lambda et un autre pour l'utilisateur administrateur.

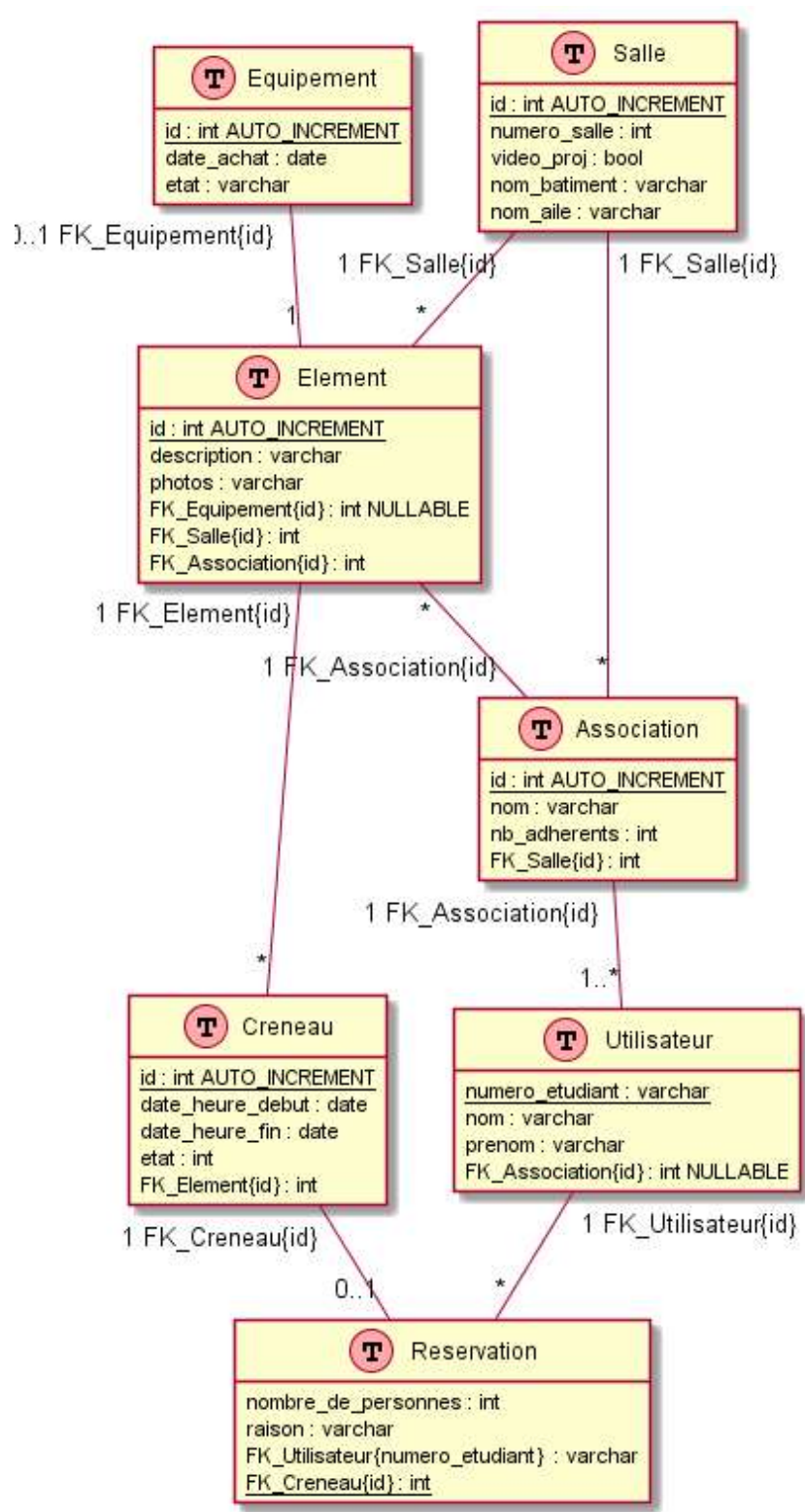
```
/src/  
  
  home.html  
  
  signin.html  
  
  login.html  
  
home/  
  accueil (liste matériel) (modales : planning, réservations, notifications)  
  salles (liste des salles) (modales : planning, réservations, notifications)  
  
css/  
  style.css  
  
js/  
  main.js  
  
admin/  
  accueil (liste matériel) (modales : notifications)  
  salles (liste des salles) (modales : notifications)  
  administration (liste des éléments que je propose)  
  
css/  
  admin.css  
  
js/  
  admin.js
```

La planning, la réservation et l'acceptation ou le refus d'une réservation se feront dans des modales, c'est pourquoi nous recommandons l'utilisation de bootstrap pour faciliter le développement de ces composants.

