Rapport Projet CSI

Système d’Information de Gestion d’une salle de sport



Membres du groupe

CALLEY Nicolas

CANTE Benoît

DIALLO Abdoulaye

DOLET Maxime

# 

# 

# Compte-rendu de l’interview du 10/11/2017

Suite à l’interview avec les clients (Mme Brun et Mr Vigneron), nous avons noté les points suivants:

1. Les noms de logins et mots de passe sont générés par le système, une fois que l’inscription est validée.
2. Il est possible de payer l’inscription directement auprès du personnel ou depuis le site Internet.
3. Un adhérent peut inviter autant de personnes qu’il veut à une séance, dans la limite des places disponibles.
4. Il est également possible d’acheter des cartes 10 séances ou des inscriptions à une seule séance depuis le site.
5. Il est impossible pour un abonné de voir le noms des autres personnes inscrites à une séance (mais un coach ou un employé le peut).
6. L’administrateur doit pouvoir supprimer des comptes inactifs depuis au moins un an. Ceci doit être “semi-automatique”, c’est à dire qu’on propose une liste de compte à supprimer avec pour éventualité de cocher tous les comptes affectés et les supprimer.
7. Un compte peut être désactivé (temporairement, à l’inverse d’une suppression).
8. On considère qu’un compte est inactif si 1 an est passé depuis sa dernière connexion.
9. Une personne doit d’abord payer des “frais de dossiers” et s’être inscrit sur le site, pour pouvoir ensuite acheter des cartes ou un abonnement, ou encore payer à la séance.
10. Une personne ayant un abonnement doit payer un supplément si elle demande à un coach de venir alors que la séance n’en prévoit pas.
11. Il est, également, possible pour un adhérent d’avoir une carte d’abonnement mensuel ou annuel et un pass 10 séances.

# Résultat de la phase de repérage du domaine

## Objectifs du projet

On part d’une salle de sport existante qui à un degré d’ouverture encore assez faible du SI vers l’extérieur. En effet, on souhaite que les adhérents à la salle de sport, les employés ainsi qu’un administrateur puissent dialoguer avec le SI. L’objectif est donc de permettre à ces différentes types de personnes d’être enregistrées dans notre base de données. Ceci doit permettre aux personnes n’ayant pas d’espace client de pouvoir s’inscrire directement à partir de la plateforme et permettre aux personnes possédant un espace client (statut : adhérent, employé, ...) de s’inscrire et gérer leur participation à des séances sans nécessairement se rendre physiquement à la salle de sport.

## Limites du domaine concerné par le projet

Le domaine du projet concerne la gestion des inscriptions des clients aux différentes séances. De ce fait, notre domaine couvre l’inscription des clients de la salle de sport aux activités, et les interactions entre clients concernant les inscriptions aux séances.

Le paiement ne fait pas partie de notre domaine, de même que tout ce qui concerne les ressources humaines.

## Listes des acteurs

Nous avons identifié les acteurs suivants:

### Acteurs internes

* Adhérent
  + C’est notre client: il paie pour accéder aux activités de notre salle.
* Administrateur
  + C’est une sorte de super-utilisateur: Il dispose de fonctions étendues par rapport à un employé de base, comme par exemple pouvoir choisir quels comptes inactifs supprimer.
* Employé
  + Il permet aux adhérents qui ne passent pas par le site web d’interagir avec le SI (par exemple, un adhérent peut acheter sur place une carte 10 séances auprès d’un employé au lieu de l’acheter en ligne).
* Coachs
  + Ce sont des personnes qui animent certaines séances. Ils peuvent également assister à des séances où aucun coach n’était prévu, mais ou un adhérent fait la demande.

### Acteurs externes

* Visiteur
  + Il n’est pas inscrit sur le site (à la salle de sport), il ne peut donc que regarder les informations générales sur la salle (par exemple, les montants des frais d’adhésion, des cartes d’abonnement, ...) et adhérer à partir à partir du site ou en se rendant à la salle de sport.
* Salles partenaires:
  + Nous avons pour objectif de communiquer certaines informations avec des salles de sports partenaires, pour proposer à nos adhérents de s’y rendre si jamais il n’y a plus de place dans notre salle.
* Service contentieux:
  + En cas d'impayé ou de problème juridique.
* Service facturation
  + règlement des cartes 10 séances.
  + règlement des abonnements.
* Gestion Clients:
  + Contient des informations détaillées sur nos clients.
* Gestion RH:
  + Contient des informations détaillées sur le personnel que nous employons (par exemple les coachs).

### Droits et groupes

#### Remarques:

* Certains acteurs ne font pas partie de notre analyse et ont pourtant des droits sur la base de donnée. par exemple, pour insérer les données relatives aux employés, c’est au rôle des RH de renseigner ces informations. Cependant, ils ne font pas partie des rôles que nous avons identifié
* Pour certain profils, certaines de leurs actions nécessitent un droit CRUD sur la base de donnée. Par exemple, quand un inscrit non adhérent paye ses droits d'inscriptions, cela revient à mettre à jour l'attribut "date\_paiement" de la ligne correspondant à l'utilisateur.
* Nous ne savons pas si c'est un droit dont l'utilisateur hérite directement, car c'est lui qui est à l'origine de l'action, ou si c'est un droit qui revient à un des personnels de la salle, qui fera l'opération d'update pour lui. Initialement la première solution nous paraît plus claire. Cependant après réflexion, nous la trouvons contre-intuitive. Par exemple, pour un visiteur: Celui-ci à le droit de s’inscrire sur notre site, ce qui revient à insérer une ligne dans la table Adhérent (et par extension Membre Inscrit, car Adhérent hérite de cette table). Nous avons beaucoup hésité à ce sujet, mais nous pensons que dire que n’importe quel visiteur à le droit d’insertion sur ces deux tables et incorrect. Notre idée est que dans pareil cas, c’est une fonction sur la base de donnée qui fait l’insertion, et pas directement le visiteur.
* Dans notre tableau récapitulatif des droits, nous utilisons la mention “[Select/Update] pour ses infos”, par exemple pour les droits du groupe “Adhérent” sur la table “employé accueil”. Cela signifie qu’un adhérent donné ne peut faire une opération de sélection que sur les informations qui le concernent directement.

#### Visiteur:

* Le visiteur ne peut qu'accéder aux informations générales sur la salle. On lui laisse regarder les informations concernant les séances en général pour qu'il ait envie de s'inscrire. Il peut également consulter les informations concernant les salles.
* Le visiteur peut consulter les prix des cartes et des abonnements afin d'avoir une indication sur les prix. Ceci correspond à une sélection sur les tables cartes et abonnements.
* Finalement, il peut déposer un dossier d'inscription en vue de devenir un adhérent. Cela se traduit par une insertion dans la table adhérent.

