Projet ACOL/CAWEB

Gestion des TAP

24 avril 2018

Cyril Carlin et Quentin Démarque

Table des matières

1 Analyse

1.1 Diagrammes de cas d'utilisations

1.1.1 En partie

Ces diagrammes représentent les cas d'utilisation possibles de l'application, lorsque l'utilisateur est un administrateur (Mairie) ou un utilisateur lambda (parent).

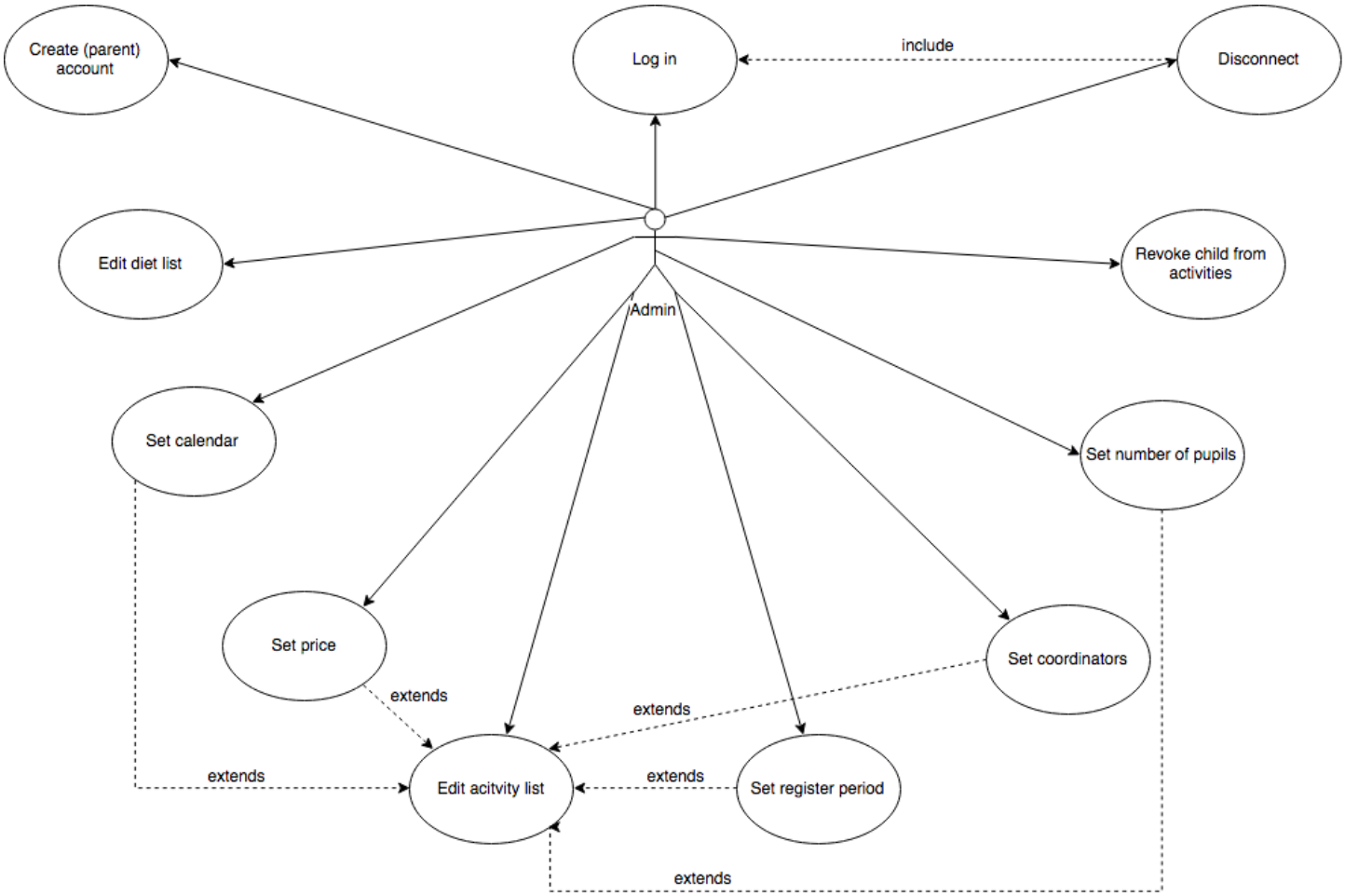

FIGURE 1 – Admin use case diagram

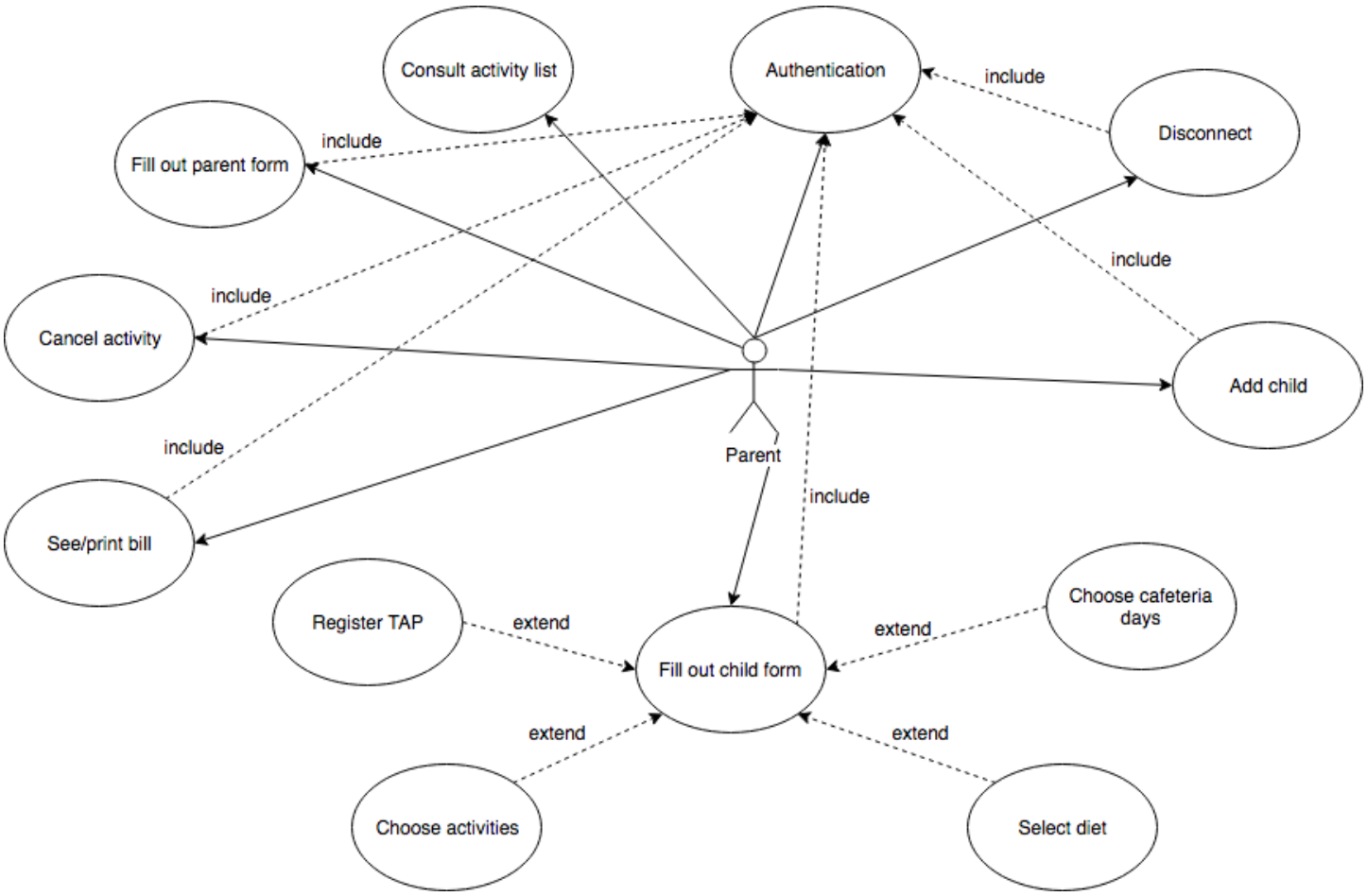

FIGURE 2 – Parent use case diagram

1.2 Diagrammes de classes d'analyse

