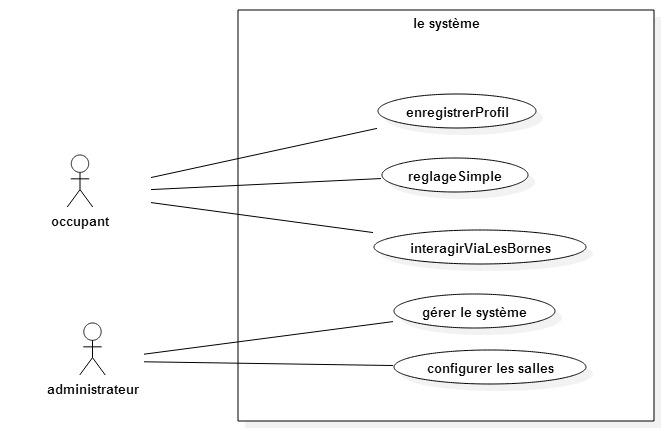
**SLR202\_UML --- Micro Projet**

GUO Lieqiang

[Lieqiang.guo@telecom-paristech.fr](mailto:Lieqiang.guo@telecom-paristech.fr)

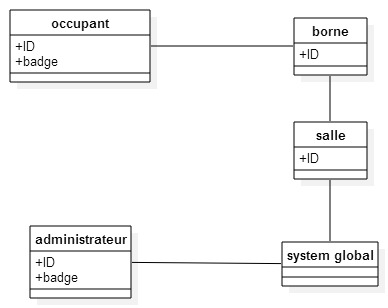
1. Itération :
   1. Le cas d’utilisation :



En générale, il y a des occupants et un administrateur interagir via le système de smart Institute, les occupants peuvent enregistrer leur profils et interagir via les bornes dans chaque salle ;

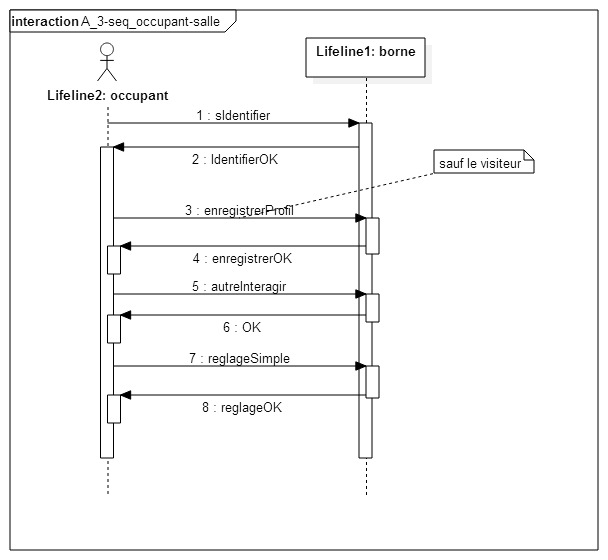
C’est l’administrateur qui peut gérer le system et configurer les salles.

* 1. Les classes principales :

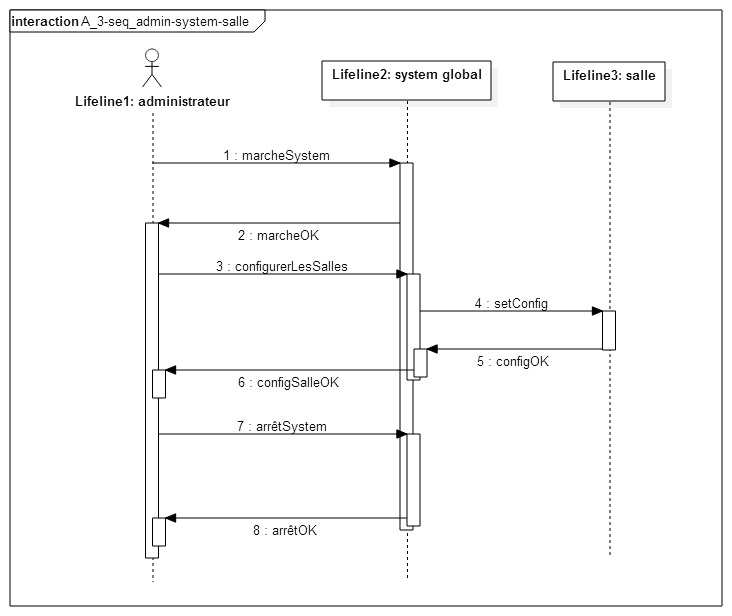


C’est le diagramme de class plus simple , il y a pas d’autre détail. Il y a Seulement les Classes principaux, un administrateur associe le système global, et le système contrôle les salles, les occupants interagissent via les bornes dans chaque salle, chaque salle a une borne.