#### Inscrit non adhérent:

* Ce profil correspond aux personnes qui ont donné les informations nécessaires pour s'inscrire, mais qui n'ont pas encore réglé les frais d'inscriptions. Ces personnes ont donc un droit de mise à jour sur la table adhérent (pour changer certaines des informations personnelles), ainsi que de sélection sur cette table afin de récupérer les informations les concernant.

#### Adhérent:

* Ce profil correspond aux personnes ayant effectivement payé les frais de dossiers en plus d'avoir donné les informations nécessaires. L'adhérent peut s'inscrire à des séances, consulter / acheter des abonnements / cartes, demander à être coaché, inviter des personnes à des séances, accepter ou refuser des demandes d'invitations.
* La sélection permet d'avoir des informations sur les réservations qu'il a effectué (et pas celle d'autres adhérents).
* L'adhérent peut désormais acheter des cartes, ce qui se traduit par une insertion sur la table de jointure entre Adhérent et Carte. Il n’insère pas la ligne directement mais doit passer par une fonction qui va le faire pour lui.
* L'adhérent peut acheter une carte, ce qui se traduit par l'insertion d'une ligne dans la table de jointure entre Adhérent et Abonnement. Il n’insère pas la ligne directement mais doit passer par une fonction qui va le faire pour lui.
* L'adhérent peut consulter les séances auxquelles il a déjà assisté dans HistoriqueSéance. Les insertions dans cette table se font par l’intermédiaire d’un trigger, et pas directement par l’adhérent

#### Coach:

* Le coach peut voir quelles personnes sont inscrites aux séances, et il peut consulter le nom des adhérents inscrits à une séance qu'il gère. Il peut décider de coacher une séance selon la demande d'un adhérent. Il peut aussi modifier certaines de ses informations personnelles, ainsi que les consulter.
* Si le coach accepte de coacher une séance, alors on met à jour l'attribut dans séance qui indique quel coach gère la séance. Il faut donc que le coach puisse indiquer que c'est lui le responsable de la séance
* Le coach doit connaître les noms des inscrits à sa séance. Donc il à un droit de selection sur adhérent (restreint au nom) pour toutes les personnes participant aux séances qu'il coache.

#### Employé accueil:

* Ils permettent de faire l'interaction entre le SI et les adhérents (ou personnes qui veulent le devenir) qui ne passent pas par le SI
* Ces opérations permettent à l'employé de créer des réservations, les supprimer, les accepter, ou simplement les consulter.
* Ceci permet d'ajouter un nouvel adhérent (quand son dossier de candidature arrive par exemple) et d'indiquer quand il a payé ses droits d'inscriptions. il peut, à la demande d'un adhérent, modifier les informations de ce dernier
* Dans le cas ou il faut changer le nom d'un coach associé à une séance
* Si un adhérent veut acheter une carte en salle, l'employé peut enregistrer cette modification en insérant une ligne dans la table de jointure entre Abonnement et Adhérent.
* A la demande d'un adhérent, il est possible de consulter son historique de séances auprès de l'employé.

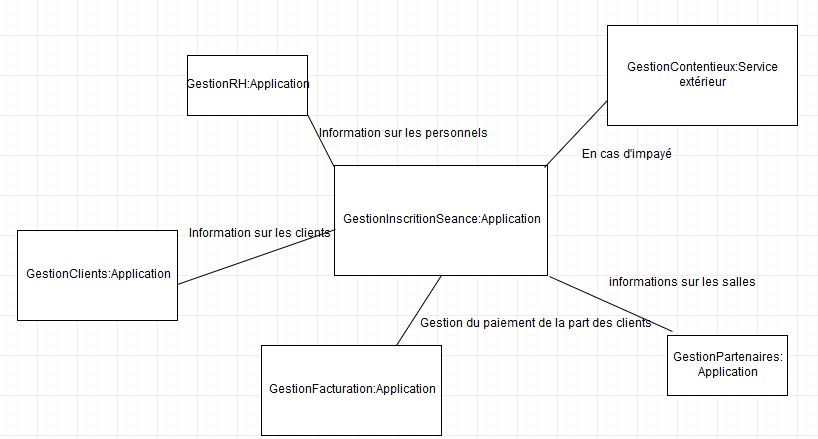
#### Administrateur:

* Tous les droits

#### Salle partenaire:

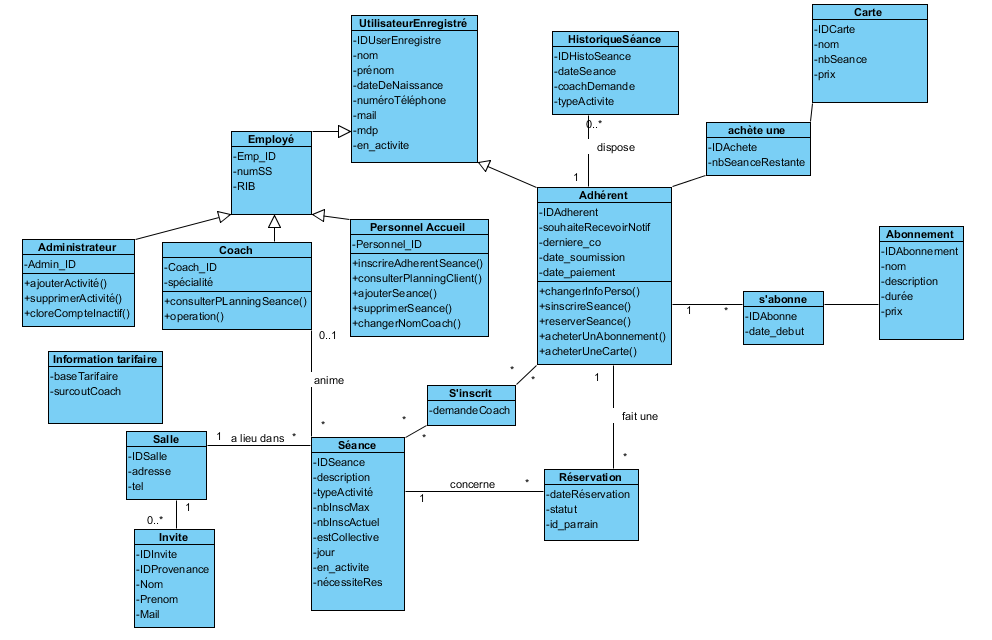
* Tous les droits de visiteurs

## Diagramme de communication



# Phase de découverte des informations & Workflow

## Diagramme de classes



## Commentaire sur le diagramme de classes

* Nous n'avons pas mis d'association entre Adhérent et Séance, bien qu'il soit possible pour un adhérent de s'inscrire à une séance sans passer par une réservation. Il ne nous semblait pas pertinent de représenter ce lien par une association.
* Nous avons changé la manière dont les invitations sont gérées pour éviter les abus :
* Quand un adhérent s’inscrit à un cours et veut inviter des amis au cours, un lien personnel est généré, il peut ensuite le partager à ses amis.
* Chaque personne qui va s’inscrire avec ce lien va générer une ligne dans la table “Réservation” avec “id\_parrain” valant l’id de l’adhérent qui a partagé le lien.
* Cette nouvelle gestion facilitera une éventuelle implémentation d’un système de parrainage.