Ce diagramme représente l'analyse des différentes classes nécessaires à la modélisation de l'application.

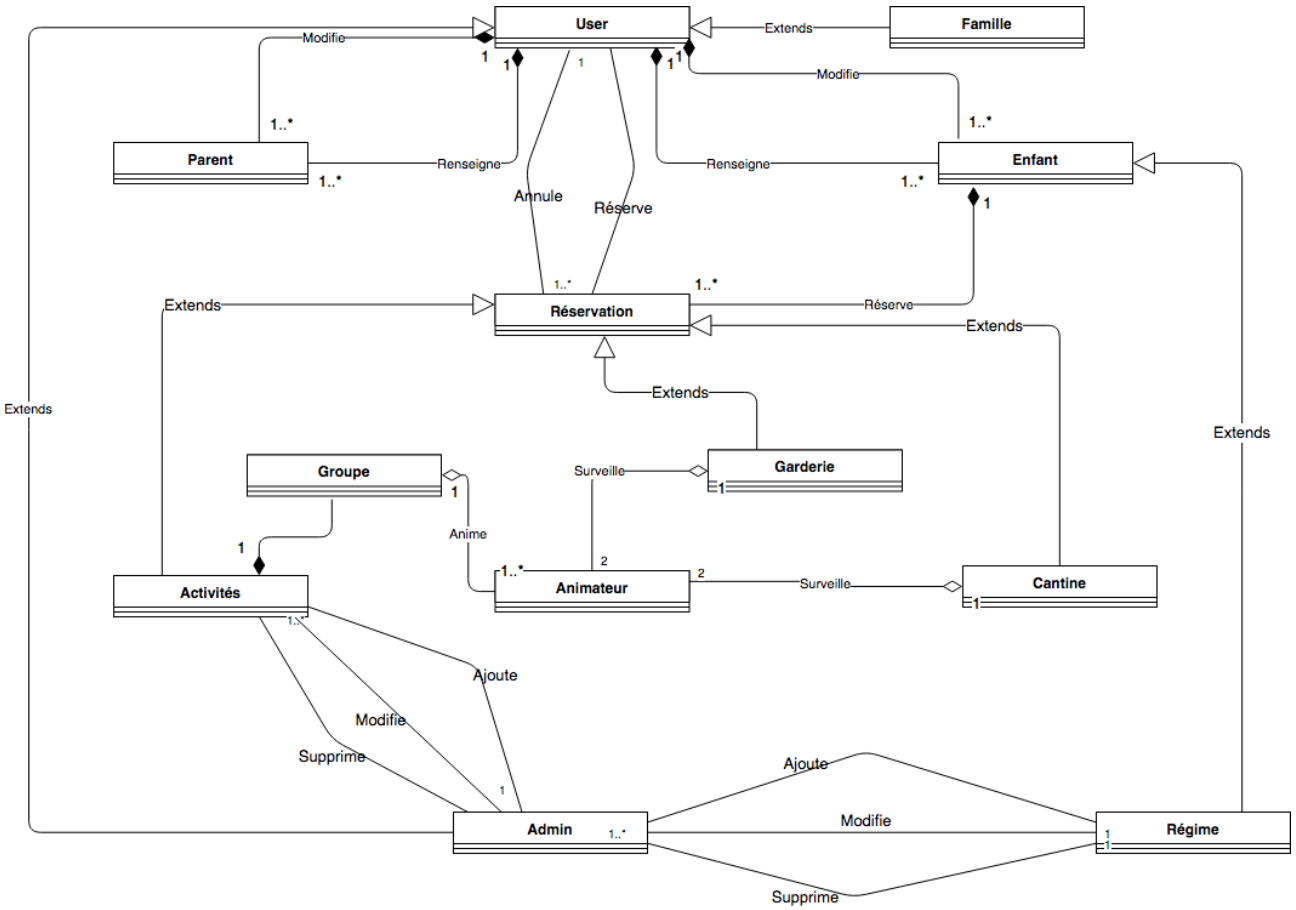

FIGURE 3 – Class analysis diagram

Note : à cette étape du projet, je pensais pouvoir implémenter la fonctionnalité d'annulation. Elle sera enlevée plus tard (cf Fonctionnalités non implémentées)

1.3 Diagrammes de séquence système

1.3.1 Login system

Ce diagramme représente la séquence de communications lorsque l'utilisateur se connecte sur le site.

