



Création d'un site internet proposant des prestataires pour de la garde d'enfants

R. BONNET - K. GIULIETTI - H. GRUNIG - V. GUERLAIS - T. L'YVONNET - N.SADOUNI

12 décembre 2016

Table des matières

1	Introduction	3
1.1	Thématique et problématique	3
1.2	Organisation du projet et du rapport	3
2	Spécifications	4
2.1	Inscription	4
2.2	Authentification	6
2.3	Edition de profil	7
2.4	Gestion de l'emploi du temps	9
2.5	Visite du site	12
2.6	Consultation du profil personnel	13
2.7	Demande de prestataire	14
2.8	Acceptation du contrat	15
2.9	Paieement du prestataire	17
2.10	Annulation	19
2.11	Évaluation prestataire/famille	21
2.12	Signalement d'un commentaire inapproprié	23
2.13	Modification d'un commentaire inapproprié	24
2.14	Validation d'un profil	25
2.15	Gestion des divers taux par les Admins	26
2.16	Création d'un compte Admin	27
3	Analyse	29
3.1	Diagrammes de plus haut niveau	29
3.2	Diagrammes de classes métier	29
3.3	Modèle logique de la base de données	29
4	Conception	31
4.1	Diagramme de séquence : modération d'un commentaire	31
5	Réalisation	32
5.1	Diagramme d'activité pour les quatre types d'utilisateurs	32
5.2	Tests fonctionnels	36
6	Retrospective	36
6.1	Captures Esitting	37
A	Gantt chart	45
B	Annexe Code php : Édition du profil	46

1 Introduction

1.1 Thématique et problématique

Les nombreux parents qui n'ont pas la possibilité d'être à temps complet en charge de leurs enfants, à cause d'impératifs professionnels ou personnels, sont à la recherche de personnes motivées mais aussi flexibles pour garder leurs enfants. D'autre part une grande partie des étudiants sont à la recherche d'emplois à temps partiel leur permettant de poursuivre leurs études, d'obtenir une rémunération leur permettant de vivre leurs vies étudiantes dans les meilleures conditions ou même de préparer leurs entrées dans la vie active. Il y a aussi un besoin perceptible pour les deux partis d'être connecté au bon correspondant afin de répondre aux attentes de chacun.

L'offre et la demande existent en matière de babysitting, nous avons d'une part des étudiants désirant travailler en dehors de leurs heures d'études et d'autre part des familles soucieuses de trouver des personnes dynamiques et impliquées dans le travail qu'on leur propose. Le but de ce projet est donc la création d'une plateforme permettant de regrouper les deux partis et faciliter l'échange de services. Cette plateforme permettra d'une part aux parents de chercher dans une zone géographique donnée les annonces qui leur conviennent le mieux, et d'autre part aux étudiants de pouvoir déposer des candidatures qui seront analysées par les familles. La recherche des profils pourra donc se faire selon divers critères notamment la zone géographique mais également selon une grille tarifaire, les notes attribuées aux babysitter par les précédents clients ainsi que les remarques qu'ils ont obtenus suites à leurs diverses missions.

Outre la lecture simple et rapide des profils et disponibilités des prestataires (étudiants), le site internet doit aussi permettre aux prestataires une lecture complète des profils des familles (adresse, nombre d'enfants, avis concernant la famille, etc.) et de fixer leur propre tarif horaire. Le site gère ensuite plusieurs cas en appliquant pour chaque cas un taux spécifique. Ces cas sont : le nombre d'enfants, les gardiennages de nuit, les week-ends, et les jours fériés. Ainsi, pour ces différentes situations, le système modifiera le taux horaire renseigné par l'étudiant.

Enfin le site internet doit se placer comme intermédiaire entre les deux partis et ce, aux différents stades de la mise en contact pour rendre celui-ci le plus simple possible. Le site doit assurer le traitement administratif des profils des familles et des étudiants ainsi que le traitement financier (virement bancaire et annulations) des gardes effectuées. Les administrateurs du site doivent pouvoir superviser les différentes étapes de la mise en contact qui sont les plus importantes (vérification des profils, des paiements, confidentialité des données récoltées, etc...).

La visibilité des informations sur le site web peut dépendre de plusieurs facteurs comme par exemple : le statut de l'utilisateur (famille ou étudiant, connecté au site ou non connecté). Le site web pourra mettre en contact des utilisateurs de France exclusivement.

1.2 Organisation du projet et du rapport

- Nous commencerons l'analyse du projet par les différents cas d'utilisation. Pour chacun d'entre eux nous présenterons les spécifications détaillées associées à ces cas d'utilisation. Grâce à cette première partie nous espérons mettre à jour toutes les fonctionnalités du site ; des plus simples aux plus complexes.
- Nous présenterons par la suite notre projet dans son ensemble avec une approche analytique en utilisant notamment des diagrammes UML qui décriront le système et ses acteurs mais aussi certains aspects plus conceptuels comme par exemple la structure des tables de notre base de données. Des captures d'écran présenteront l'aspect final de notre site.
- Ensuite nous nous attacherons à présenter les différents problèmes et solutions que nous avons pu rencontrer et comment nous les avons résolus. Nous présenterons les forces et les faiblesses de l'interface que nous avons construite par le biais d'un tableau représentant les différents cas fonctionnels rencontrés par les différents utilisateurs.
- Enfin, nous tâcherons de porter un regard critique sur notre travail et sur le projet en général.
- Un exemple de code (php/html) sera également présenté en annexe.

2 Spécifications

Cas d'utilisation sous la forme de scénarios de Cockburn & détaillée

— 2.1 Inscription

Spécification Fonctionnelle

Acteurs : Tous utilisateurs non-inscrit. (pas de session active)

Pré-condition : L'utilisateur non-inscrit se situe sur la page d'accueil.

Scénario nominal :

1. L'utilisateur sélectionne "Inscription".
2. Le Système propose un choix entre une Inscription "Etudiante" ou "Famille".
 - (a) L'utilisateur choisit une inscription "Etudiante"
 - i. Le système affiche un formulaire spécifique aux étudiants
 - ii. Ce formulaire possède des champs dont certains annotés d'une étoile (obligatoires) : "E-mail*", Mot de passe*, Nom*, Prénom*, Age*, Adresse* (complète), N°tel*, photocopie CNI*, véhiculé ou non, niveau d'étude, description personnelle (passions, personnalités, etc.)
 - iii. L'étudiant renseigne une option pour accepter automatiquement les missions ou renseigner un délai de réponse (3h, 6h, 24h par exemple)
 - (b) L'utilisateur choisit une inscription "Famille"
 - i. Le système affiche un formulaire spécifique aux familles
 - ii. Ce formulaire possède des champs dont certains annotés d'une étoile (obligatoires) : "E-mail*", Mot de passe*, Nom*, Prénom*, Adresse* (complète), N°tel*, Nombre d'enfants, Age des enfants, Photos, description personnelle (à propos des enfants, caractères, préférences...)
3. L'utilisateur remplit le formulaire
 - (a) Le système vérifie l'exactitude des informations
 - i. Adresse e-mail existante dans la base de données → go to 3 avec message d'erreur
 - ii. L'adresse e-mail entrée correspond à une adresse mail → go to 3 avec message d'erreur
 - iii. Mot de passe retapé différent du mot de passe initial → go to 3 avec message d'erreur
 - (b) Le système enregistre l'utilisateur → go to P1
 - (c) L'utilisateur est redirigé sur la page d'accueil
4. L'utilisateur annule l'inscription → go to P2

Post condition :

— P1 : L'utilisateur est inscrit, un mail de confirmation est envoyé à l'acteur

— P2 : L'utilisateur est renvoyé en page d'accueil

Status Entrée/Sortie :

— Entrée : Utilisateur visiteur.

— Sortie : Utilisateur visiteur ou utilisateur inscrit.

Spécification Détaillée

— Scripts

- **Inscription.html** : Page qui permet de rediriger soit vers InscriptionEtudiant.html soit vers InscriptionFamille.html
- **InscriptionEtudiant.html** : Contient un formulaire à remplir par l'utilisateur qui renvoie sur InscriptionEtudiant.php. Si il y a eu une erreur dans le choix de profil à remplir, on peut se rendre tout de même sur famille "InscriptionFamille.html".
- **InscriptionEtudiant.php** : On récupère les \$_POST de la page précédente, on vérifie si les \$_POST des champs obligatoires (*) ne sont pas vides, on vérifie le format de certains champs (email), on chiffre le mot de passe via "password hash", on reformate la date pour la stocker. On a attribué un nom aléatoire temporaire à l'image chargé par l'utilisateur afin d'éviter de se retrouver avec des images possédant le même nom. Les informations saisies sont envoyées vers la base de donnée "projetsi". L'image est chargée dans un dossier "Image" via "move uploaded file". L'utilisateur est ensuite renvoyé vers la page d'accueil des utilisateurs connectés avec une confirmation.
- **InscriptionFamille.html** : Contient un formulaire qui envoie sur InscriptionFamille.php.
- **InscriptionFamille.php** : Fonctionne de manière identique à InscriptionEtudiant.php. Il s'agit de 2 types d'utilisateurs différents donc les champs à remplir ne sont pas strictement identiques.

— Fonctions

- **filter_var(\$_email,FILTER_VALIDATE_EMAIL)** : Vérifie qu'il y a du texte type ***@***.***

— 2.2 Authentification

Spécification Fonctionnelle

Acteurs : Tous utilisateurs.

Pré-condition : L'utilisateur est sur la page d'accueil du site et possède un compte sur le site.