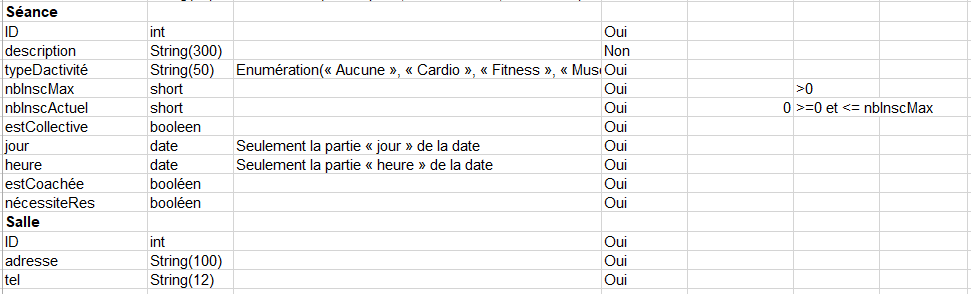
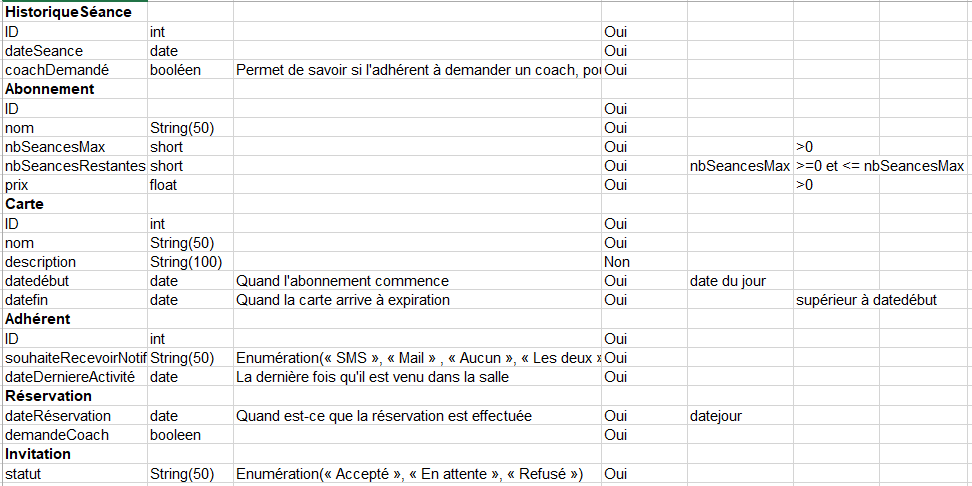
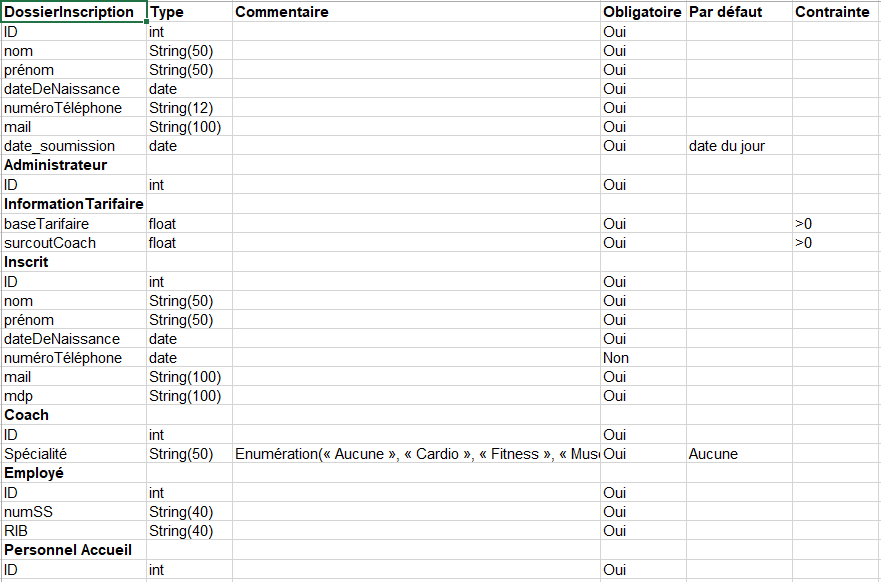
Nous avons choisi de supprimer la table "InvitationRéservation" pour centraliser toutes les réservations dans la table réservation.

* Nous avons pris pour principe de spécialiser les employés en catégories distinctes, même si cela laisse certaines d'entre elles avec très peu d'attributs différents. Cependant, nous avons trouvé ce choix judicieux car les méthodes qu'ils auront changent beaucoup d'une classe à une autre.
* Nous avons découpé heure et jour dans la classe Séance, car il faudra proposer à l'utilisateur des séances variant sur ces deux critères si jamais il ne trouve pas une séance qui l'intéresse directement.

La demande de coach est maintenant portée par une classe intermédiaire s’inscrit ce qui permet une gestion plus claire de la demande.

Lorsqu’une personne venant d’une salle partenaire veut accéder à une de nos séances on lui crée un compte de classe Invité qui a juste le droit de s’inscrire à la séance.