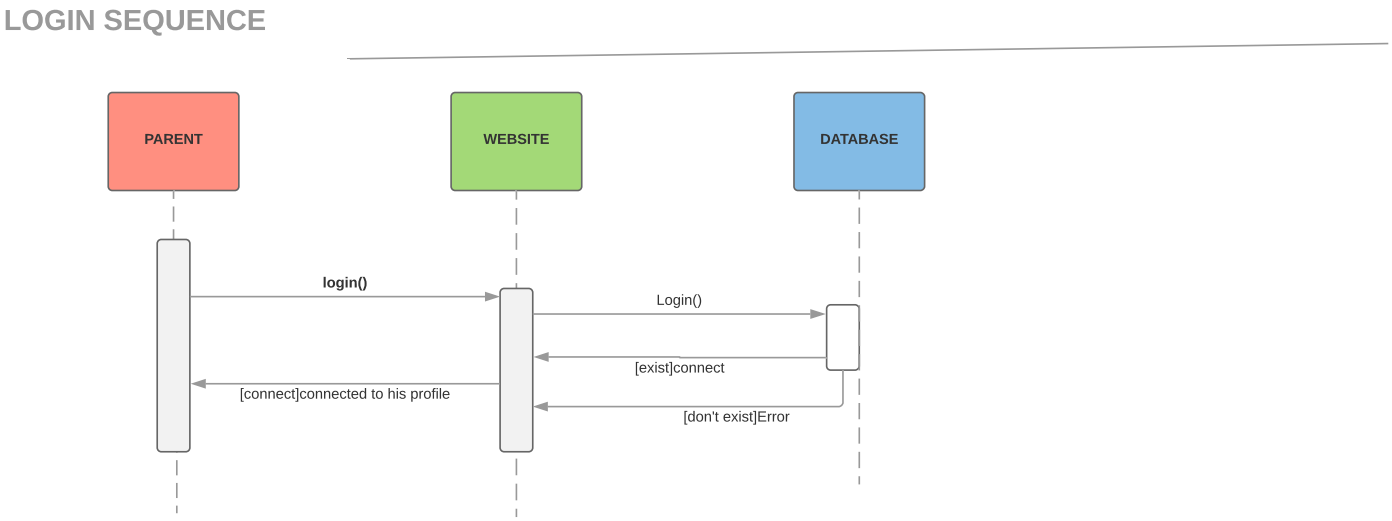

FIGURE 4 – Login sequence diagram

1.3.2 Form editing system

Ces diagramme de séquence représente le renseignement, la modification et pour certains cas la suppression ou l'annulation d'objets à travers les formulaires implémentés dans l'appliction.