Schéma des interactions entre les pages



• Schéma de base de données



• API RESTful

# Element

## Ajouter un équipement

URL POST /api/element/equipement/add

Paramètres Requis:

- jour\_achat=[int]
- mois\_achat=[int]
- annee\_achat=[int]
- etat=[string]
- description=[string]
- photos=[string]
- alle\_id=[int]
- association\_id=[int]

Requête

```
INSERT INTO Equipement VALUES(:date, :etat);
INSERT INTO Element VALUES(:description, :photos, :last_inserted_equipement_id, :salle_id, :association_id);
```

## Ajouter une salle (et l'élément associé)

URL POST /api/element/salle/add

Paramètres Requis:

- numero\_salle=[int]
- nom\_batiment=[string]



- nom\_aile=[string]
- video\_proj=[boolean]
- description=[string]
- photos=[string]
- association\_id=[int]

Requête

```
INSERT INTO Salle VALUES(:name, :video_proj, :nom_batiment, :nom_aile);
INSERT INTO Element VALUES(:description, :photos, NULL, :salle_id, :association_id);
```

Lister les équipements

URL GET /api/element/equipement

Requête

```
SELECT * FROM Equipement;
```

Lister les salles

URL GET /api/element/salle

Requête

```
SELECT * FROM Salle;
```

Association

Ajouter une association

URL POST /api/association/add

Paramètres Requis:

- nom=[string]
- nb\_adherents=[int]
- salle\_id=[int]

Requête

```
INSERT INTO Association VALUES(:nom, :nb_adherents, :salle_id);
```

Lister les associations

URL GET /api/association

Requête

```
SELECT * FROM Association;
```

Utilisateur

Ajouter un utilisateur

URL POST /api/utilisateur/add

Paramètres Requis:

- nom=[string]
- prenom=[string] Optionnel:
- association\_id=[int]

Requête

```
INSERT INTO Utilisateur VALUES(:nom, :prenom, :association_id);
```

Créneau

Ajouter un créneau

URL POST /api/creneau/add

Paramètres Requis:

- heure\_debut=[int]
- minute\_debut=[int]
- jour\_debut=[int]

- mois\_debut=[int]
- annee\_debut=[int]
- heure\_fin=[int]
- minute\_fin=[int]
- jour\_fin=[int]
- mois\_fin=[int]
- annee\_fin=[int]
- etat=[int]
- element\_id=[string]

Requête

```
INSERT INTO Creneau VALUES(:date_debut, :date_fin, :etat, :element_id);
```

Lister les créneaux pour un élément

URL GET /api/creneau/:element\_id

Paramètres Requis:

- element\_id=[int]

Requête

```
SELECT * FROM Creneau WHERE FK_Element=:element_id;
```

Reservation

Ajouter une reservation

URL POST /api/reservation/add

Paramètres Requis:

- nombre\_de\_personnes=[int]
- raison=[string]
- numero\_etudiant=[string]
- creneau\_id=[int]

Requête

```
INSERT INTO Reservation VALUES(:nombre_de_personnes, :raison, :numero_etudiant, :creneau_id);
```

Lister les reservations

URL GET /api/reservation

Requête

```
SELECT * FROM Reservation;
```

Ajouter une reservation

URL GET /api/reservation/:reservation\_id

Paramètres Requis:

- reservation\_id=[int]








Requête

```
SELECT * FROM Reservation WHERE id=:reservation_id;
```

• Planification développement Gantt

Le diagramme de Gantt permettra de montrer un plan pour les différentes phases de développement pour l’application et de baliser les fonctionnalités à développer. Celles-ci sont définies dans la partie “Besoins priorités”.



	Start Date	End Date	Timeline	Status
<b>Polylend</b>	May 14, 2019	May 31, 2019		
Structure du site (Front-End)	May 14, 2019	May 19, 2019		Upcoming
Création d'un item	May 14, 2019	May 18, 2019		Upcoming
Réservation d'un item	May 19, 2019	May 23, 2019		Upcoming
Notifications	May 20, 2019	May 24, 2019		Upcoming
Validation	May 25, 2019	May 27, 2019		Upcoming
Préparation du rendu	May 26, 2019	May 31, 2019		Upcoming

Cette planification nécessite de bien de répartir les tâches entre membres du projet, et de bien respecter les délais car les tâches sont liées entre elles (par exemple la création d'un item et la réservation).

## Retour d'expérience

Au niveau des besoins priorisés, il était difficile de faire des choix sur les développements des fonctionnalités prioritaires, puisque souvent, on a envie de vouloir tout faire pour avoir un produit qui soit complet. Mais il a fallu ici faire des choix car le temps qu'il restait pour la phase de développement ne pouvait pas permettre de tout faire. Cet exercice était intéressant puisqu'il nous a permis de nous focaliser sur le plus important.

Le peu de temps disponible pour la dernière phase a aussi été un problème pour la planification avec le diagramme de Gantt. En effet, il n'est pas forcément facile d'établir un plan, de baliser les fonctionnalités à développer sur seulement 2 semaines. Il faut aussi arriver à estimer le temps qu'il faudra pour la développer, ce qui nécessite donc de prévoir la complexité de la tâche. Même si nous n'allons au final pas suivre ce diagramme puisque nous allons développer un autre sujet, cela reste intéressant pour nos futurs projets car nous arriverons mieux à faire des estimations de temps et de charge de travail, ce qui est très important pour tout projet.

La réalisation du diagramme d'interaction entre les pages du site nécessite une vision globale du projet. De plus, il faut bien prendre en compte uniquement les aspects du projet qui seront effectivement développés. C'est un exercice intéressant car il permet de déceler des problèmes ou lacunes dans le cahier des charges.

Les diagrammes d'activité étaient relativement simple à réaliser car l'application n'est pas très complexe et son fonctionnement est assez instinctif. Cependant, certains points étaient plus difficiles car certaines spécifications fournies par le groupe précédent n'étaient pas très claire ou pas très instinctives. Il a alors parfois fallu suivre ce qui était spécifié, même si de notre point de vue, une solution différente était peut-être meilleure. Il ne faut donc pas oublier que le client est roi.