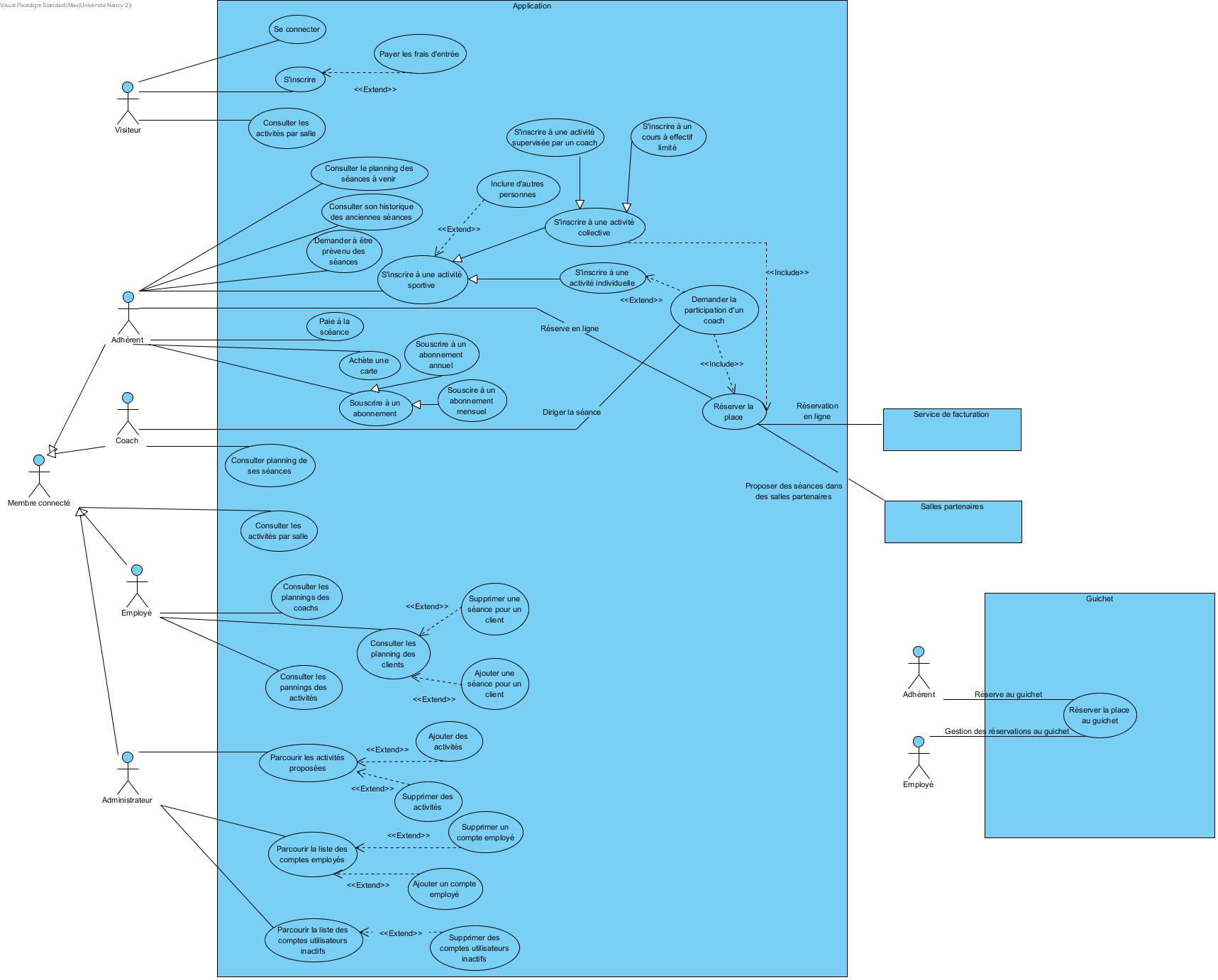
## Dictionnaire de données

****

# Phase de modélisation du Workflow

## Diagramme des cas d’utilisation

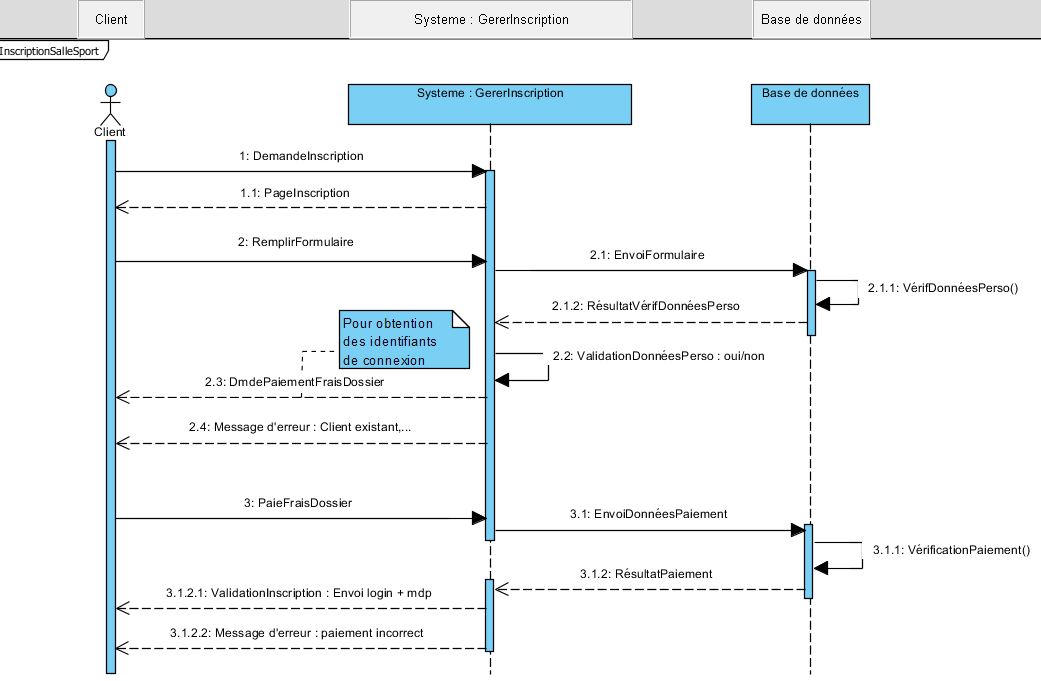
Nous avons choisis d’utiliser les compositions pour simplifier la lecture du diagramme. Nous avons eu quelques difficultés à fixer le niveau de détail de notre diagramme , nous avions une version plus détaillée, mais nous nous sommes ravisés, favorisant ainsi la facilité de lecture.