Scénario nominal :

1. L'utilisateur sélectionne "Se connecter"
2. Le système propose un formulaire
3. L'utilisateur entre ses informations (E-mail et mot de passe)
4. Le système vérifie l'exactitude des informations
 - (a) La vérification aboutit
 - (b) La vérification n'aboutit pas
 - i. l'utilisateur se trompe dans son mot de passe
 - ii. au troisième essai le système propose à l'utilisateur "Avez-vous oublié votre mot de passe?"
 - A. l'utilisateur sélectionne "je n'ai pas oublié mon mot de passe"
 - B. le système affiche à nouveau les champs pour se connecter à son compte
 - C. l'utilisateur sélectionne mot de passe oublié → go to P2
 - iii. l'utilisateur a oublié son mot de passe
 - iv. il sélectionne un champ "mot de passe oublié" → go to P2
5. Le système charge les informations de l'utilisateur → go to P1
6. L'utilisateur est redirigé sur la page de garde du site

Post condition :

- P1 : L'utilisateur est authentifié
- P2 : Le système envoie un mail pour mot de passe oublié

Status Entrée/Sortie :

- Entrée : Utilisateur inscrit non authentifié.
- Sortie : Utilisateur inscrit non authentifié ou utilisateur inscrit connecté (UIC).

Spécification Détaillée

Scripts

- **Connexion.html** :
Ce script contient les formulaires nécessaires à la connexion ; il envoie les données via \$_POST à **login.php**.
- **login.php** :
Ce script reçoit des données \$_POST de la page de connexion et prend en entrée un login (email) et un mot de passe. Le script interroge la base de données afin de savoir si le visiteur qui se connecte est bien membre du site. \$_SESSION['type'] est défini en fonction du type d'utilisateur, la fonction **get_infos()** est appelée et le script redirige l'utilisateur connecté en page d'accueil.

Fonctions

- **get_infos(\$bdd,\$login)** :
Cette fonction prend en entrée les informations sur la base de données et le login (email de l'utilisateur) afin de compléter l'array \$_SESSION en fonction des informations renseignées par l'utilisateur. Cette fonction interroge la base de données pour obtenir ces informations (email, nom, prénom, ID).

— 2.3 Edition de profil

Spécification Fonctionnelle

Acteurs : Tous utilisateurs inscrits.

Pré-condition : L'utilisateur possède un compte sur le site.

Scénario nominal :

1. L'utilisateur s'authentifie
2. L'utilisateur sélectionne "Mon compte"
3. Le système charge une page où différentes sections sont présentes
4. L'utilisateur peut modifier sa photo de profil, ses coordonnées, sa description personnelle, ses identifiants et son adresse email
 - (a) L'utilisateur est un étudiant, il peut modifier son emploi du temps
 - (b) l'étudiant peut renseigner son tarif horaire
 - (c) le système prend en compte le tarif horaire de l'étudiant
 - (d) le système calcul les cas spéciaux en fonction du tarif horaire de l'étudiant, les cas spéciaux sont :
 - i. jours fériés en France, le système applique une majoration de 25%
 - ii. les week-ends, le système applique une majoration de 25%
 - iii. la nuit (à partir de 22h jusqu'à 08h) le système applique une majoration de 50%
 - iv. pour chaque enfants supplémentaire le système applique une majoration de 50%
 - (e) l'étudiant peut modifier la grille tarifaire
 - (f) l'étudiant peut modifier son mode d'acceptation des missions (automatique ou manuel) et doit préciser le cas échéant son temps de réponse moyen.
5. Le système propose une validation ou annulation des modifications effectuées
 - (a) L'utilisateur valide les modifications en sélectionnant sur "Valider" → go to P1
 - (b) L'utilisateur annule les modifications en sélectionnant sur "Annuler" → go to P2

Post condition :

- P1 : Le profil de l'utilisateur est modifié, une notification est envoyée aux admins pour qu'ils valident la modification
- P2 : Le profil de l'utilisateur n'est pas modifié

Status Entrée/Sortie :

- Entrée : Utilisateur inscrit connecté (UIC) dont le profil n'est pas à jour.
- Sortie : UIC dont le profil n'est pas à jour ou UIC dont le profil n'est pas à jour.

Spécification Détaillée

- Scripts
 - **BDprofilUtilisateur.php** Récupère les informations d'un utilisateur dans la base de données (Famille / Etudiant / Admin) si certains champs sont vide, la valeur NULL est récupérée.
 - **ModificationProfilUtilisateur.php**

Cette page permet de modifier les informations de l'utilisateurs grâce à des champs prérempli des informations déjà existante sur le profil grâce à la fonction BDProfilUtilisateur.php. Sur cette page se trouve aussi un accès à l'emploi du temps pour les étudiants. Vérifie si l'utilisateur est un super admin / admin ou une famille / étudiant. En fonction du type d'utilisateur renverra vers "dash_admin.php" si l'utilisateur est un admin, "ModificationBDDProfilEtudiant.php" si l'utilisateur est un étudiant, "ModificationBDDProfilFamille.php" s'il s'agit d'une famille. L'utilisateur pourra modifier ses informations. S'il ne modifie pas un champ, récupère l'ancienne information.
 - **ModificationBDDProfilEtudiant.php** Enregistre les modifications que l'étudiant saisit dans la base de données. Le mot de passe est crypté, ainsi il devra forcément entrer un mot de passe (identique, ou nouveau et le confirmer).
 - **ModificationBDDProfilFamille.php** Enregistre les modifications que la famille saisit. Le mot de passe est ici aussi crypté.

— 2.4 Gestion de l'emploi du temps

Spécification Fonctionnelle

Acteurs : Étudiants inscrits.

Pré-condition : L'utilisateur est un étudiant et possède un compte sur le site.

Scénario nominal :

1. L'utilisateur s'authentifie
2. L'utilisateur sélectionne "Éditer mon profil"
3. L'étudiant sélectionne "Mon emploi du temps"
4. Le système affiche un emploi du temps
 - (a) Il s'agit de la première connexion de l'étudiant
 - i. Le système lui affiche un calendrier du mois actuel et du mois prochain
 - ii. L'étudiant sélectionne le/les jour(s) où il est disponible
 - iii. Le système lui affiche des plages horaires à sélectionner
 - iv. L'étudiant sélectionne les tranches horaires pour lesquelles il est disponible
 - v. Le système demande à l'étudiant de valider son emploi du temps
 - vi. L'étudiant valide son emploi du temps en sélectionnant sur "Valider" → go to P1
 - vii. L'étudiant ne valide pas son emploi du temps → go to P2
 - (b) Il ne s'agit pas de sa première connexion, il souhaite modifier son EdT
 - i. Le système lui affiche son emploi du temps
 - ii. L'étudiant sélectionne le(s) jour(s) qu'il souhaite modifier
 - iii. Le système lui détaille les jours sélectionnés
 - iv. L'étudiant désélectionne les plages horaires sélectionnées et l'étudiant a la possibilité d'en sélectionner d'autres
 - Impossible de modifier une plage horaire sur laquelle une mission a été acceptée → go to CU : Annulation
 - v. Le système demande à l'étudiant de valider sa modification
 - vi. L'étudiant valide son emploi du temps → go to P1
 - vii. L'étudiant ne valide pas sa modification → go to P2
5. L'étudiant sélectionner une case lui permettant de dupliquer l'emploi du temps du mois au mois suivant et au mois d'après

Post condition :

- P1 : L'emploi du temps de l'étudiant a été modifié avec succès
- P2 : L'étudiant a annulé ses modifications, son emploi du temps est vierge ou similaire à la précédente version.

Status Entrée/Sortie :

- Entrée : Emploi du temps à modifier.
- Sortie : Emploi du temps à modifier ou emploi du temps modifié.

Spécification Détaillée

Scripts

- **ConsultationProfilUtilisateur.php** : Cette page permet de se connecter à la base de données grâce à connect.php et d'afficher les informations de l'utilisateur en utilisant son ID transmis par \$_SESSION. Les informations proviennent de la table users et varient en fonction du statut de l'utilisateur (Etudiant ou Famille). Pour les étudiants, les commentaires provenant de la table commentaire sont affichés toujours en utilisant le même ID. Ils sont disposés sous la forme d'un formulaire pour pouvoir permettre le signalement de commentaires inappropriés par l'utilisateur en transmettant l'ID des commentaires. Les contrats de l'utilisateur sont affichés, toujours grâce à l'ID de l'utilisateur, depuis la table bookings. En fonction du statut du contrat, il sera affiché dans une rubrique plutôt qu'une autre.
- **ModificationProfilUtilisateur.php** : Précédemment décrite, cf 2.3
- **Calendrier/calendar_etudiant.php** : Page qui permet de gérer son emploi du temps. En rouge s'affiche les créneaux non renseignés par l'étudiant (de base tout le calendrier). L'étudiant sélectionne les jours où il est disponible et les créneaux par tranche de 30 minutes (les créneaux sélectionnés s'affichent en vert, étudiant disponible). S'il est réservé par une famille, le créneau s'affichera en orange.
- **case_contrats.php** : Script permettant de récupérer les données relatives aux contrats en fonction du statut.
- La création du calendrier comporte trois fichiers principaux :
book_slots.php ; calendar_famille.php et class_calendar.php
- **class_calendar.php** : permet dans un premier temps à l'aide de variables facilement modifiables de gérer les jours autorisés de booking sur le site, mais aussi les horaires de départ d'une journée et de fin, la taille des créneaux etc.
- **calendar.php** : Permet l'initialisation du calendrier via l'appel de la méthode make_calendar() du fichier class_calendar.php. On initialise ce calendrier par rapport au jour, au mois et à l'année renseignés dans la barre de recherche de la page Accueil.php.
- **book_slots.php** : permet de récupérer l'ensemble des créneaux cochés par l'étudiant, de les séparer en sous-créniaux individuels qui seront stockés dans la base de donnée.
On crée donc dans la table bookings une ligne par créneau avec l'heure de départ et l'heure de fin du créneau, le champ login de la base de donnée égal à l'id de l'étudiant de la table users, id_famille égal à l'id de la famille qui réserve de la table users et enfin on stocke le prix du créneau dans la base de données.
Ce fichier permet également de s'assurer que le prix total ne dépasse pas une variable seuil (facilement modifiable), dans le cas contraire le contrat devra être validé par un administrateur (c'est à dire proposé aux deux intervenants).
L'acceptation d'un contrat passe également par une nouvelle sécurité qu'est le renseignement à nouveau du nom d'utilisateur et du mot de passe lié au compte afin de s'assurer que ce soit bien le propriétaire du compte qui réalise la réservation.