* 1. Les diagrammes de séquence système :

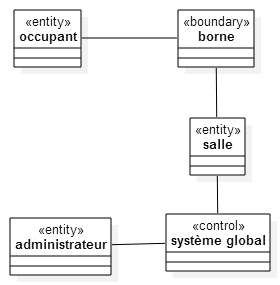


C’est la séquence entre l’occupant et la borne, premièrement, l’occupant doit se identifier sur la borne, quand en identifiant, il peut faire quelques choses comme réglage simple, enregistrer son profil, mais si le occupant est un visiteur qui n’a pas profil, il ne peut pas faire ça. Il peut aussi faire d’autres choses si nécessaire.



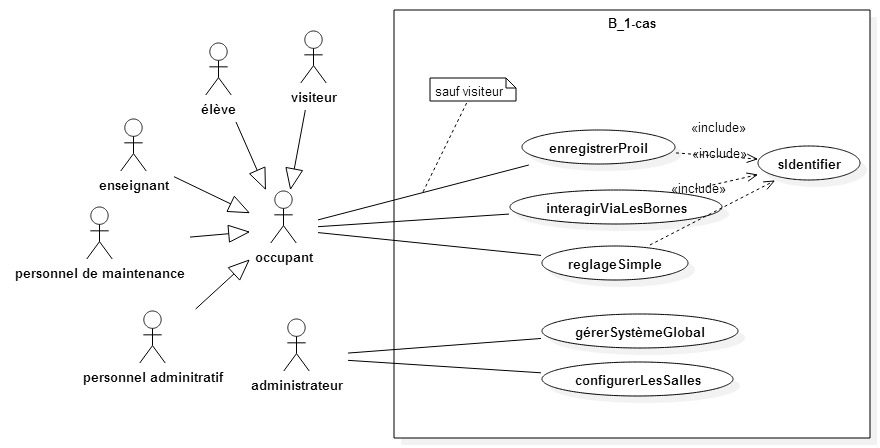
L’administrateur peut marcher le système et l’arrêter, il peut aussi configurer les salles via le système d’où le système va utiliser le méthode de salle pour faire ça.

* 1. Le MVC :



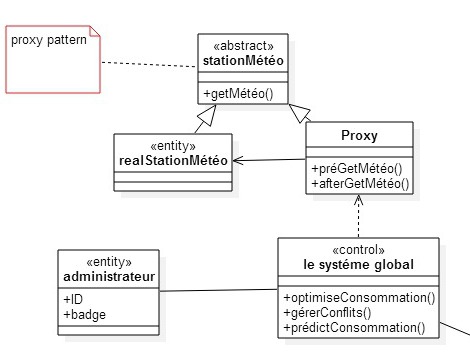
Le occupant l’administrateur et le salle sont ‘entity’, et la borne est ‘boundry’, le système global est le contrôleur.

1. itération :
   1. Le cas d’utilisation :