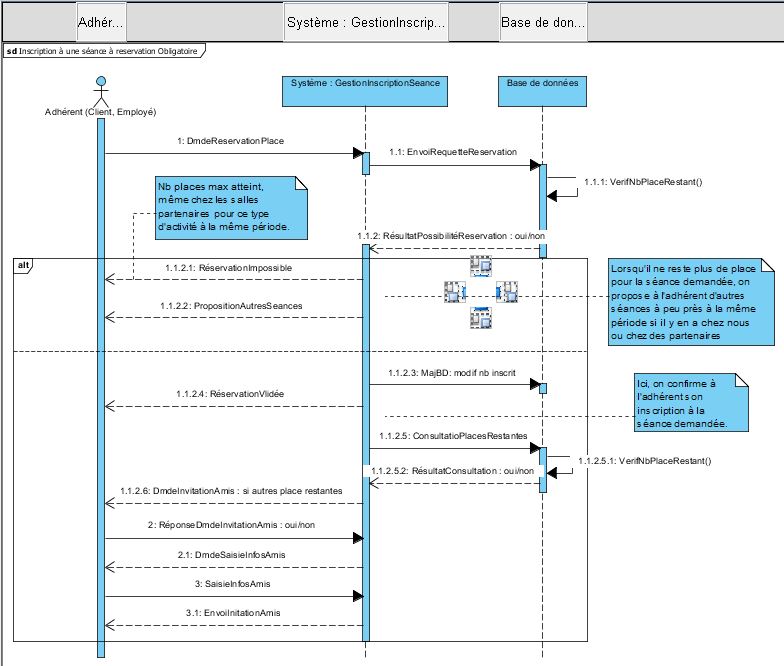
Dans cette nouvelle version nous avons externalisé les actions qui ne sont pas traitées par le SI. Comme vous nous l’aviez fait remarquer dans notre dernier rendu, nous avons déplacé les actions traitées au guichet.

## Diagramme de séquence

### Inscription à la salle de sport & paiement des frais de dossier

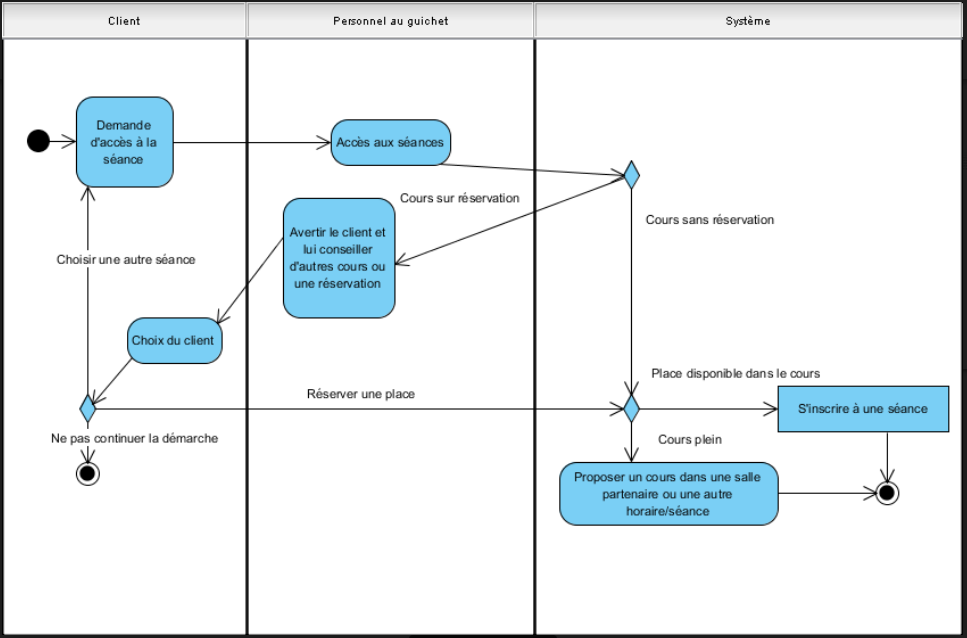


### Inscription à une séance

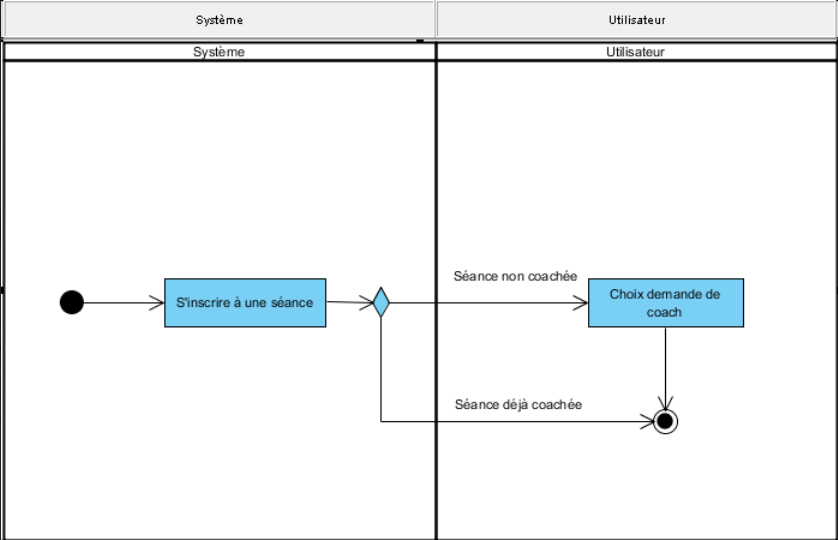


## Diagrammes d’activité

### Arriver sans réserver



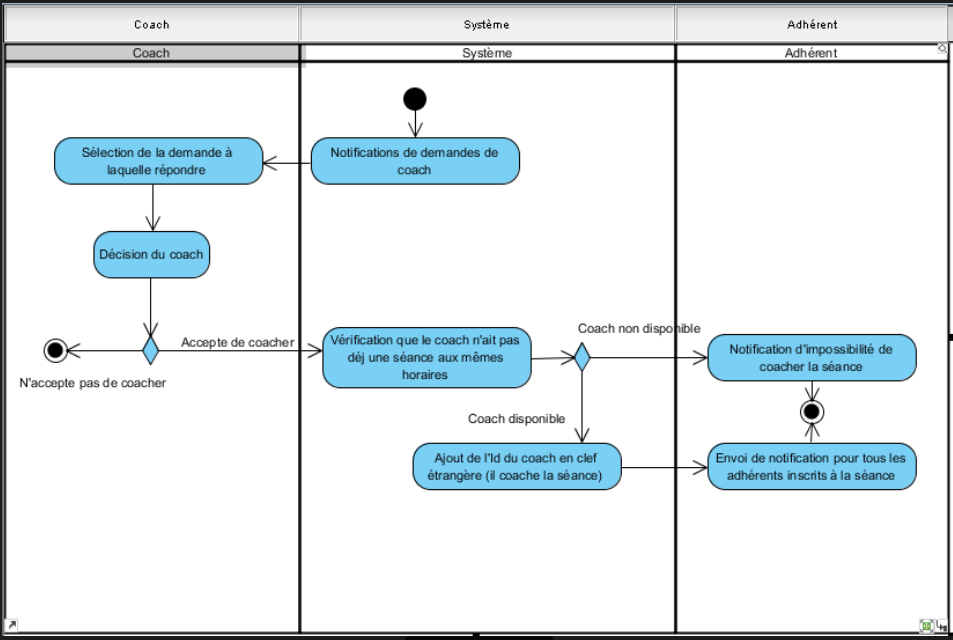
### Demande de coach



Commentaires:

Quand l'adhérent veut faire une réservation pour une séance, on regarde si la séance est coachée. Si elle ne l'est pas, on propose à l'adhérent de demander un coach. On mémorise son souhait, et la notification aux coaches sera envoyé depuis le diagramme principal si l'adhérent a demandé un coach et qu'il peut payer.