ADMIN

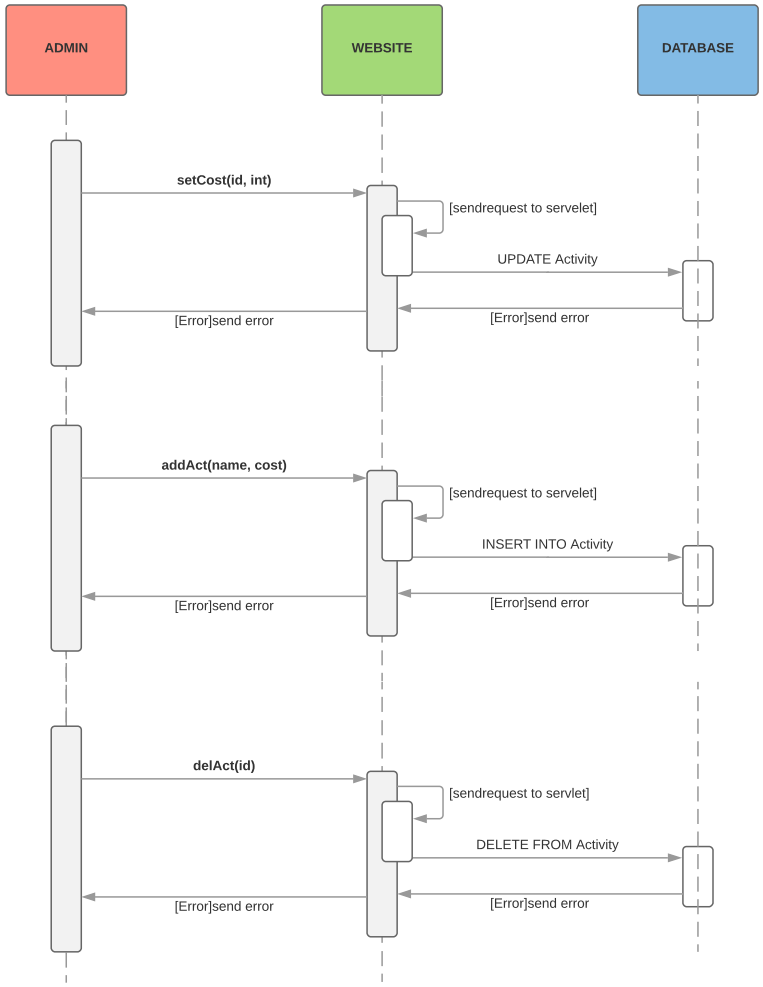

FIGURE 5 – Admin editing system

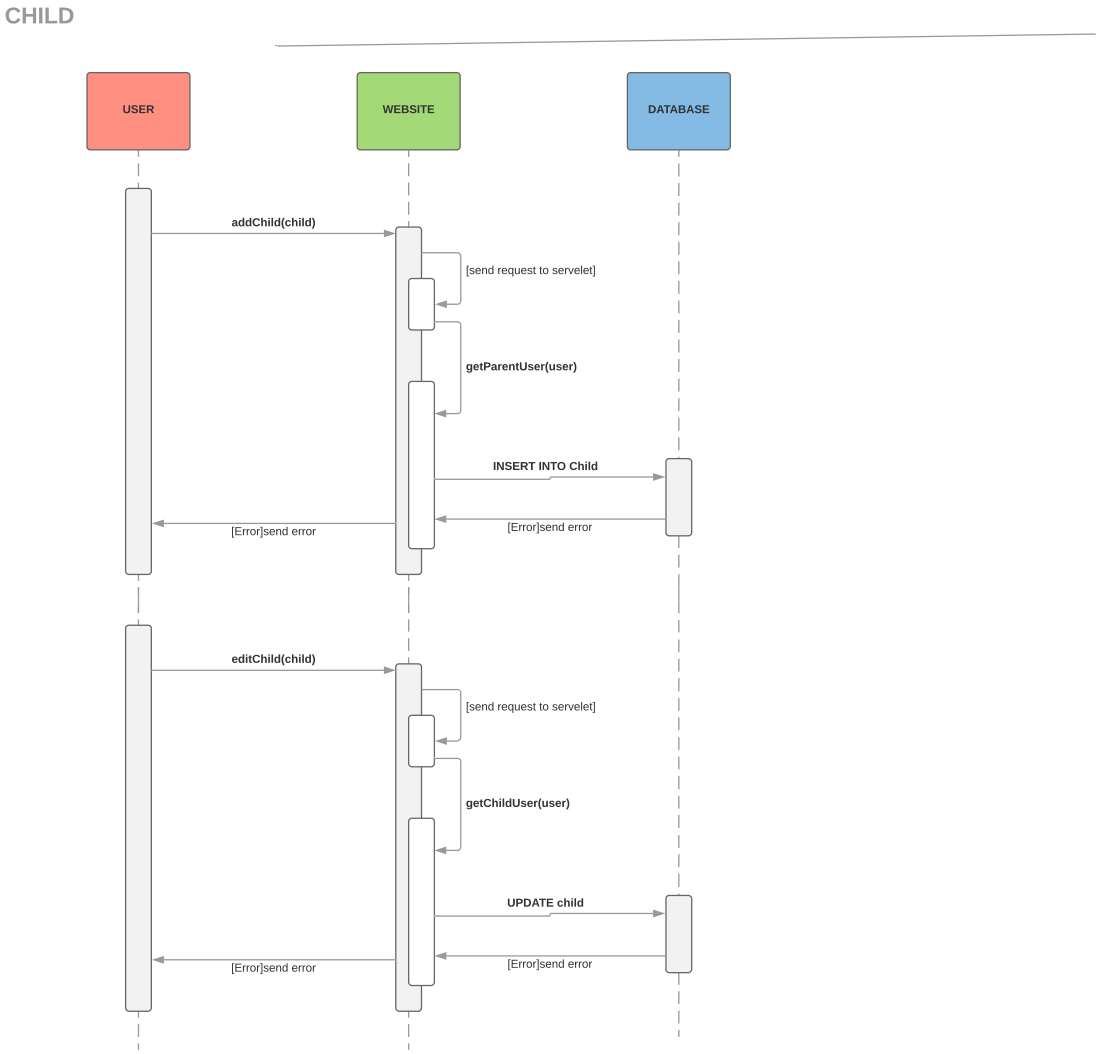

FIGURE 6 – Child editing system

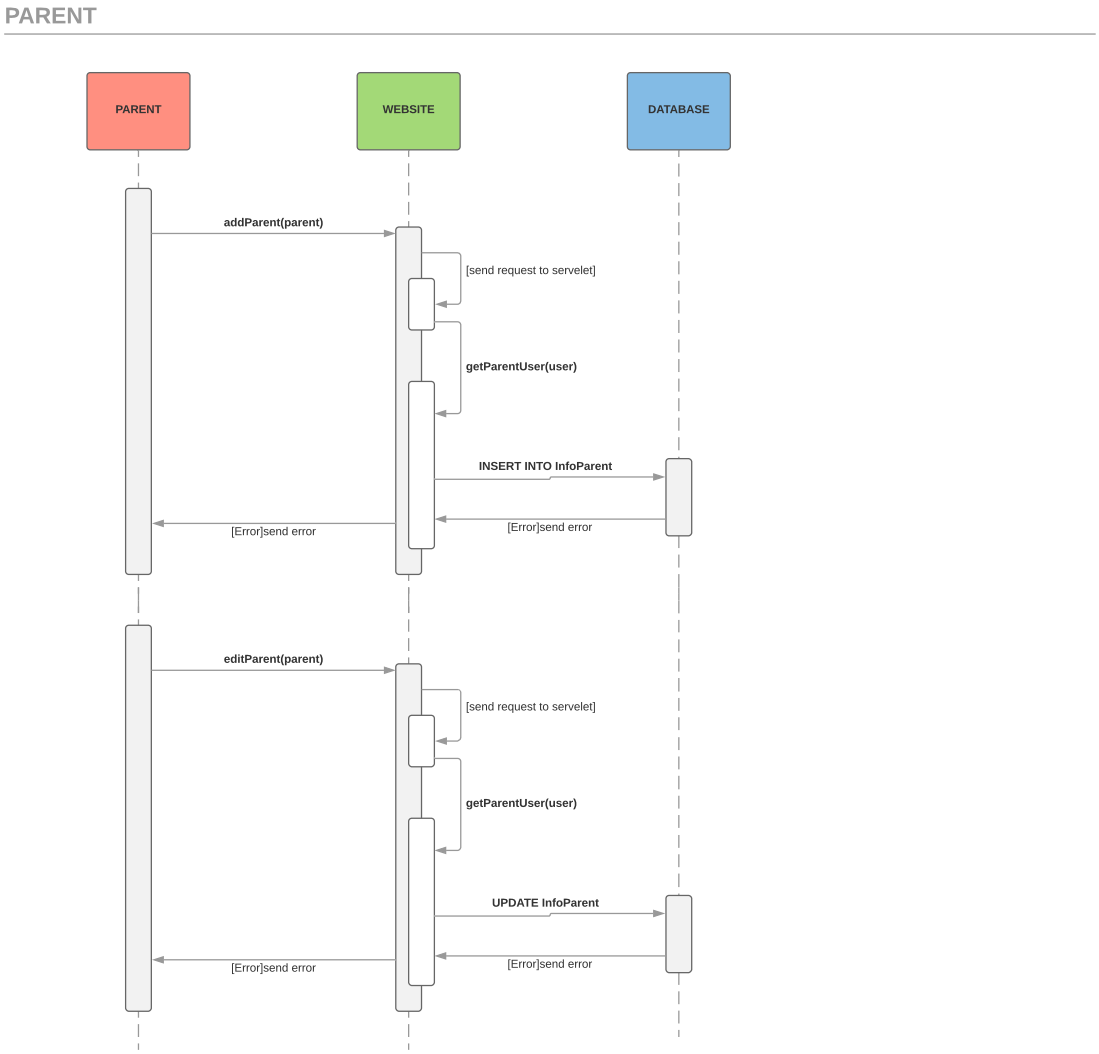

FIGURE 7 – Parent editing system

1.4 Diagrammes d'activité