Fonctions

- **make_calendar()** : Est la fonction principale et permet de créer le calendrier, elle assure le positionnement par rapport au jour actuel mais également permet de définir les mois passés, les mois à suivre, l'appel à la méthode d'import du prix (Change_cost()) mais aussi la vérification des créneaux déjà bookées (make_booking_array()) et de ceux qui restent libres etc.
- **Change_cost()** : Permet d'importer depuis la base de donnée propre au profil de l'étudiant consulté son prix mais également les taux du site (jours fériés, tarifs nocturnes post 18 heure etc.).
- **make_booking_array()** : Permet de définir si oui ou non un créneau a déjà été réalisé ou non. On calcule également le nombre de créneaux par jours en fonctions des paramètres initialement définis (horaire de départ & de fin d'une journée ainsi que la durée d'un créneau). Cette méthode fait appel à la méthode make_days_array().
- **make_days_array()** : Permet de calculer le nombre de jours dans le mois courant mais également dans les autres mois.
- **calendar_top()** : Crée un header au calendrier permettant d'accéder facilement aux mois précédents & aux mois suivants. Cette méthode permet également d'afficher le mois courant.
- **booking_form()** : Affiche les créneaux disponibles dans la journée si l'utilisateur clique sur une journée spécifique dans le calendrier.

— **basket()** : permet de renseigner le panier résultant des créneaux cochés dans le `booking_form()`.

Utilisation de javascript afin de pouvoir cocher les créneaux d'intérêt, et en fonction des modifications changer en temps réel la variable `basket` correspondant au prix total du panier (appelé dans la méthode `basket()` du `class_calendar.php`)

— 2.5 Visite du site

Spécification Fonctionnelle

Acteurs : Tous utilisateurs non connectés

Pré-condition : L'utilisateur est sur la page principale du site

Scénario nominal :

• Consultation de prestataires hors connexion

1. L'utilisateur entre code postal, date voulue (plage horaire / nombre d'heures), nombre d'enfant
2. L'utilisateur sélectionne "Rechercher" pour valider sa recherche
3. Le système traite les informations
4. Le système recherche les étudiants qui sont à proximité, ils sont sélectionnés grâce à leur code postal
 - (a) Le système renvoie à l'utilisateur les différents profils d'étudiants correspondant
mais n'affiche que très peu d'informations, affiche seulement photo, nom, prénom, description perso
 - (b) Le système n'a pas trouvé de babysitter et propose d'élargir la recherche → go to 1
5. Le système propose à l'utilisateur de se connecter ou de consulter en accès libre les données chargées en 3
 - (a) L'utilisateur sélectionne "S'inscrire" → go to CU : Inscription
 - (b) L'utilisateur sélectionne "Se connecter" → go to CU : connexion
 - (c) L'utilisateur sélectionne "Ignorer" → go to 3a
6. L'utilisateur a le choix de
 - (a) Sélectionner un prestataire pour accéder à son profil → go to P1
 - (b) Retourner à la page d'accueil

Post condition :

- P1 : L'utilisateur consulte le prestataire sélectionné

Status Entrée/Sortie :

- Entrée : Utilisateur UIC ou autre voulant consulter des prestataires.
- Sortie : Utilisateur ayant consulté des informations limitées ou non sur des prestataires.

• Contacter le site web

1. L'utilisateur sélectionne "Contact"
2. Le système le redirige en bas de la page
3. L'utilisateur a accès aux informations nécessaires pour contacter l'équipe qui gère le site → go to P1'

Post condition :

- P1' : L'utilisateur contacte les administrateurs

Status Entrée/Sortie' :

- Entrée : Tout utilisateur ayant une question.
- Sortie : Utilisateur ayant posé sa question.

Spécification Détaillée

Fonctions

- **try_session()** : Cette fonction permet de rediriger un utilisateur inscrit et connecté vers la page AccueilConnecté.php grâce à la variable \$_SESSION

Scripts

- **Accueil.php** : Ce script affiche les informations relatives au site et permet à un utilisateur non inscrit ou non connecté de s'inscrire ou de se connecter ainsi que d'effectuer une première visualisation partielle des prestataires qui proposent leurs services. Cette page permet aussi de contacter l'équipe du site grâce à la fonction mailto en php.

— 2.6 Consultation du profil personnel

Spécification Fonctionnelle

Acteurs : Familles et prestataires

Pré-condition : L'acteur est sur la page principale du site

Scénario nominal :

1. L'acteur sélectionne "Mon profil"
2. Le système affiche une page contenant le profil de l'acteur
3. L'utilisateur est un étudiant :
 - (a) L'étudiant peut consulter son profil (informations personnelles) et les commentaires/notes qui lui ont été laissés → go to P1
4. L'utilisateur est une famille :
 - (a) La famille peut consulter son profil (informations personnelles) mais ne voit pas les commentaires/notes qui lui ont été laissés → go to P1

Post condition :

- P1 : L'acteur a consulté son profil

Status Entrée/Sortie :

- Entrée : UIC (famille ou étudiant) voulant consulter son profil personnel.
- Sortie : UIC ayant son profil personnel complet ou non.

Spécification Détaillée

Scripts

- **ConsultationProfilUtilisateur.php** : Cette page permet de se connecter à la base de données grâce à connect.php et d'afficher les informations de l'utilisateur en utilisant son ID transmis par \$_SESSION. Les informations proviennent de la table users et varient en fonction du statut de l'utilisateur (Etudiant ou Famille). Pour les étudiants, les commentaires provenant de la table commentaire sont affichés toujours en utilisant le même ID. Ils sont disposés sous la forme d'un formulaire pour pouvoir permettre le signalement de commentaires inappropriés par l'utilisateur en transmettant l'ID des commentaires. Les contrats de l'utilisateur sont affichés, toujours grâce à l'ID de l'utilisateur, depuis la table bookings. Les utilisateurs peuvent agir sur les contrats. L'étudiant peut accepter dans un premier temps la mission ou la refuser. Si l'étudiant a accepté la mission, la famille et l'étudiant peuvent annuler cette mission. Si la mission a été effectuée, les deux partis peuvent s'évaluer et la famille peut déclarer un litige. En fonction du statut du contrat, il sera affiché dans une rubrique plutôt qu'une autre.
- **case_contrats.php** : Script permettant de récupérer les données relatives aux contrats en fonction du statut.
- **decision_contrat.php** : Traite les données envoyées depuis "ConsultationProfilUtilisateur.php" (Accepter / Refuser / Annuler / Déclarer un litige) et met à jour les statuts des contrats dans la table booking. Ce script gère les différents cas d'annulation (> 48h ou < 48h) et la déclaration d'un litige. Il permet également d'envoyer un e-mail lorsqu'il y a acceptation / refus / ou annulation.

— 2.7 Demande de prestataire

Spécification Fonctionnelle

Acteurs : Utilisateur famille

Pré-condition : L'utilisateur est sur le site et connecté à son compte

Scénario nominal :

1. L'utilisateur entre code postal, date voulue (plage horaire / nombre d'heures), nombre d'enfants
2. L'utilisateur sélectionne "Rechercher" pour valider sa recherche
3. Le système traite les données envoyées
4. le système filtre les étudiants à proximité grâce à leur code postal
 - (a) Le système renvoie à l'utilisateur les différents profils d'étudiants correspondant (note visible) ainsi qu'une évaluation du prix.
 - (b) Le système n'a pas trouvé de babysitter → go to 1
5. L'utilisateur à plusieurs choix
 - (a) L'utilisateur peut sélectionner un profil
 - (b) L'utilisateur peut ré-effectuer une recherche → go to 1
6. Le profil s'ouvre dans un nouvel onglet
7. L'utilisateur a accès à toute les informations du profil (commentaires, etc...) et
 - (a) L'utilisateur sélectionne "contacter"
 - (b) L'utilisateur peut fermer l'onglet → go to 3a
8. Le système affiche une fenêtre de validation
 - (a) L'utilisateur valide et le système prends en compte la requête → P1
 - (b) L'utilisateur choisit de ne pas valider → go to 6

Post condition :

- P1 : Le système a envoyé une demande de mission détaillée (nombre d'enfant, date, rémunération, lieux) à un étudiant

Status Entrée/Sortie :

- Entrée : UIC (famille) voulant réserver un prestataire.
- Sortie : UIC (famille) ayant réservé un prestataire ou non.

Spécification Détaillée

Scripts

— **ResultatConnecte.php** :

Récupère les \$_POST de AccueilConnecte.php, puis va chercher dans la BDD les étudiants dont les champs répondent à la requête. SI il n'y a pas de résultats ou trop peu, le bouton "Elargir la recherche?" reprend le code postal en prenant cette fois les 2 premiers caractères et renvoie sur la même page à l'aide d'un \$_GET le code postal raccourci, toujours le même nombre d'enfants et même plage horaire.

— **ConsultationProfilUtilisateurTiers.php** :

Cette page permet de se connecter à la base de données grâce à connect.php et d'afficher les informations d'un utilisateur visité en utilisant l'ID de ce dernier transmis par \$_POST. Les informations proviennent de la table users et varient en fonction du statut de l'utilisateur (Etudiant ou Famille). Les commentaires provenant de la table commentaire sont affichés toujours en utilisant le même ID. Ils sont disposés sous la forme d'un formulaire pour pouvoir permettre le signalement de commentaires inappropriés par l'utilisateur en transmettant l'ID des commentaires. Si l'utilisateur consultant est un étudiant, il pourra soit accepter ou refuser l'offre de contrat soit revenir directement à son profil. Si l'utilisateur consultant est une famille, il pourra soit enclencher la mise en forme d'un contrat soit revenir à la recherche d'étudiants.

Fonctions

- **substr()** : Fonction php qui permet de récupérer seulement les deux premiers caractères du code postal.