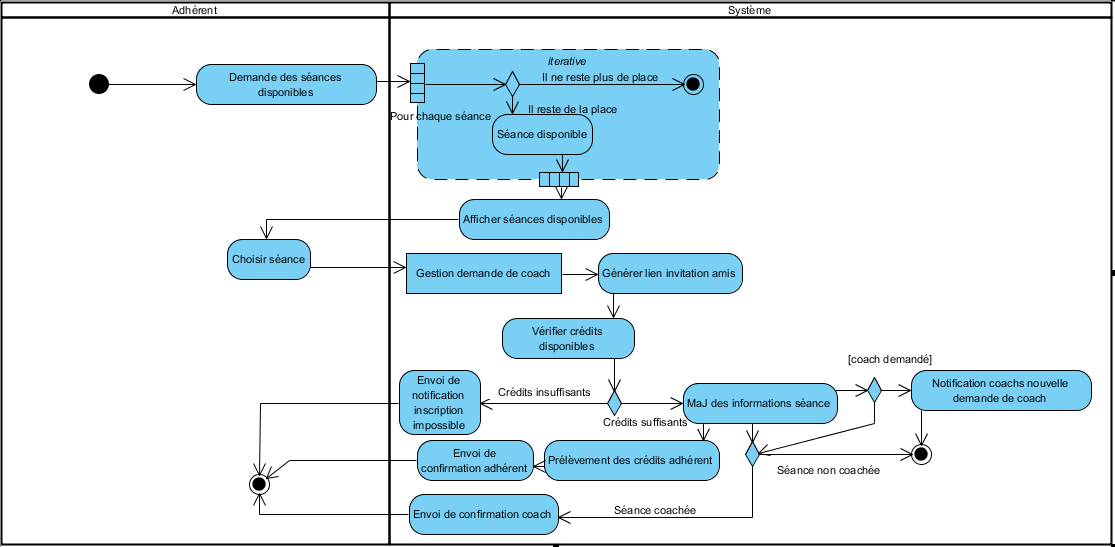
### Choix de coach



Commentaires:

Le coach choisi depuis un menu (basé sur les réservations qui ont un attribut demande de coach à vrai, et dont la séance associée à un id\_coach null) une séance qu'il souhaite coacher. Si il souhaite toujours coacher cette séance, il le confirme, et le système vérifie que la séance concernée n'a pas lieu au même moment qu'une autre séance qu'il coach (en regardant les séances ou l'id de coach correspond à l'id de coach enregistré)  
Si il n'y a pas de problème, alors on enregistre que le coach s'occupe de cette séance, et on notifie tous les adhérents inscrits à la séance.

#### S’inscrire à une séance



Commentaires:

L’adhérent demande à voir les séances disponibles, le système lui renvoie toutes celles où il reste de la place, l’adhérent choisit la séance qu’il veut puis choisit s’il veut un coach et/ou générer un lien pour inviter ses amis à la séance. Ensuite on vérifie ses crédits, s’il n’en a pas assez on le lui dit et on l’envoie sur la page de rechargement (les informations qu’il a déjà entré sur la séance sont perdues dans ce cas). S’il a assez de crédits on l’inscrit, on notifie le coach s’il a été demandé, on prélève les crédits adhérent et on lui confirme son inscription. On notifie également le coach de la séance qu’un participant supplémentaire est prévu.

Ce diagramme fait référence au diagramme “demande de coach”, Cela nous permettait de fragmenter les tâches pour les expliciter plus facilement et de ne pas surcharger un diagramme déjà conséquent.

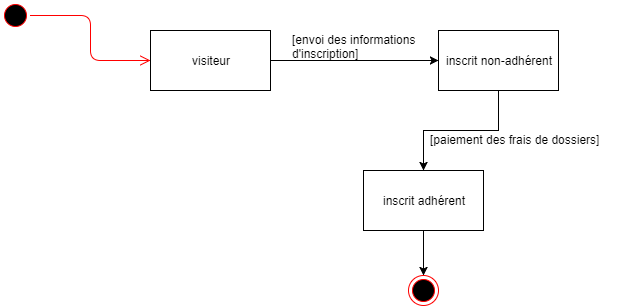
Si l’adhérent demande un coach mais que ce dernier n’accepte pas on prélève quand même les crédits du coaching à l’inscription. Si le coach n’a toujours pas accepté au moment de la séance, on rembourse l’adhérent des crédits du coaching. Nous nous sommes rendu compte de ce problème relativement tard et sommes parvenu à la conclusion qu’il s’agissait de la manière la plus simple de gérer ce problème dans le délai imparti, cela évite également les abus des adhérents (réservation à crédit de coachs).

### 

## Diagrammes d’état

### Adhésion

Ce diagramme montre les différentes étapes de l’inscription telle que nous l’avons conçue.

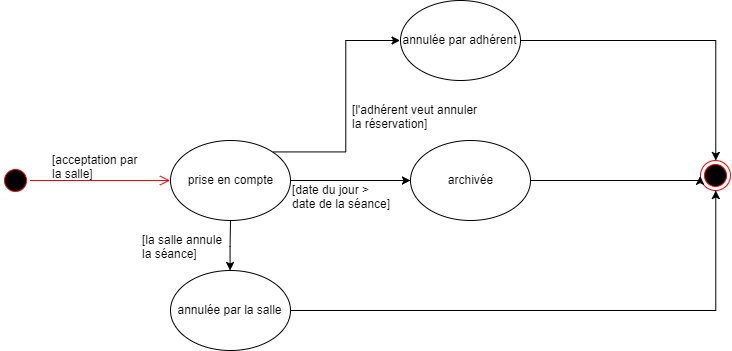
Commentaires:

Une personne qui est sur le site sans avoir rempli les informations d'inscription est un visiteur. Une fois que les informations relatives à l'inscription sont reçues, le visiteur devient un inscrit non adhérent. Ce n'est qu'une fois qu'il a payé les frais de dossiers qu'il devient un membre adhérent.