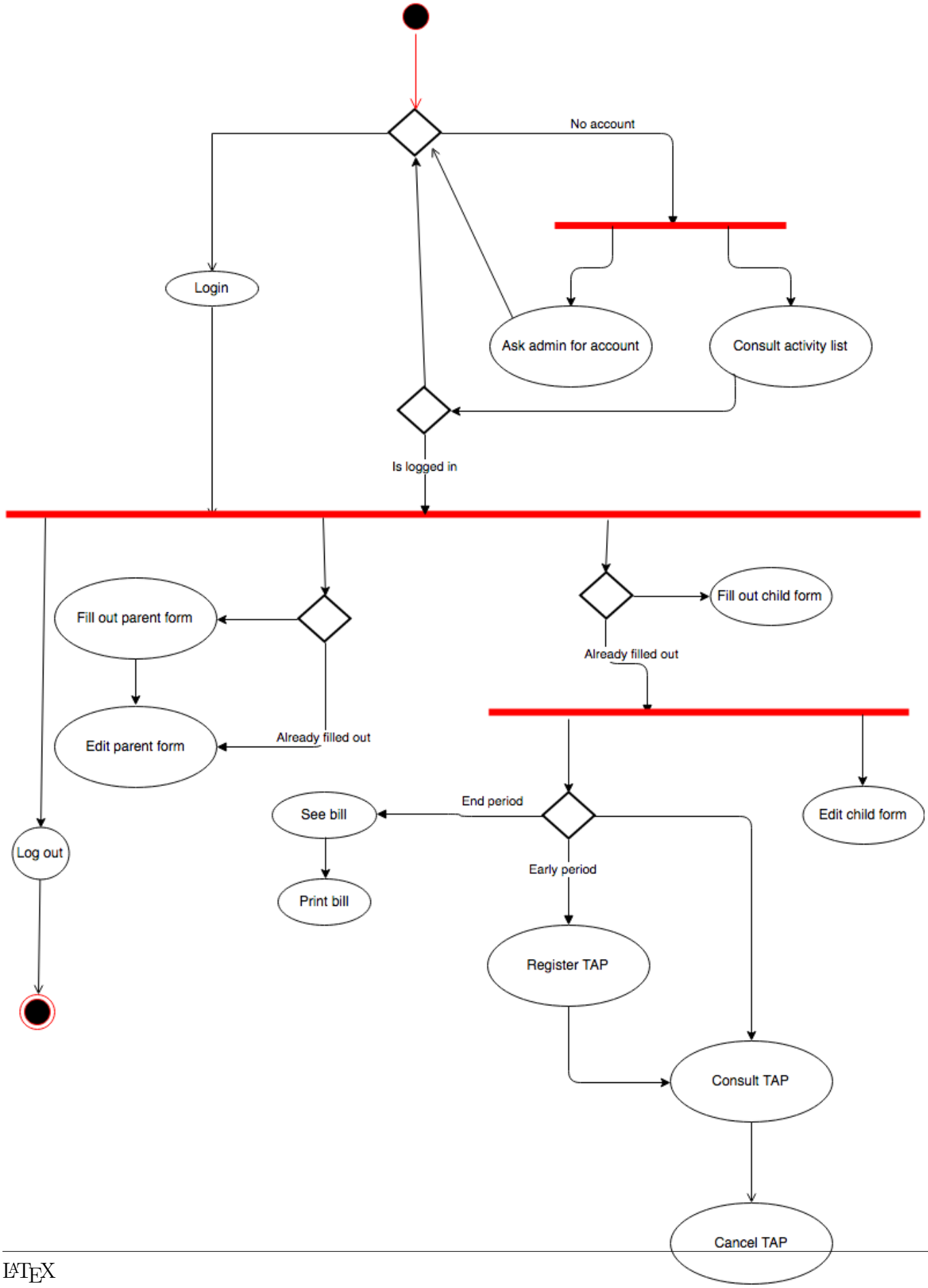

FIGURE 8 – User activity diagram

2 Conception

2.1 Architecture MVC

La figure ci-dessous montre comment l'architecture MVC est implémentée dans notre projet. Tous les liens entre les classes ne sont pas représentés par souci de lisibilité.