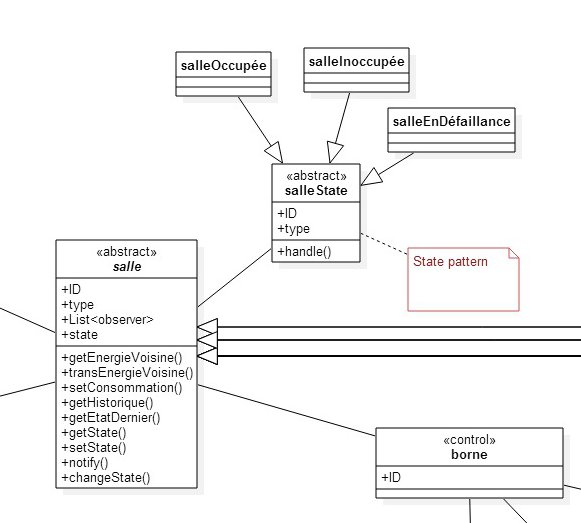
Il y a cinq types d’occupant, le visiteur, l’élève, l’enseignant, le personnel de maintenance, et le personnel administratif. Tous les occupants peut régler la salle selon leur badge, et le visiteur n’a pas profil, donc il ne peut pas faire enregistrer le profil, tous les occupants doivent se identifier avant interagir via la borne d’une salle, et la borne va vérifier ses badges et déterminer ses droits selon la salle où la borne se trouve, par exemple, si un élève s’identifier sur la borne dans le forum, il ne peut pas régler le forum, puisque la borne ne le donne pas ou ne le répond pas.

* 1. Design pattern-Proxy pattern :



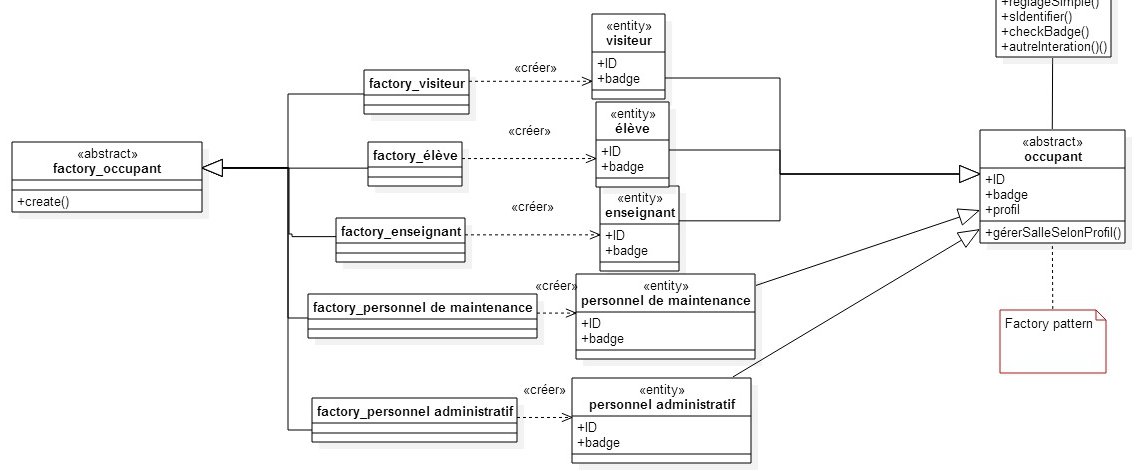
Puisque l’algorithme dans le système global doit prédiquer la consommation d’énergie de chaque salle, le système global doit communiquer avec la station météo extérieure, j’utilise proxy pattern pour faire ça. On peut ajouter nouveaux services facilement pour le système global.

* 1. Design pattern-State Pattern :



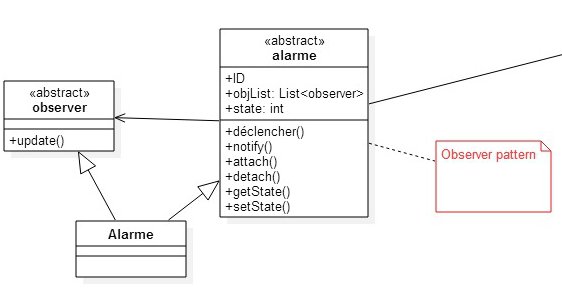
Une salle a trois états, occupée, inoccupée, et en défaillance, c’est plus facile de changer la manière de salle, quand elle est inoccupée, s’il y a un occupant entrer dans la salle, elle peut changer l’état vers l’occupée automatiquement, si elle est en défaillance, l’état va changer vers la défaillance tout de suite.

* 1. Design pattern-Factory Method Pattern :



On utilise ‘factory method’ pattern pour créer les occupants différents, c’est pratique quand on ajoute un nouveau type d’occupant, on doit juste ajouter un nouveau ‘factory’ et un nouveau class de ce type d’occupant.

1. Design pattern-State Pattern :



Puisque une dans une salle

1. f
2. f
3. f
4. f
5. itération :