#### 

#### 

### Réservation



Commentaires:

On part du principe que lorsqu'un adhérent veut réserver pour une séance, on n'enregistre la   
réservation que dans le cas ou est accepté. Si la réservation est impossible à satisfaire, on ne l'enregistre pas en base de donnée.  
Une fois la réservation enregistrée en base, 3 cas de figures sont possibles: Soit la salle est contrainte d'annuler la séance associée, il y a répercussion sur la réservation qui est annulée par la salle. Si l'adhérent ne peut se rendre à la séance et prévient avant, alors la séance est annulée par l'adhérent. Sinon, on archive la séance comme effective si la date du jour est égale à la date de la séance.

## 

# Dossier d’administration

Nous avons décidé d’utiliser des fonctions pour gérer nos accès aux données, en évitant de donner directement des autorisations Select / Update / Insert / Delete sur les tables. Ceci n’est pas permis de base par PostgreSQL, car un utilisateur appelant une fonction qui insère des lignes sur une table donnée est sensé avoir le droit d’insertion sur cette table.  
Cependant, en précisant SECURITY DEFINER après avoir précisé le type de retour de la fonction, alors la personne exécutant la fonction exécutera cette dernière en héritant (uniquement pour la fonction) des droits de la personne qui l’a créé.

Nous avons opté pour cette idée car la plupart des droits que nous souhaitons accorder paraissaient sur dimensionnés par rapport à l’information dont il avait le droit de consulter. Par exemple, un adhérent doit avoir le droit d’update sur la colonne “souhaiterecevoirnotif” alors que nous souhaitons qu’il n’ait le droit que de modifier sa ligne.

Nous avons aussi pensé à utiliser des vues, et donner aux utilisateurs uniquement le droit sur les vues. Cependant il n’est pas possible d’appeler des paramètres pour les vues. Si je veux donner le droit à un utilisateur pour qu’il ai accès à une vue uniquement sur la ligne dont l’id correspond à son id, je suis obligé de passer par la fonction **current\_user**.

Cela oblige donc à créer un rôle sur postgre pour chaque utilisateur, ce qui nous paraissait disproportionné en ce qui concerne les adhérents (on peut imaginer se retrouver avec des milliers de rôle sur la base). De plus, chaque vue est une instanciation d’un objet en mémoire, ce qui à terme amènera à des baisses de performances, comparé à l’exécution d’une fonction.

Finalement, l’exécution d’une fonction nous permet de contrôler plus effectivement l’insertion des données, sans recourir à des triggers supplémentaires. Une fonction permet de vérifier la donnée, puis de l’insérer dans le même temps si tout va bien. Si nous passions par des insertions sur la vue, il faudrait dès fois utiliser des triggers (notamment pour les contraintes dynamiques.).

En utilisant des fonctions, nous pouvons retourner un booléen qui confirme si la fonction est arrivée à son terme (auquel cas elle renvoie True) ou non. Cela facilite le travaille du côté développement web, car on peut en php appeler une fonction et stocker directement la réponse, et ainsi pouvoir informer la personne ayant appelée la requête pour qu’elle sache si l’opération s’est bien déroulée.

## Répartition des fonctions par groupes

### Visiteur:

auth

inscription\_adherent

### Inscrit adhérent

accepter\_invitation\_seance

achat\_abonnement

achat\_carte

effectue\_inscription\_seance

effectue\_reservation\_seance

verif\_inscription\_seance

vertif\_reservation\_seance

**Inscrit non adhérent**

### paiement\_adherent

### Coach

consulter\_son\_planning

accepter\_coacher

### administrateur

ajouter\_compte\_coach

ajouter\_compte\_personnelaccueil

ajouter\_seance

annule\_seance

clore\_compte\_inactif

### personnel\_Accueil

effectue\_inscription\_seance

effectue\_reservation\_seance

verif\_inscription\_seance

vertif\_reservation\_seance

Pour les fonctions suivantes: Les personnels d’accueil ont ces fonctions dans le cadre ou l’adhérent vient à la salle de sport pour faire des opérations qu’il pourrait faire chez lui. Au lieu d’être identifié par le site, il est identifié par l’employé.

paiement\_adherent

inscription\_adherent

achat\_abonnement

achat\_carte

Exemple: Pour le rôle “coach”:

Create role coach;

Grant execute on function consulter\_son\_planning(idUsr int) to coach;

Grant execute on function accepter\_coacher(idUsr int, idSeance int) to coach;

Finalement, en ce qui concerne l’attribution des rôles et des logins:

Nous avons décidé que pour toutes les personnes internes à l’entreprise avaient un rôle dédié dans la base de donnée. A des fins d’archivage, il est plus simple de savoir quel rôle dans la base est en train de faire quelque chose. On imagine qu’il n’y aura pas plus de quelques dizaines d’employés, ce qui reste à notre sens suffisamment restreint.

Nous créons donc un compte pour un coach donné, puis nous lui donnons les droits d’exécution des fonctions de coach.

Exemple: Pour le rôle “coach”

Create role coach\_1 with password 'a' connection limit 1;

Grant coach to coach\_1;

Pour les utilisateurs du site externes (comme les adhérents non inscrits) nous avons un seul rôle qui peut accepter un grand nombre de connexion.

Exemple: Pour le rôle “inscrit\_non\_adhérent”:

Create role inscrit\_non\_adherent CONNECTION LIMIT 100 PASSWORD 'a';

Grant execute on function paiement\_adherent(idUsr int) to inscrit\_non\_adherent

## Droits donnés à certains profils

Nous avons tout de même jugé nécessaire de donner des droits directement à certains profils. Nous avons crée un rôle “information\_de\_base” qui correspond à ce que n’importe qui peut consulter (les informations tarifaires, sur les séances …) et qui est octroyé à tous les rôles.

Nous avons également décidé de donner les droits d’insert update delete à l’administrateur, au cas où il y a des informations qu’il faut corriger à la main.

# Partage d'informations avec la salle partenaire:

Le but de cette partie est de permettre, comme indiqué dans le sujet de permettre un partage d'informations avec une salle partenaire. Ce partage d'informations se fera dans les deux sens. Il nous faut donc cerner les données accessibles par la salle partenaire et les données dont nous auront besoin pour proposer à nos adhérents une solution alternative s'il n'est pas possible d'accéder à la séance de leur choix dans notre salle.

Durant ce projet nous avons travaillé conjointement avec le groupe 8 dont les membres sont:

* FERRY Emeline
* KELLER Rowane
* GHARNATI Souheib
* SPEICH Jules

## Envoi des données

Dans cette partie nous verrons quelles données sont nécessaires à la solution de notre partenaire pour fonctionner et une solution algorithmique pour récupérer ces données dans notre base et les envoyer à notre partenaire. Nous verrons ci-dessous les formes pseudo-algorithmiques des fonctions d’envoi et de réception de données que nous employons. Malheureusement nous n’avons pas pu implémenter ces fonctions dans l’application par manque de temps.