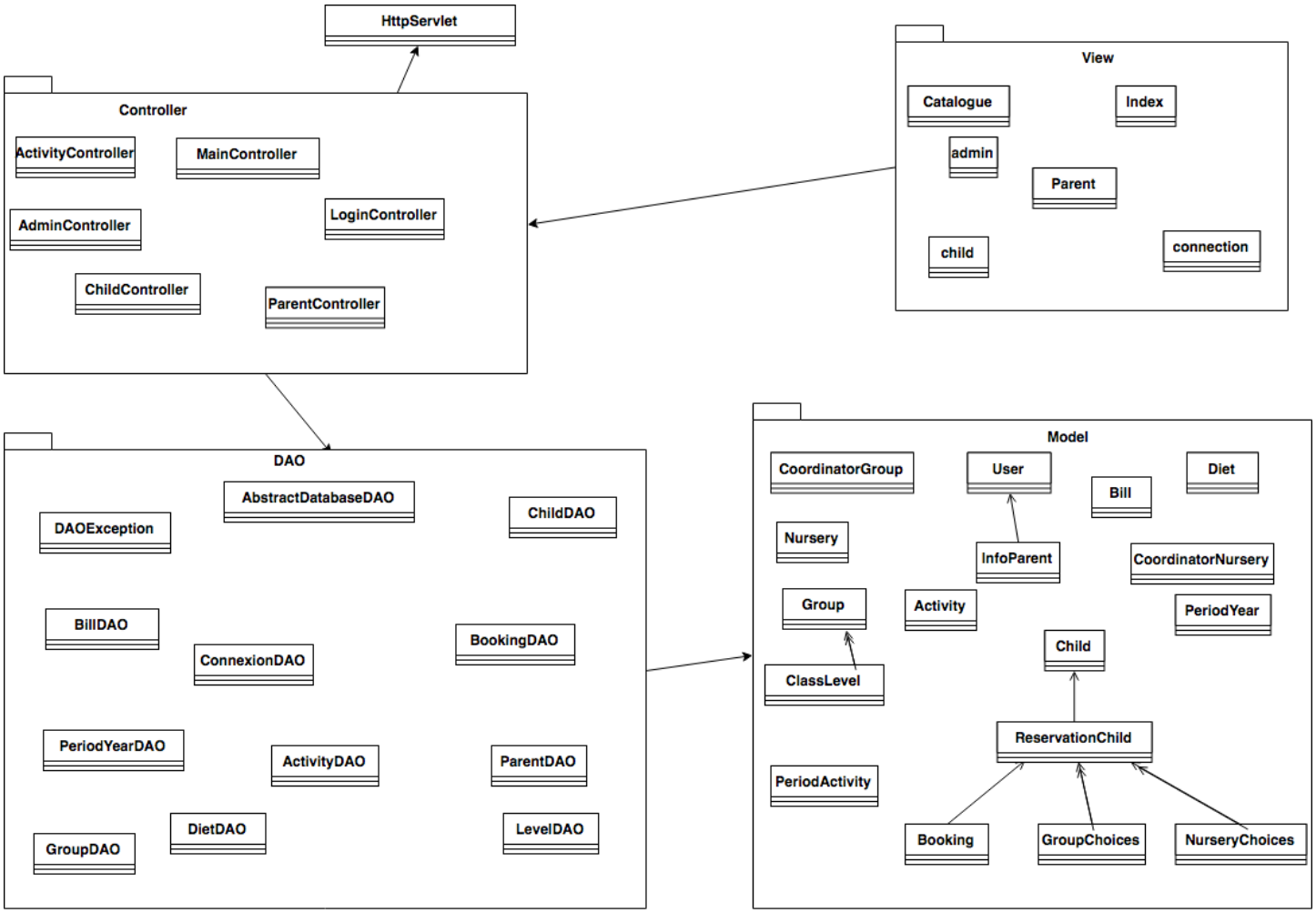

FIGURE 9 – MVC Conception

2.2 Conception de la base de données

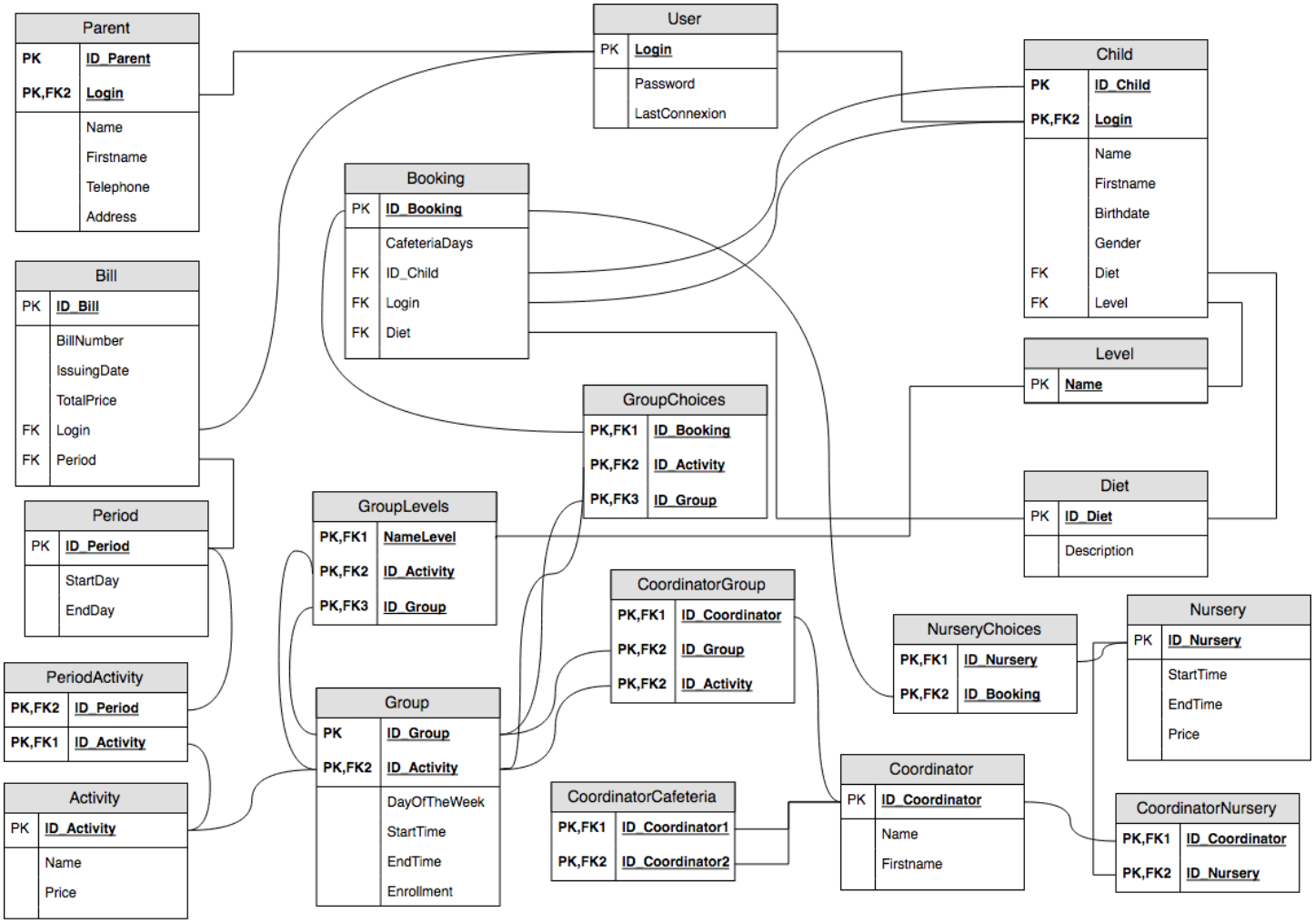

FIGURE 10 – Database conception diagram

Note : Initialement, une table était présente pour pouvoir gérer les absences, mais comme finalement la fonctionnalité n'a pas été gérée et par souci de lisibilité, celle-ci n'apparaît pas sur le schéma ci-dessus.

3 Manuel utilisateur

3.1 Hors connexion

Quand l'utilisateur n'est pas connecté, il peut, comme indiqué dans le cahier des charges, consulté la liste des activités proposées par la mairie. Cette liste est exhaustive et contient également les activités non prévu pour la période en cours. J'ai pensé qu'il était logique que ces activités devaient apparaître quelque part et cette page me semblait la plus adaptée. Les activités sont présentées sous forme d'un tableau où chaque ligne est cliquable et permet de faire appraître les groupes associées à l'activité cliquée. Un bouton "Réserver" en bout de ligne permet de redirigier l'utilisateur sur :

1. La page de connexion si l'utilisateur n'est pas connecté.
2. La page ENFANTS sinon

En guise d'extension, un formulaire de recherche a été ajouté côté client (et non côté serveur) afin de rechercher un groupe d'activités selon des filtres proposées. Ce filtre est accessible depuis la page ACCUEIL et la page ACTIVITES.
Enfin, le bouton "Se connecter" permet d'afficher la page de connexion où l'utilisateur peut rentrer ses identifiants pour accéder à son compte. En guise d'extension, un lien "Identifiants oubliés" a été ajouté côté client uniquement afin de pouvoir fournir une solution à l'utilisateur en cas d'oubli d'identifiants.