— 2.8 Acceptation du contrat

Spécification Fonctionnelle

Acteurs : Prestataires inscrits et connectés

Pré-condition : Le prestataire est alerté d'une nouvelle offre/a une nouvelle offre

Scénario nominal :

1. L'étudiant se connecte grâce à ses identifiants
2. L'étudiant va sur son profil/onglet mission
3. L'étudiant consulte l'offre proposée et peut accéder au profil complet de la famille (informations personnelles + commentaires)
4. Le système affiche l'adresse E-mail de la famille
5. Le prestataire doit accepter ou refuser l'offre
 - (a) Le prestataire accepte la mission → go to P1
 - (b) Le prestataire refuse l'offre, son créneau est toujours libre → go to P2
6. Son créneau est bloqué et réservé pour la famille → go to P1

Post condition :

- P1 : Le prestataire a accepté l'offre, le créneau correspondant est réservé, un mail est envoyé pour informer la famille et lui demander de payer.
- P2 : Le prestataire a refusé l'offre, pas de modification, un mail est envoyé pour informer la famille du refus de l'offre.

Status Entrée/Sortie :

- Entrée : UIC (étudiant) sollicité pour une mission.
- Sortie : UIC (étudiant) missionné ou non.

Spécification Détaillée

Scripts

— **ConsultationProfilUtilisateur.php :**

Cette page permet de se connecter à la base de données grâce à connect.php et d'afficher les informations de l'utilisateur en utilisant son ID transmis par \$_SESSION. Les informations proviennent de la table users et varient en fonction du statut de l'utilisateur (Etudiant ou Famille). Pour les étudiants, les commentaires provenant de la table commentaire sont affichés toujours en utilisant le même ID. Ils sont disposés sous la forme d'un formulaire pour pouvoir permettre le signalement de commentaires inappropriés par l'utilisateur en transmettant l'ID des commentaires. Les contrats de l'utilisateur sont affichés, toujours grâce à l'ID de l'utilisateur, depuis la table bookings. Les utilisateurs peuvent agir sur les contrats. L'étudiant peut accepter dans un premier temps la mission ou la refuser. Si l'étudiant a accepté la mission, la famille et l'étudiant peuvent annuler cette mission. Si la mission a été effectuée, les deux partis peuvent s'évaluer et la famille peut déclarer un litige. En fonction du statut du contrat, il sera affiché dans une rubrique plutôt qu'une autre.

— **case_contrats.php :**

Script permettant de récupérer les données relatives aux contrats en fonction du statut.

— **decision_contrat.php :**

Traite les données envoyées depuis "ConsultationProfilUtilisateur.php" (Accepter / Refuser / Annuler / Déclarer un litige) et met à jour les statuts des contrats dans la table booking. Ce script gère les différents cas d'annulation (> 48h ou < 48h) et la déclaration d'un litige. Il permet également d'envoyer un e-mail lorsqu'il y a acceptation / refus / ou annulation.

Fonctions

— **recup_data_contrat_unlock_etudiant(\$bdd) :**

récupère des données de la base de données (bdd) pour un étudiant et une famille suite à une demande de réservation par la famille de l'étudiant en question. Le statut est égale à 1.

— **recup_data_contrat_lock_etudiant(\$bdd) :**

récupère des données de la base de données (bdd) pour un étudiant et une famille suite à l'acceptation d'une mission par l'étudiant. Le statut est égale à 2.

— **recup_data_contrat_unlock_famille(\$bdd) :**

récupère des données de la base de données (bdd) pour un étudiant et une famille suite à une demande de réservation par la famille de l'étudiant en question. Le statut est égale à 1.

— **recup_data_contrat_lock_famille(\$bdd) :**

récupère des données de la base de données (bdd) pour un étudiant et une famille suite à l'acceptation d'une mission par l'étudiant. Le statut est égale à 2.

— **recup_contrat_passes_famille(\$bdd) :**

récupère les contrats passés de la famille inférieurs à 48h. Validation est égale à 1.

— **recup_contrat_passes_etudiant(\$bdd) :**

récupère les contrats passés des étudiants inférieurs à 48h. Validation est égale à 1.

— 2.9 Paiement du prestataire

Spécification Fonctionnelle

Acteurs : Famille inscrites et identifiées

Pré-condition : Le prestataire a accepté l'offre

Scénario nominal :

1. La famille se connecte sur le site
2. Le système redirige la famille pour le paiement (Redirection vers la page PayPal)
 - (a) La famille remplit les champs nécessaires
 - (b) La famille peut annuler le paiement avant de remplir les champs → go to P2
3. La famille valide ou invalide le formulaire
 - (a) La famille valide les champs
 - (b) La famille peut annuler le paiement après avoir rempli les champs → go to P2
4. Le système vérifie les informations
 - (a) Les champs sont tous remplis correctement
 - (b) Les champs ne sont pas remplis correctement → Go to 2a
5. La famille confirme la validation
 - (a) La famille sélectionne : "Oui"
 - (b) La famille sélectionne : "Non" → Go to P2
6. Paiement validé → Go to P1

Post condition :

- P1 : Le paiement est effectué, l'argent est bloqué jusqu'à 48h après la garde, l'étudiant est informé
- P2 : Le paiement est annulé

Status Entrée/Sortie :

- Entrée : Prestataire non rémunéré.
- Sortie : Prestataire rémunéré ou non.

Spécification Détaillée

Scripts

- **Calendrier/calendar_etudiant.php :**
Page qui permet de gérer son emploi du temps. Lorsque la famille recherche un étudiant, et qu'il en sélectionne un. Il est redirigé vers le calendrier afin de réserver les créneaux qui lui conviennent. Il réservera ensuite l'étudiant en entrant son login. Une fois le créneau validé, il sera redirigé vers une page Paypal pour procéder au virement.
- **ConsultationProfilUtilisateur.php :**
Cette page permet de se connecter à la base de données grâce à connect.php et d'afficher les informations de l'utilisateur en utilisant son ID transmis par \$_SESSION. Les informations proviennent de la table users et varient en fonction du statut de l'utilisateur (Etudiant ou Famille). Pour les étudiants, les commentaires provenant de la table commentaire sont affichés toujours en utilisant le même ID. Ils sont disposés sous la forme d'un formulaire pour pouvoir permettre le signalement de commentaires inappropriés par l'utilisateur en transmettant l'ID des commentaires. Les contrats de l'utilisateur sont affichés, toujours grâce à l'ID de l'utilisateur, depuis la table bookings. Les utilisateurs peuvent agir sur les contrats. L'étudiant peut accepter dans un premier temps la mission ou la refuser. Si l'étudiant a accepté la mission, la famille et l'étudiant peuvent annuler cette mission. Si la mission a été effectuée, les deux partis peuvent s'évaluer et la famille peut déclarer un litige. En fonction du statut du contrat, il sera affiché dans une rubrique plutôt qu'une autre. Si le prix contrat est trop élevé le statut passe à 3, et un admin doit intervenir. Suite à un litige si l'admin ne donne pas suite au litige, il peut débloquent le paiement. Si il n'y a pas de litige, les fonds sont débloqués après 48h et l'étudiant est payé.
- **case_contrats.php :**
Script permettant de récupérer les données relatives aux contrats en fonction du statut.
- **Update.php :**
Passe les anciens contrats à validation =1.

Fonctions

- **recup_data_contrat_unlock_etudiant(\$bdd) :**
Récupère des données de la base de données (bdd) pour un étudiant et une famille suite à une demande de réservation par la famille de l'étudiant en question. Le statut est égale à 1.
- **recup_data_contrat_lock_etudiant(\$bdd) :**
Récupère des données de la base de données (bdd) pour un étudiant et une famille suite à l'acceptation d'une mission par l'étudiant. Le statut est égale à 2.
- **recup_data_contrat_unlock_famille(\$bdd) :**
Récupère des données de la base de données (bdd) pour un étudiant et une famille suite à une demande de réservation par la famille de l'étudiant en question. Le statut est égale à 1.
- **recup_data_contrat_lock_famille(\$bdd) :**
Récupère des données de la base de données (bdd) pour un étudiant et une famille suite à l'acceptation d'une mission par l'étudiant. Le statut est égale à 2.
- **recup_contrat_passes_famille(\$bdd) :**
Récupère les contrats passés de la famille inférieurs à 48h. Validation est égale à 1.
- **recup_contrat_passes_etudiant(\$bdd) :**
Récupère les contrats passés des étudiants inférieurs à 48h. Validation est égale à 1.

— 2.10 Annulation

Spécification Fonctionnelle

Acteurs : Utilisateurs inscrits et connectés

Pré-condition : Contrat en cours

Scénario nominal :

1. L'utilisateur se connecte grace à ses identifiants (voir CU : Connexion)
2. L'utilisateur accède à son profil et sélectionne "Mission"
3. Le système affiche les contrats de l'utilisateur
4. L'utilisateur sélectionne la mission qu'il souhaite annuler
5. Le système propose une confirmation de l'annulation de la mission de l'utilisateur
 - (a) L'utilisateur valide l'annulation
 - (b) L'utilisateur annule l'annulation → go to P2
6. L'utilisateur est un étudiant :
 - (a) Si la mission aboutit dans plus de 48 heures : la mission est annulée le créneau est libéré, la famille est remboursée → go to P1
 - (b) Si la mission aboutit dans moins de 48 heures : la mission est annulée, le système modifie le taux d'annulation du prestataire, la famille est remboursée, le créneau est libéré → go to P1
7. L'utilisateur est une famille :
 - (a) Si la mission aboutit dans plus de 48 heures : La mission est annulée, le créneau du prestataire est libéré, la famille n'a rien est remboursée → go to P1
 - (b) Si la mission aboutit dans moins de 48 heures : La mission est annulée, la totalité du paiement est tout de même effectuée → go to P1

Post condition :

— P1 : La mission est annulée

— P2 : La mission est toujours valide

Status Entrée/Sortie :

— Entrée : UIC (étudiant) missionné.

— Sortie : UIC (étudiant) missionné ou non.