On a tout d'abord besoin d'une fonction qui montre les séances disponibles:

Début :

-Tant que le partenariat est en cours

- Chercher dans la BDD les séances non pleines qui sont postérieures à la date courante (pas encore passées)

- De ces séances on récupère :

- le nom

- la description

- la date

- l'horaire

- Si elle est coachée

- s'il faut réserver

- Quand un utilisateur sélectionne une séance d'un partenaire

- On récupère :

- ses informations

- l'ID de la séance qu'il a choisie

- Envoi des données aux partenaires

- Fin Tant que

Fin

## Réception des données

Pour que les utilisateurs des salles partenaires puissent accéder au site/séances il leur faut un compte spécial invité. Il s'agit d'un compte temporaire valide seulement pour la séance donnée et qui a seulement le droit de s'inscrire à une séance, elle contient:

- ID

- IDProvenance //id de la personne chez le partenaire

- SALLE

- Nom

- Prénom

- mail

PK = IDProvenance, SALLE

On enregistre quand un client d'une autre salle veut s'inscrire dans une séance de chez nous:

Début

- On récupère les données de l'adhérent de la base partenaire

- On récupère l'ID de la séance à laquelle il veut s'inscrire

- S'il n'existe pas de compte invité correspondant:

o On crée le compte invité

o On génère ID

o On lie la salle

o On récupère les autres arguments de la salle partenaire et on remplit les colonnes

IDProvenance

Nom

Prénom

Mail

- Sinon

o On retrouve le compte invité lié

- On vérifie qu'il reste des places disponibles dans la séance

- On inscrit l'invité (on décrémente le nombre de places disponibles dans la séance)

Fin

L'inscription d'un invité est un peu spéciale par rapport à celle d'un adhérent de la base car elle ne se fait pas avec la relation s'inscrire mais simplement en décrémentant le nombre de places disponibles dans la séance.

Informations supplémentaires:

- Les vérifications de validité des infos rentrées dans invite sont faites par la salle partenaire

Il nous faut donc une table invité à part:

create table INVITE (

IDINVITE SERIAL,

IDPROVENANCE INT4 not null,

IDSALLE INT4 not null,

NOM TEXT null,

PRENOM TEXT null,

MAIL TEXT null,

constraint PK\_INVITE primary key (IDPROVENANCE, IDSALLE)

);

create unique index INVITE\_PK on INVITE (

IDPROVENANCE,

IDSALLE

);

create index INVITE\_FK on INVITE (

IDSALLE

);

## Intégrité et sécurité des données:

Le fait d’employer des fonctions pour transmettre les données assure que seules les données que nous désirons communiquer sortent de la base et aucune autre. Les personnes de l’autre salle n’ont pas d’accès direct à la base de données et n’ont donc pas la possibilité de consulter ou d’altérer des données qui ne leur sont pas destinées.

Cela garantit la sécurité des données de la base.

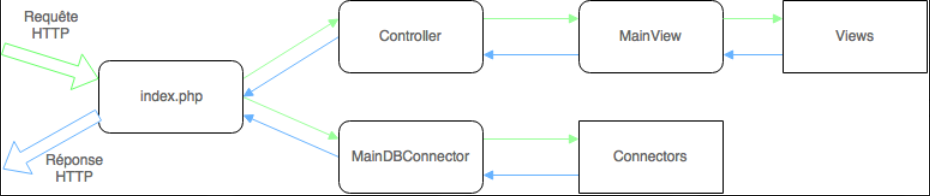
De plus les informations des personnes extérieures à la base sont stockées dans une table Invité bien distincte des autres adhérents dont l’utilisation est temporaire. Il n’y a donc pas d’insertion malicieuse de faux comptes d’adhérent (ou doublons) possibles.

Cela garantit l’intégrité des données de la base.

# 

# 

# L’application



Commentaires:

Toutes les requêtes HTTP envoyées au serveur sont redirigées vers index.php par le module de redirection géré par le module url\_rewrite d’apache. Celles-ci sont ensuite traitées par le framework Slim 3, qui va autoriser uniquement les requêtes spécifiées sur les URI traitées.

On ne peut donc pas accéder aux fichiers du serveur, et on ne peut utiliser que les requêtes HTTP autorisées, sur les URI associées. Ce qui garantit la sécurité des données sur le serveur, et centralise la gestion des interactions avec l’utilisateur.

De plus, la plateforme suit une architecture rigoureuse et modulable, lui permettant d’évoluer facilement. En effet il suffit de modifier des constantes globales pour ajouter de nouvelles routes utilisables par la solution.

Les différentes vues, spécifiques à chaque page (Connect.php gère le contenu à afficher lors de la connexion, etc) héritent toutes de la classe View. Cette dernière contient des fonctions retournant le code HTML d’éléments souvent utilisés sur la plateforme. Cela permet d’éviter les redondances de code, et de centraliser les traitements communs. Le code est donc plus facile à maintenir et à faire évoluer.

Cependant cette architecture à été difficile à mettre en place, et le temps à fini par manquer pour créer toutes les pages de la plateforme. Cependant, avec les méthodes utilisées pour la construction du site, et celles qui permettent d'interagir avec la base de données, on peut facilement imaginer que la finalisation de la plateforme ne nous prendrait pas très longtemps. Étant donné que tous les outils nécessaires sont désormais en place.

# 

# 

# Conclusion

Ce projet fut à la fois un très bon entrainement aux épreuves futures mais aussi une importante mise en application des compétences que nous avons acquis tout au long de cette formation. De plus, ce projet nous a permis de renforcer la cohésion du groupe et d’opérer une montée en compétence globale de chacun dans les domaines que nous maîtrisions le moins.

Au delà du développement personnel et du groupe, ce projet nous a permis d’appréhender la création d’une application de type “réel” dans un cadre très concret en étant toutefois toujours guidé par les conseils avisés de nos professeurs. Lier la conception et le développement dans un même projet de manière si poussée nous a poussé à adopter un état d’esprit et une méthode adéquate.

Malgré quelques problèmes de temps au niveau du développement, le projet est arrivé à un niveau de rendu assez complet. Les différentes étapes nous ont souvent rappelé l’importance d’une bonne conception aux diagrammes fournis qui permet la bonne compréhension de l’ensemble du projet.

L’application actuelle, bien que pas encore maturée, permet d’assurer les fonctions de base demandées par le sujet. Cependant, cela nous montre que l’amélioration est toujours possible.

Ce projet était donc un enrichissement à tout point de vue.