3.2 Utilisateur parent connecté

Une fois connecté, de nouvelles pages sont accessibles à l'utilisateur parent.
En effet, l'utilisateur peut désormais accéder aux fiches parents associées à son compte depuis l'onglet PARENTS en haut-de-page. Il peut en ajouter (dans le cas d'une première connexion au compte, l'utilisateur parent doit rentrer ses coordonnées), et également modifier celles déjà remplies.
Aussi, les fiches enfants sont accessibles depuis l'en-tête depuis l'onglet ENFANTS. Ici, l'utilisateur parent peut ajouter une fiche enfant contenant les informations de base requises décrites dans le cahier des charges. Ensuite, il peut consulter les fiches existantes et les modifier si nécessaire. Une fois la fiche créée, l'onglet "Inscriptions" vient s'ajouter à l'onglet "Informations". Cet onglet contient le formulaire d'inscription aux TAP proposées par la mairie.
Si nous sommes en début de période, un bouton "Réserver" apparaîtra en bas du formulaire et permet à l'utilisateur de valider ses choix d'inscription. Pour les activités, un tableau est présenté et chaque ligne est cliquable pour pouvoir présenter les groupes associés à l'activité cliquée. En bout de ligne des groupes, un sélecteur nommé "Choix" permet à l'utilisateur de classer ses 3 vœux.
Si ce n'est pas le début de période, le bouton n'apparaît pas et l'utilisateur ne peut que consulter les choix qu'il a fait pour cette inscription et les activités dans lequel l'enfant a été accepté. Le sélecteur de choix sur la ligne des groupes ne comporte à présent qu'une seule option : "- Accepté(e) -" et seuls les groupes auxquels l'enfant a été accepté sont présentés.
Enfin, en fin de période, un nouvel onglet est accessible à l'utilisateur : "Facture". Cet onglet permet de faire apparaître la facture associée à l'enfant et à la période qui se termine. Un bouton d'impression est également présent et permet à l'utilisateur d'imprimer la facture en question.

3.3 Utilisateur admin connecté

Un administrateur a accès un onglet supplémentaire "ADMIN" qui lui permet d'ajouter, de modifier et de supprimer des actitivités, des groupes et des régimes.

4 Bilan

4.1 Choix d'implémentation

4.1.1 Gestion de l'affichage

Afin de répondre au maximum aux attentes du cahier des charges, j'ai longtemps pensé l'architecture des pages et notamment des formulaires afin d'y faire apparaître les informations requises sans pour autant réduire la lisibilité. C'est pourquoi j'ai ajouté des menu déroulant, des onglets en accordéons, des sous-onglets, etc. L'utilisateur peut ainsi choisir d'afficher les informations auxquelles il veut accéder et seulement celles-ci.
De même, le choix du thème est censé être neutre, sobre et mettre en valeur les informations importantes, et les liens cliquables. C'est pourquoi un grand nombre d'onglet, ou de bouton change (légèrement parfois) de couleur quand la souris passe dessus.
Le but de se projet était pour moi de produire un projet fini, net et professionnel afin de me perfectionner à la fois à l'implémentation MVC mais sans délaisser le design de l'interface qui, selon moi, est tout aussi important car il permet à l'utilisateur de comprendre instinctivement le fonctionnement du site.

4.1.2 Gestion des périodes

Par manque de temps, la gestion des périodes est géré dans le servlet ChildController qui, en fonction de la date du jour, affiche ou pas le bouton "Réserver", affiche ou pas les factures. Par conséquent, les périodes sont gérées seulement si un accès au servlet ChildController est effectué. Les variables (DAYS_BOOKING et DAYS_BILL) privées et finales ont été ajoutées à ce servlet et déterminent le nombre de jours du "Début de période" et "Fin de période".
Une meilleur gestion du temps aurait pu être fait à travers un thread créé au démarrage du serveur. Ce thread pourrait alors, indépendamment des servlets, décider si telle ou telle fonctionnalité est accessible au jour donné.

4.2 Gestion des vœux

La gestion des vœux se fait au "premier arrivé, premier servi". C'est pourquoi une fois la réservation effectuée, l'utilisateur peut directement consulter son inscription et les activités auxquelles son enfant a été accepté.
J'aurais voulu faire quelque chose de plus optimiser, notamment attendre la fin de la période afin d'avoir les choix hiérarchisés de tout le monde. Le choix de l'utilisateur i au groupe j peut être modélisé par la variable $x_{i,j}$ qui vaut : 1 si accepté, 0 sinon. En pondérant ces variables par l'importance du vœux $a_{i,j}$ (exemple : 3 pour le vœux 1, 2 vœux 2, 1 vœux 3, et 0 si non choisi), on peut réduire l'affectation des enfants aux groupes à une équation de Bellman qui consiste à maximiser la somme :

$$\sum_{i>0}\sum_{j>0} a_{i,j}x_{i,j}$$

4.2.1 Automatisation des requêtes : *PL/SQL*

Un bon nombre de fonctions dans les fichiers DAO tel que *createBill* dans *BillDAO* effectuent plusieurs accès à la base de données pour pouvoir remplir leur fonction. En fonction de la distance séparant le serveur et la BDD, ces accès peuvent coûter cher. C'est pourquoi il aurait été préférable de faire appel à des fonctions/procédures PL/SQL directement implémentées en BDD afin de limiter les appels et ainsi le temps d'exécution de ces fonctions.

4.3 Interprétations du sujet

Nous avons fait certain choix d'interprétations au cours du développement du projet :

1. Les factures ne sont pas des entités faibles d'un utilisateur. La mairie pourra ainsi toujours consulter les factures émises, même si l'utilisateur a été supprimé.
2. La fiche enfant et la fiche d'inscription aux TAP ont été différenciées. Si ce n'était pas le cas, le renseignement d'une fiche enfant ne pourrait être fait qu'en début de période ce qui rendrait compliqué l'inscription d'un enfant en milieu d'année. Aussi, le "début de période" étant plus moins large, on peut laisser le temps aux parents de s'organiser pour choisir les activités auxquels ils désirent inscrire leurs enfants.
3. Un animateur est une entité propre car il peut être affecté à plusieurs TAP. Cela évite ainsi à devoir renseigné ses informations à chaque fois.
4. Par souci de légalité, je n'ai pas limité le nombre d'animateur par groupe à 1 car la législation impose un nombre d'animateurs minimum par nombre d'enfants (excepté pour la cantine).