Spécification Détaillée

Scripts

— **ConsultationProfilUtilisateur.php :**

Cette page permet de se connecter à la base de données grâce à connect.php et d'afficher les informations de l'utilisateur en utilisant son ID transmis par \$_SESSION. Les informations proviennent de la table users et varient en fonction du statut de l'utilisateur (Etudiant ou Famille). Pour les étudiants, les commentaires provenant de la table commentaire sont affichés toujours en utilisant le même ID. Ils sont disposés sous la forme d'un formulaire pour pouvoir permettre le signalement de commentaires inappropriés par l'utilisateur en transmettant l'ID des commentaires. Les contrats de l'utilisateur sont affichés, toujours grâce à l'ID de l'utilisateur, depuis la table bookings. Les utilisateurs peuvent agir sur les contrats. L'étudiant peut accepter dans un premier temps la mission ou la refuser. Si l'étudiant a accepté la mission, la famille et l'étudiant peuvent annuler cette mission. Si la mission a été effectuée, les deux partis peuvent s'évaluer et la famille peut déclarer un litige. En fonction du statut du contrat, il sera affiché dans une rubrique plutôt qu'une autre.

— **case_contrats.php :**

Script permettant de récupérer les données relatives aux contrats en fonction du statut.

— **decision_contrat.php :**

Traite les données envoyées depuis "ConsultationProfilUtilisateur.php" (Accepter / Refuser / Annuler / Déclarer un litige) et met à jour les statuts des contrats dans la table booking. Ce script gère les différents cas d'annulation (> 48h ou < 48h) et la déclaration d'un litige. Il permet également d'envoyer un e-mail lorsqu'il y a acceptation / refus / ou annulation.

Fonctions

— **recup_data_contrat_unlock_etudiant(\$bdd) :**

récupère des données de la base de données (bdd) pour un étudiant et une famille suite à une demande de réservation par la famille de l'étudiant en question. Le statut est égale à 1.

— **recup_data_contrat_lock_etudiant(\$bdd) :**

récupère des données de la base de données (bdd) pour un étudiant et une famille suite à l'acceptation d'une mission par l'étudiant. Le statut est égale à 2.

— **recup_data_contrat_unlock_famille(\$bdd) :**

récupère des données de la base de données (bdd) pour un étudiant et une famille suite à une demande de réservation par la famille de l'étudiant en question. Le statut est égale à 1.

— **recup_data_contrat_lock_famille(\$bdd) :**

récupère des données de la base de données (bdd) pour un étudiant et une famille suite à l'acceptation d'une mission par l'étudiant. Le statut est égale à 2.

— **recup_contrat_passes_famille(\$bdd) :**

récupère les contrats passés de la famille inférieure à 48h. Validation est égale à 1.

— **recup_contrat_passes_etudiant(\$bdd) :**

récupère les contrats passés des étudiants inférieurs à 48h. Validation est égale à 1.

— 2.11 Évaluation prestataire/famille

Spécification Fonctionnelle

Acteurs : Utilisateurs inscrits et connectés

Pré-condition : Une mission vient d'être réalisée

Scénario nominal :

1. L'utilisateur se connecte à son compte (voir CU : Connexion)
2. Une notification du système propose à l'utilisateur d'évaluer sa mission précédente
 - (a) L'utilisateur refuse, la notification est effacée → go to P2
 - (b) L'utilisateur accepte
3. L'utilisateur est renvoyé sur la page de son profil listant ses dernières missions
4. L'utilisateur attribue une note sur 5 et peut laisser un commentaire
5. Le système change les infos du profil utilisateur concerné → go to P1

Post condition :

- P1 : Un utilisateur a reçu une évaluation et son profil a été modifié en conséquence
- P2 : Aucune évaluation ni modification

Status Entrée/Sortie :

- Entrée : UIC (étudiant) ayant effectué une mission n'ayant pas reçu de commentaire.
- Sortie : UIC (étudiant) ayant effectué une mission ayant reçu un commentaire ou non.

Spécification Détaillée

Script

— **ConsultationProfilUtilisateur.php :**

Cette page permet de se connecter à la base de données grâce à connect.php et d'afficher les informations de l'utilisateur en utilisant son ID transmis par \$_SESSION. Les informations proviennent de la table users et varient en fonction du statut de l'utilisateur (Etudiant ou Famille). Pour les étudiants, les commentaires provenant de la table commentaire sont affichés toujours en utilisant le même ID. Ils sont disposés sous la forme d'un formulaire pour pouvoir permettre le signalement de commentaires inappropriés par l'utilisateur en transmettant l'ID des commentaires. Les contrats de l'utilisateur sont affichés, toujours grâce à l'ID de l'utilisateur, depuis la table bookings. Les utilisateurs peuvent agir sur les contrats. L'étudiant peut accepter dans un premier temps la mission ou la refuser. Si l'étudiant a accepté la mission, la famille et l'étudiant peuvent annuler cette mission. Si la mission a été effectuée, les deux partis peuvent s'évaluer et la famille peut déclarer un litige. En fonction du statut du contrat, il sera affiché dans une rubrique plutôt qu'une autre.

— **Notation.php :**

Suite à une mission l'utilisateur commenter la prestation et la noter. Les notes vont de 1 à 5 (1 = Nul / 5 = Parfait), et commenter la prestation. Les différentes informations seront envoyées dans la base de données par "commentaire_bdd.php".

— **commentaire_bdd.php :**

Envoie les commentaires et notes des utilisateurs dans la base de données. Les commentaires seront ensuite visible sur le profil des utilisateurs.

— 2.12 Signalement d'un commentaire inapproprié

Spécification Fonctionnelle

Acteurs : Utilisateurs inscrits et connectés

Pré-condition : L'utilisateur connecté consulte un profil et lit les avis

Scénario nominal :

1. L'utilisateur observe un profil
2. L'utilisateur repère un message abusif
3. L'utilisateur sélectionne un jalon de signalisation
4. Le système envoie un message aux administrateurs → go to P1

Post condition :

- P1 : Un message d'avertissement a été envoyé aux administrateurs au sujet d'un commentaire inapproprié

Status Entrée/Sortie :

- Entrée : Message non signalé.
- Sortie : Message définit comme inapproprié et signalé.

Spécification Détaillée

Scripts

- **ConsultationProfilUtilisateur.php** : Script présenté en section 2.3
- **ConsultationProfilUtilisateurTiers.php** : Script présenté en section 2.7
- **Moderation_com.php** : Cette page récupère l'ID du ou des commentaires transmis en \$_POST par une page de consultation de profil. le champs moderation de la table commentaire de ce(s) dernier(s) sera(ont) modifié(s) en "1". Lorsque cette modification a été faite, un message de confirmation est affiché et renvoie automatiquement l'utilisateur sur la page précédente.

— 2.13 Modification d'un commentaire inapproprié

Spécification Fonctionnelle

Acteurs : Admin ou super admin

Pré-condition : L'admin connecté a reçu une signalisation à propos d'un message abusif

1. L'admin observe le profil signalé
2. L'admin repère le message abusif
3. L'admin joue de son droit décisionnel :
 - (a) Le message est supprimé → go to P1
 - (b) Le message est indemne → go to P2

Post condition :

- P1 : Le message abusif a été supprimé, le profil utilisateur est mis à jour, la note n'est pas modifiée
- P2 : Aucune modification

Status Entrée/Sortie :

- Entrée : Message défini comme inapproprié et signalé.
- Sortie : Message considéré inapproprié ou non (message supprimé ou ignoré).

Spécification Détaillée

Scripts

- **dash_admin_moderation.php** :

Ce script affiche la page de modération où les commentaires signalés sont affichés (appelés depuis la base de données) les uns à la suite de autres. Ce script envoie un formulaire au script suivant en fonction du choix de l'utilisateur.

- **moderate_comment.php** :

Ce script permet de modifier le caractère 'signalé' d'un commentaire dans la base de données ou de simplement le supprimer si nécessaire. Il nécessite l'envoi de l'array \$_POST contenant le champ 'modifié' ou 'supprimé'. Le script redirige l'utilisateur vers la page de modération une fois la tâche accomplie.

Fonctions

- **get_comments(\$bdd)** :

Cette fonction permet d'accéder à la base de données et d'afficher les données de l'utilisateur ayant laissé un commentaire signalé ainsi que le commentaire en question et les choix de l'utilisateur.

— 2.14 Validation d'un profil

Spécification Fonctionnelle

Acteurs : Admin ou super admin

Pré-condition : L'admin connecté a reçu une signalisation à propos d'un nouveau profil ou d'une modification de profil.

1. L'admin observe le profil signalé
2. L'admin joue de son droit décisionnel :
 - (a) Le profil est acceptable → go to P1
 - (b) Le profil est non conforme → go to P2

Post condition :

- P1 : Le profil est validé
- P2 : Le profil est modéré et validé

Status Entrée/Sortie :

- Entrée : Profil en attente de validation
- Sortie : Profil validé ou modéré

Spécification Détaillée

Scripts

- **dash_admin_profile_etu.php** :
Ce script affiche la page de modération où les profils étudiants récemment créés ou signalés sont affichés (appelés depuis la base de données). Ce script envoie un formulaire au script moderate_profile.php.
- **dash_admin_profile_fam.php** :
Idem pour les utilisateurs : famille
- **moderate_profile.php** :
Ce script permet de modifier le caractère 'signalé' d'un profil dans la base de données ou de simplement le supprimer ou de le modifier si nécessaire. Il nécessite l'envoi de l'array \$_POST contenant le champ 'ignoré', 'modifié' ou 'supprimé' ainsi que l'ID du profil pour un affichage cohérent. Le script redirige l'utilisateur vers la page de modération une fois la tâche accomplie.

Fonctions

- **get_comments(\$bdd)** :
Cette fonction permet d'accéder à la base de données et d'afficher les données de l'utilisateur ayant laissé un commentaire signalé ainsi que le commentaire en question et les choix de l'utilisateur.

— 2.15 Gestion des divers taux par les Admins

Spécification Fonctionnelle

Acteurs : Admins du site

Pré-condition : L'utilisateur est un Admin

Scénario nominal :

1. L'utilisateur se connecte en tant qu'Admin
2. Il accède à la grille des différents taux gérés par le site
3. L'admin peut modifier les cas spéciaux (taux jour férié, taux week-end, taux nuit)
4. L'admin peut modifier le taux de commission prélevé par le site
5. Après chaque modifications le système propose une confirmation de la modification du taux → go to P1
6. L'admin annule la modification → go to p2
7. Un mail stipulant que les conditions d'utilisations générales ont été modifiées est envoyé à tout les inscrits

Post condition :

- P1 : Le système a pris en compte la modification du taux
- P2 : Aucun taux n'a été modifié

Status Entrée/Sortie :

- Entrée : Taux précédemment fixées.
- Sortie : Taux modifiés ou non ; email envoyé ou non.