4.4 Outils

J'ai utilisé :
— `draw.io` comme modeleur UML pour réaliser les diagrammes d'analyse et de conception.
— `Trello` pour organiser le projet et garder en vue les objectifs à réaliser.
— `Git` comme gestionnaire de version.

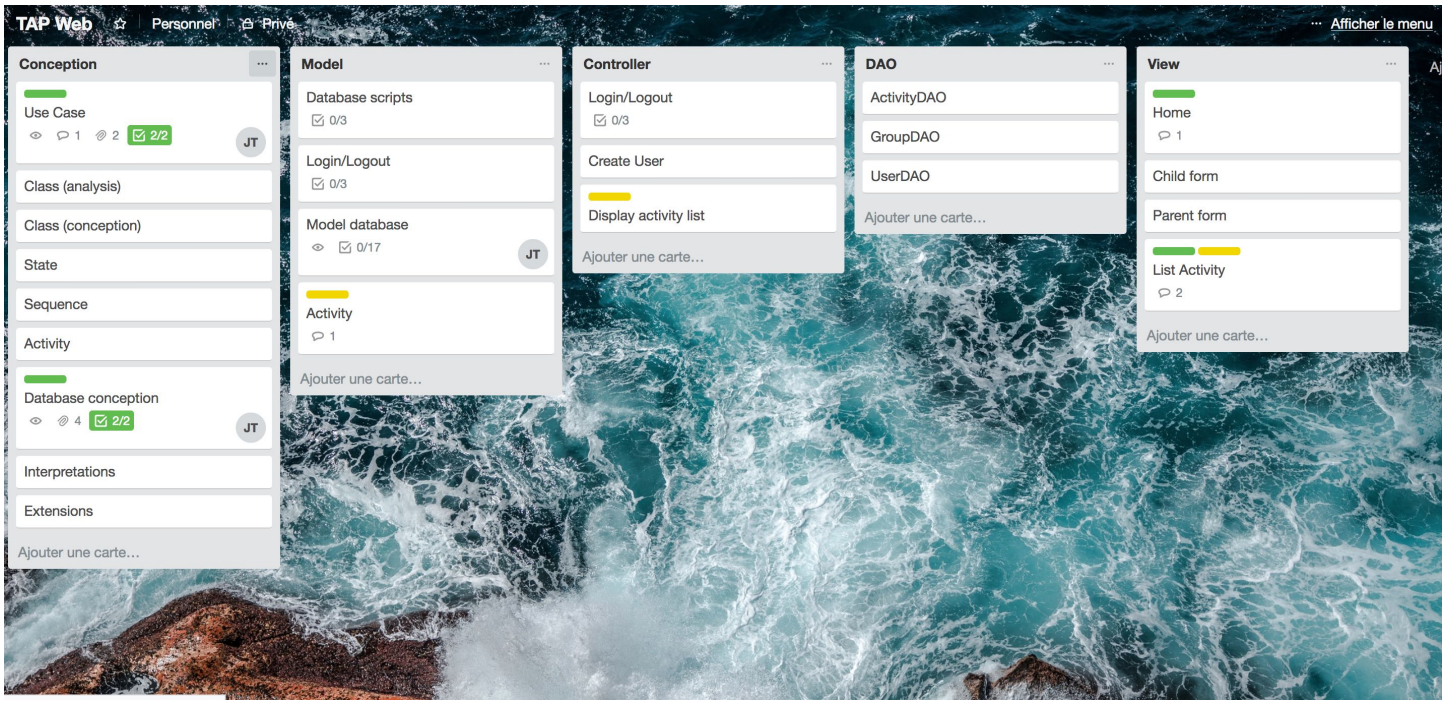

FIGURE 11 – Screenshot of our Trello organisation

4.5 Contenu du rendu

La racine du rendu contient la liste de dossier suivante :
— `conception` qui contient la liste des diagrammes réalisés au début du projet.
— `sql` qui contient :

  - `bd_install.sql` pour créer : les tables et les remplir.
  - `bd_drop.sql` pour détruire la base de données.
  - `bd_fill.sql` pour remplir une base de données avec des tables vides .
- `src` qui contient le projet maven contenant :
  1. Les sources côté serveur (java) : *src/main/java*
  2. Les sources côté client (html, jsp et javascript) : *src/main/webapp*

4.6 Bugs/Fonctionnalités non implémentées

Un grand nombre de bugs sont susceptibles d'être rencontrés sur la page ADMIN.
Fonctionnalités non implémentées :

1. Gestion des absences
2. Gestion de la hiérarchisation des vœux
3. Affichage des animateurs
4. Gestion des animateurs

4.7 Instructions de déploiement

4.7.1 Serveur Apache Tomcat

Il faut également ajouter un rôle dans le fichier *apache-tomcat/conf/tomcat-users.xml* afin que Tomcat puisse se connecter à la base de données avec l'identifiant et le mot de passe souhaité :

```
<role rolename="manager-gui"/>
<role rolename="manager-script"/>
<user username="CyrilCarlin" password="apacheEnsi"
        roles="manager-gui, manager-script"/>
</tomcat-users>
```

4.8 Base de données

Les identifiants pour l'accès à la base de données se fait dans : $src/TAP_Web/src/main/webapp/META-INF/context.xml$.
La variable *url* contient l'adresse du serveur de la base de données. *password* et *username* sont explicites.

```
<Resource
        auth="Container"
        type="javax.sql.DataSource"
        driverClassName="oracle.jdbc.OracleDriver"
        url="jdbc:oracle:thin:@ensioracle1.imag.fr:1521:ensioracle1"
        maxTotal="5"
        maxIdle="1"
        maxWaitMillis="20000"
        name="jdbc/tap"
        password="carlinc"
        username="carlinc"
    />
```

4.8.1 Déployer le site web

Une fois la configuration terminée, il faut lancer le serveur Tomcat avec *apache-tomcat/bin/startup.sh*, puis utiliser **Maven** pour déployer le site web sur le serveur :

`mvn tomcat:deploy` ou `mvn tomcat:redeploy`

Attention, tomcat nécessite d'avoir renseigné ses identifiants dans le fichier `~/.m2/settings.xml`. Le site web est accessible avec l'url suivante : *http ://localhost :8080/werewolves/*.