Spécification Détaillée

Scripts

- **dash_admin_taux.php** :
Ce script affiche dans un formulaire les taux définis et appelés depuis la base de données grâce à la fonction `get_taux($bdd)`. Ce script permet de d'envoyer un formulaire au script suivant.
- **Moderate_taux.php** :
Ce script reçoit par la variable `$_POST` les informations relatives aux taux à modifier.

Fonctions

- **get_taux(\$bdd)** :
Cette fonction prend en entrée les informations sur la base de données et renvoie les taux définis dans celle ci.

— 2.16 Création d'un compte Admin

Spécification Fonctionnelle

Acteurs : Super-admin

Pré-condition : Le super admin est connecté sur sa page d'administration

Scénario nominal :

1. Le super admin sélectionne l'onglet "Ajout d'administrateurs"
2. Le super admin renseigne un formulaire : adresse e-mail et mot de passe
3. Le système vérifie la validité de l'adresse E-mail
 - (a) L'e-mail est validé, un nouveau compte a été créé → go to P1
 - (b) Le-mail n'est pas validé → go to 3

Post condition :

— P1 : L'admin a reçu un login et un mot de passe

Status Entrée/Sortie :

— Entrée : n administrateurs enregistrés.

— Sortie : n+1 administrateurs enregistrés.

Spécification Détaillée

Scripts

— **dash_admin_nouvel_admin.php** :

Ce script affiche des formulaires permettant d'ajouter ou de supprimer un administrateur de la base de données. Grâce à la fonction `get_admin($bdd)`, le script permet d'afficher les adresses mail des administrateurs (`admin`) présents dans la base. Cette page étant réservée exclusivement à l'administrateur 'super', la fonction `which_admin()` se charge de définir le type d'administrateur. D'autre part l'accès aux pages administrateur est restreint pour les autres utilisateurs par la fonction `try_session()`.

Fonctions

— **get_admin(\$bdd)** :

Cette fonction prend en entrée les informations sur la base de données et affiche les administrateurs définis dans celle ci.

— **inscription_admin()** :

Cette fonction prend en entrée les données transmises par la variable `$_POST` (email et mot de passe) et permet d'ajouter un administrateur à la base.

— **suppression_admin()** :

Cette fonction prend en entrée les données transmises par la variable `$_POST` (email) et permet de supprimer un administrateur de la base.

— **which_admin()** :

Cette fonction renvoi un booléen permettant de définir à partir de la variable super globale `$_session` le type d'administrateur étant connecté.

— **try_session()** :

Cette fonction est utilisé ici pour restreindre l'accès aux pages admins à des utilisateurs non connectés ou connectés en temps que famille ou étudiant. Cette fonction utilise la variable super globale `$_session` et redirige l'utilisateur en conséquence.

— Autres scripts et fonctions importantes

Scripts php

- **connect.php** :
Script utilisé dans la plupart des scripts de ce projet qui permet de se connecter à la base de données.
- **logout.php** : Ce script permet de déconnecter un utilisateur connecté.
- **functions.php** :
Script qui contient les fonctions relatives à l'interface étudiants et familles.
- **functions_admin.php** :
Script qui contient les fonctions relatives à l'interface administrateurs.

Fonctions

- **logged_in()** : Permet à un utilisateur déjà enregistré et connecté d'être redirigé vers la page des utilisateurs connectés sur le site.
- **autologout()** : Cette fonction permet de déconnecter un utilisateur après 1/2h.

Scripts CSS

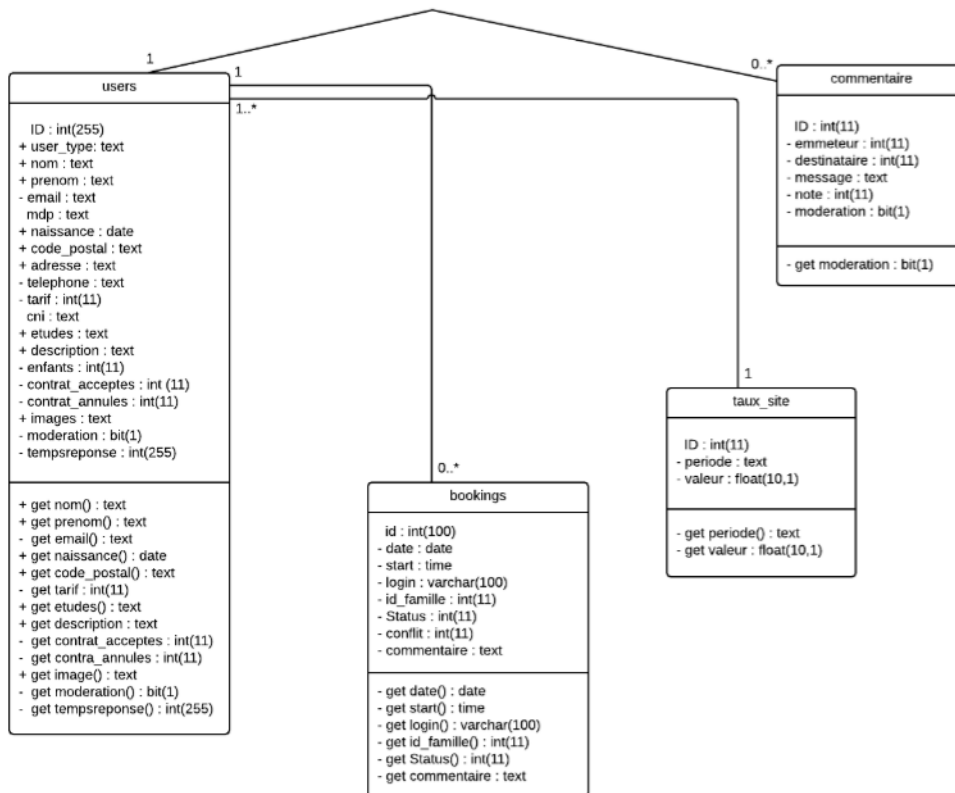
- **Style_profil.css** Ce CSS permet d'afficher les informations des pages de profils dans 5 à 6 rubriques dont l'une comporte une commande `overflow : auto ;` permettant d'afficher les commentaires avec une barre de scrolling, ce qui nous a permis d'afficher tout les commentaires sans pour autant avoir une page qui paraisse surchargée.
- Le site a été conçu dans une optique de responsive design à l'aide des media query en css. C'est dans cette optique d'auto adaptabilité que nous avons permis un affichage optimisé quel que soit le support utilisé (testé sur les marques Samsungs, Ipad, Nvidia Shield K1, etc.). Si par exemple l'utilisateur décide de réaliser une partie de l'inscription sur son ordinateur, effectuer une recherche et bloquer un créneau sur sa tablette et enfin valider le contrat suite à un mail reçu sur son portable ; cela lui est tout à fait possible.

3 Analyse

3.1 Diagrammes de plus haut niveau

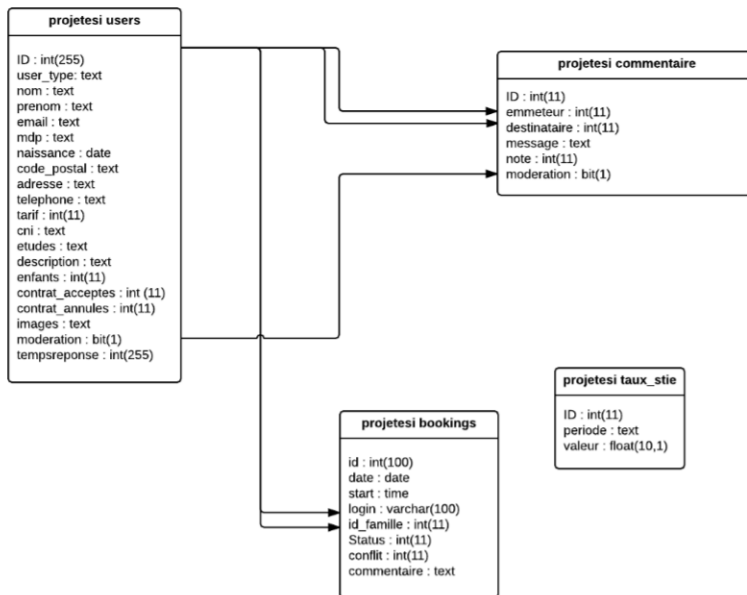
Le projet met en jeux quatre types d'utilisateurs.

3.2 Diagrammes de classes métier



3.3 Modèle logique de la base de données

Modèle logique de la base de données (les tables avec les clés primaires et les clés étrangères - expliquez comment vous avez traduit l'héritage)



La base de données est répartie en 4 tables

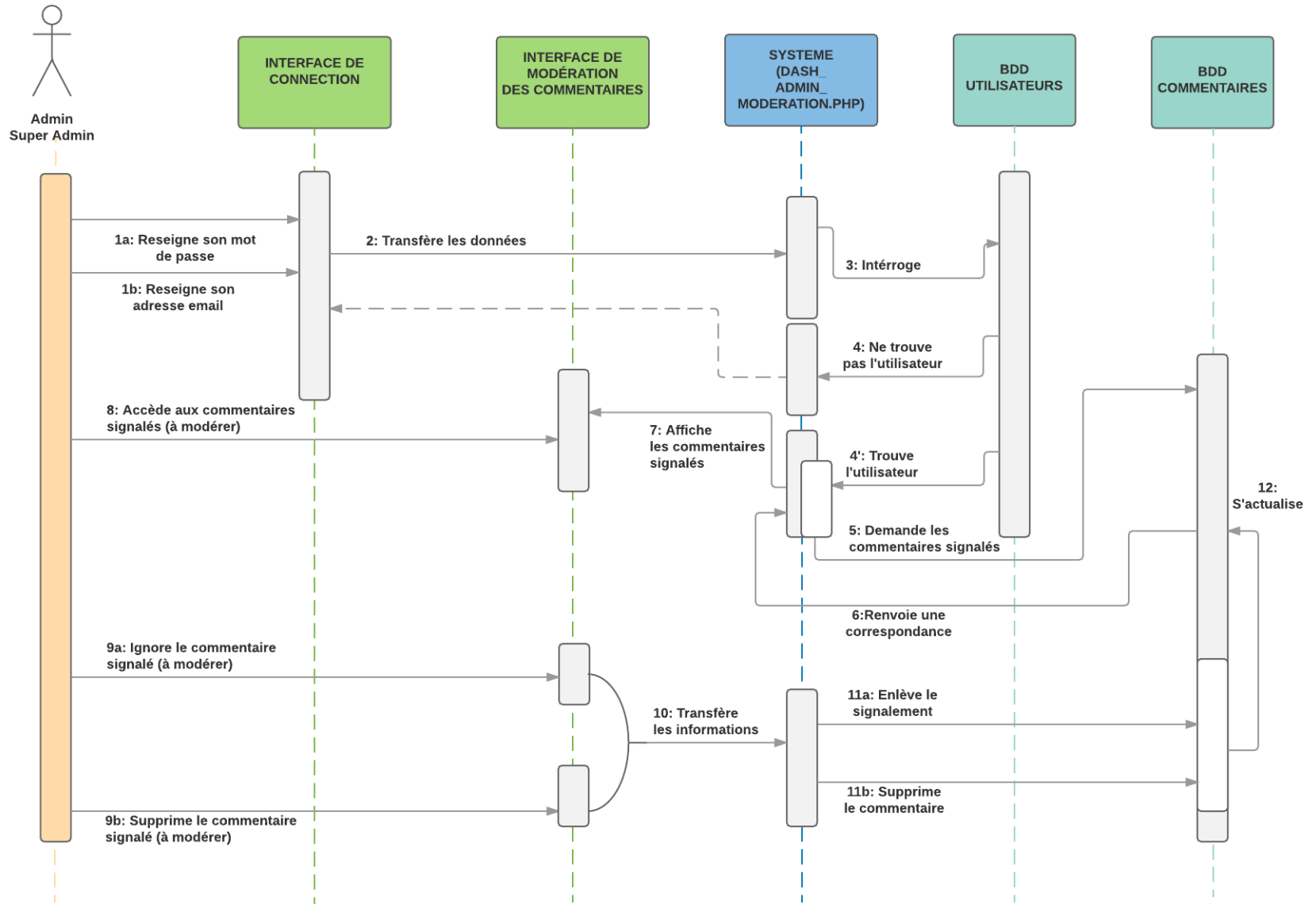
- Booking
- Commentaire
- Users
- Taux_sites

4 Conception

4.1 Diagramme de séquence : modération d'un commentaire

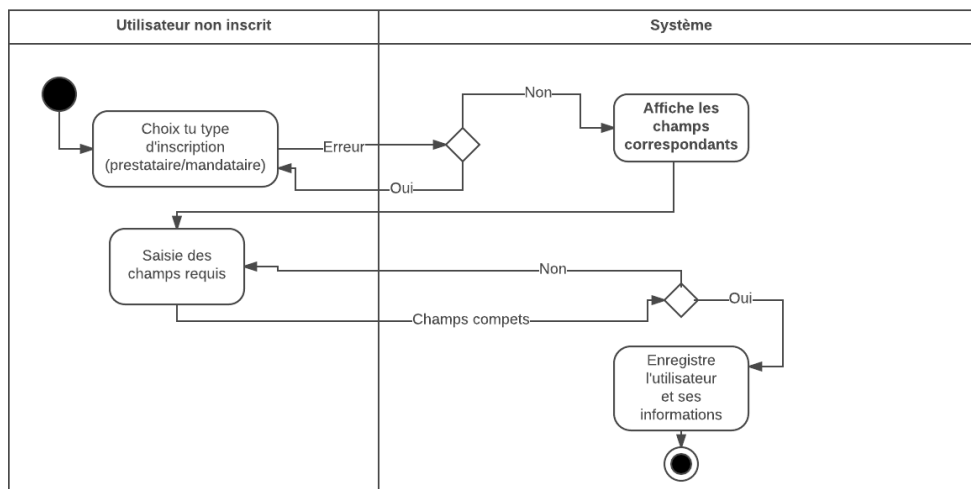
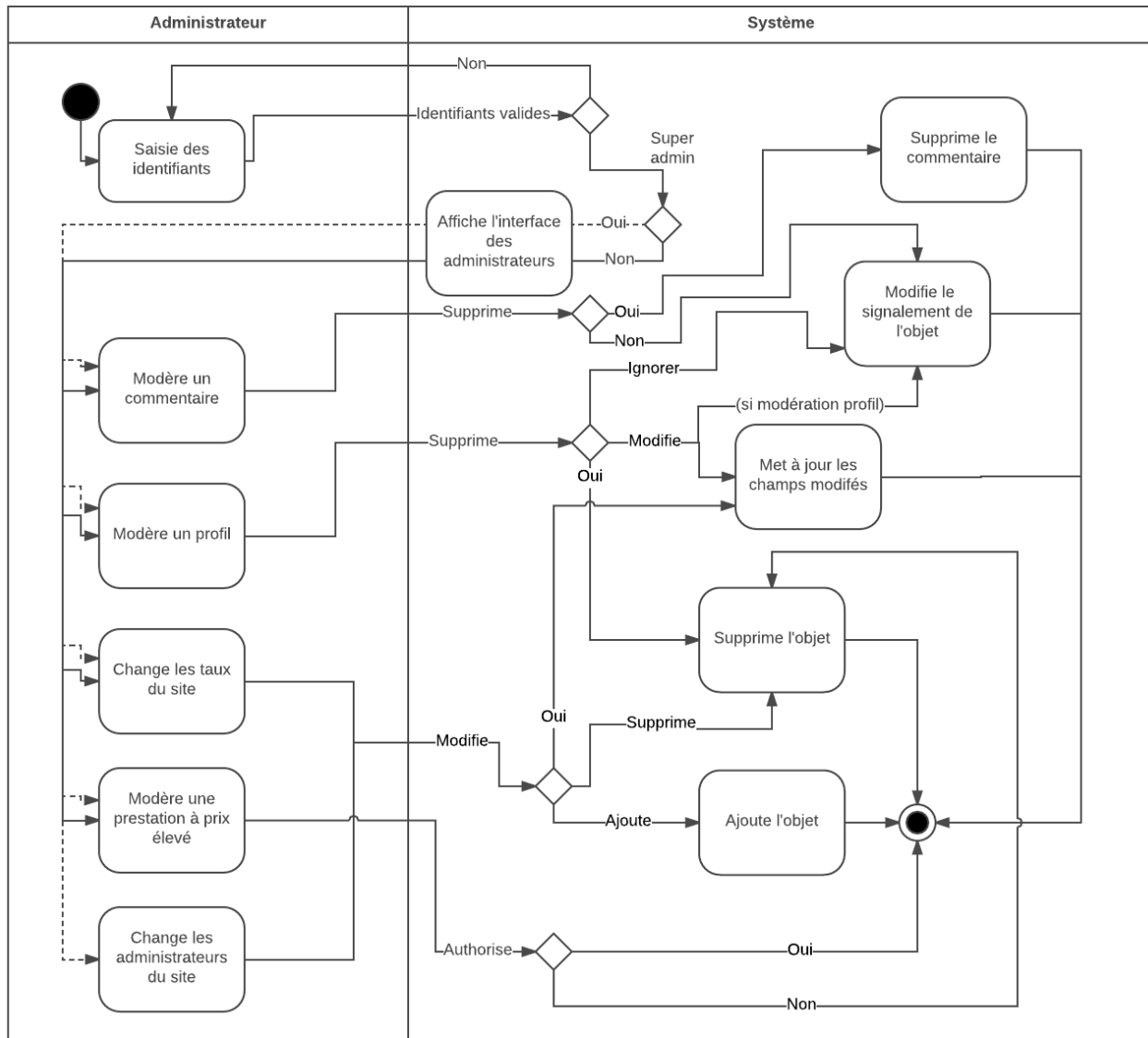
DIAGRAMME DE SÉQUENCE: MODÉRATION DES COMMENTAIRES SIGNALÉS

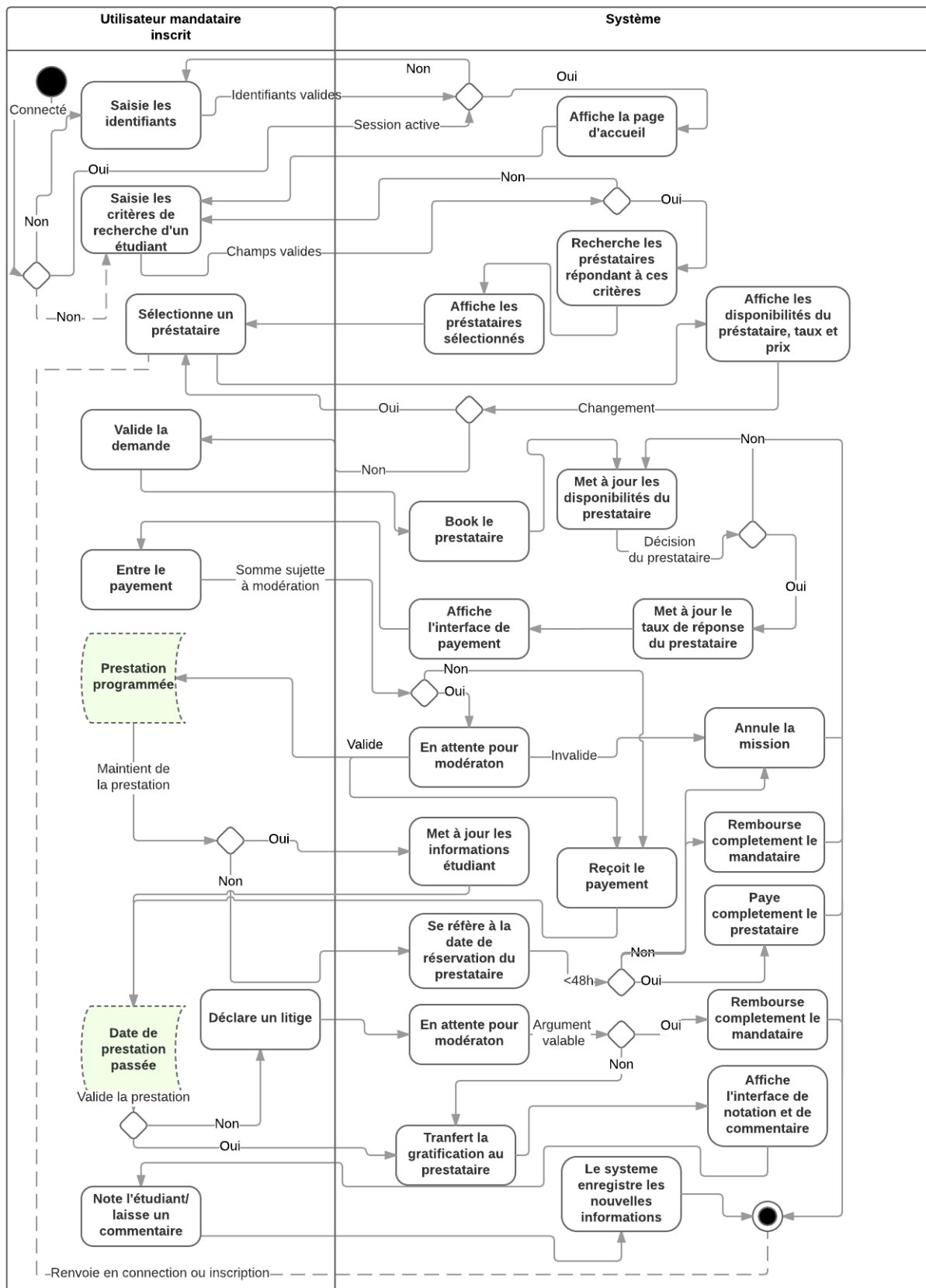
Esitting | Décembre 2016

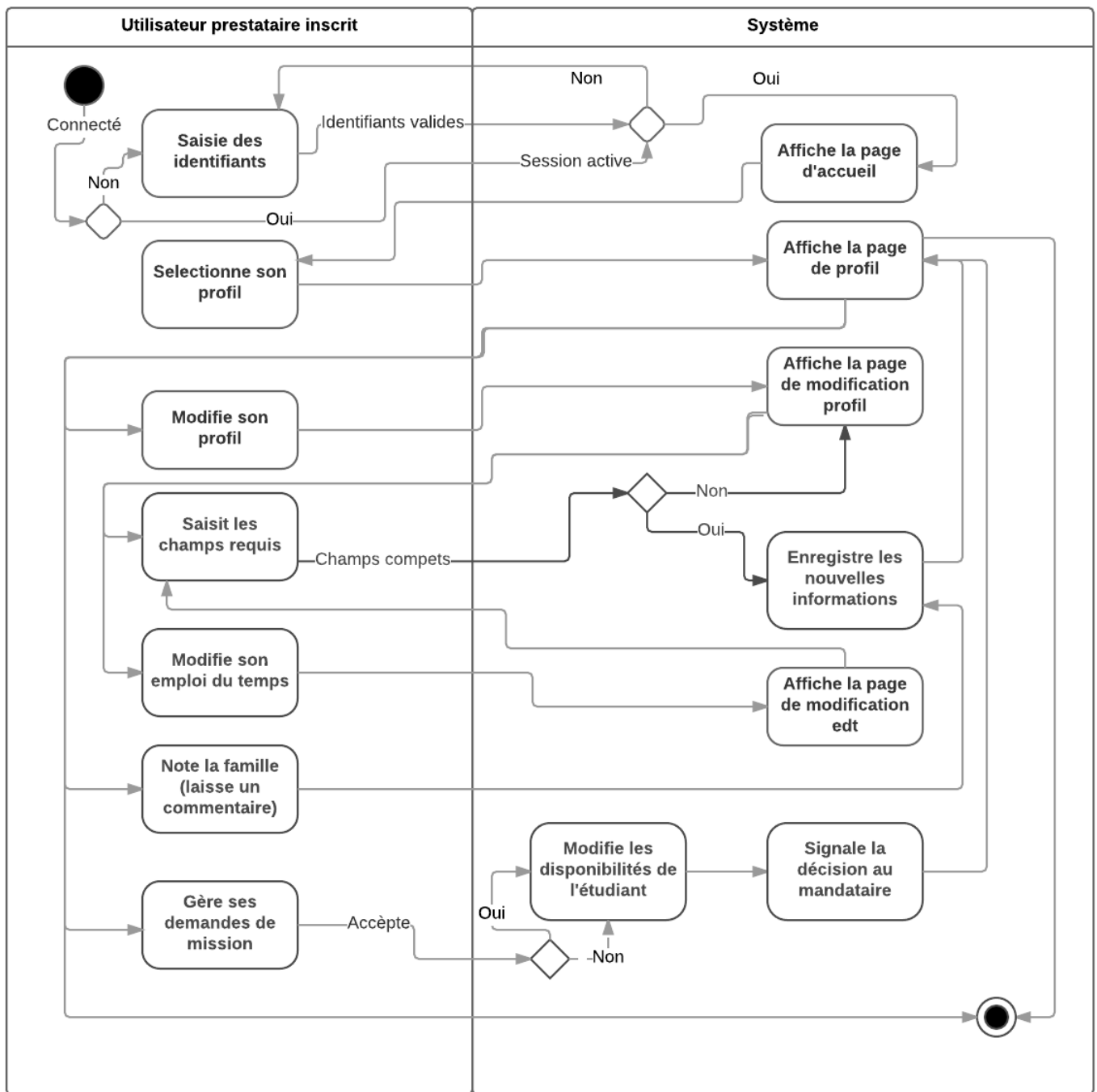


5 Réalisation

5.1 Diagramme d'activité pour les quatre types d'utilisateurs







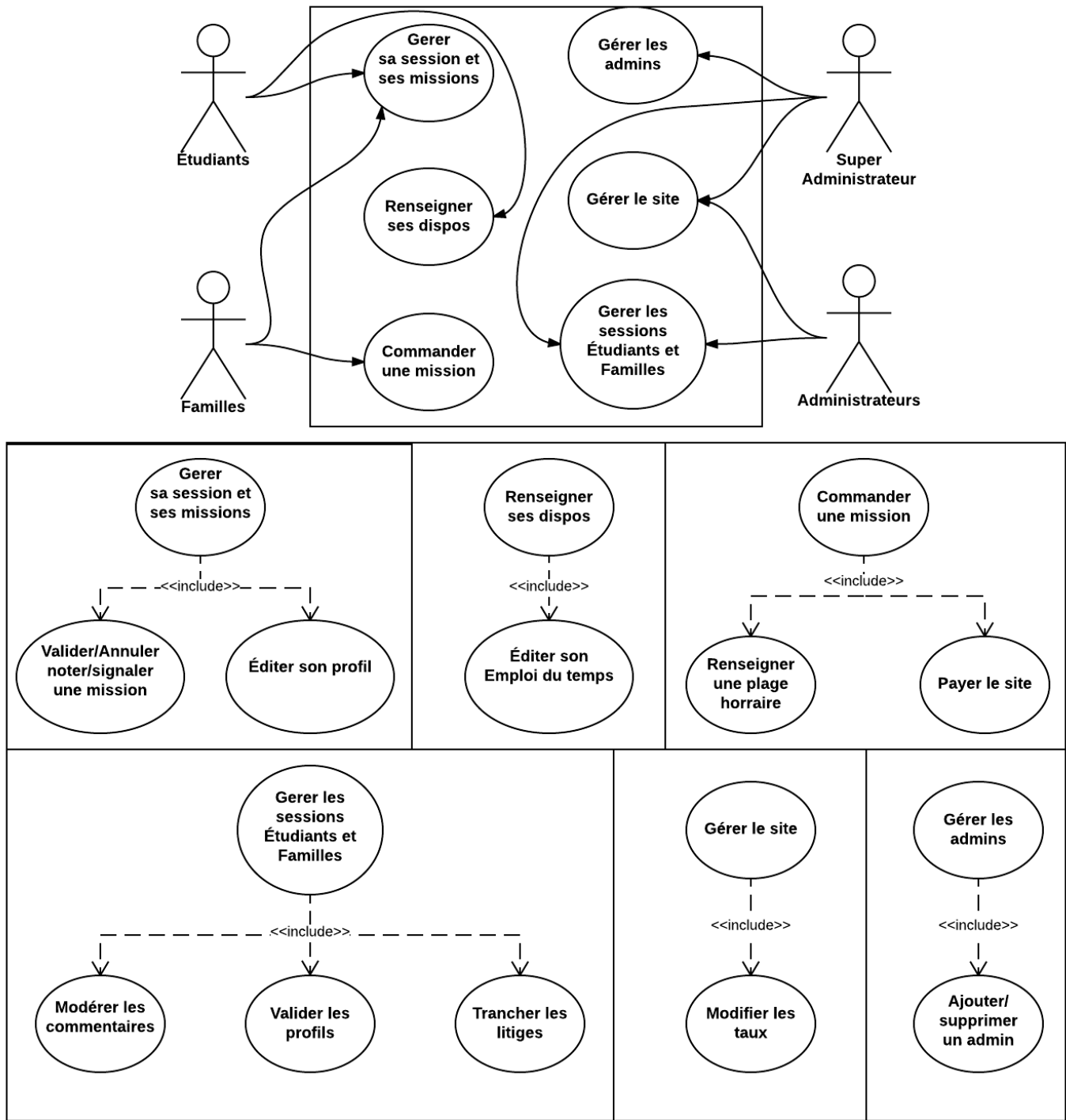


FIGURE 1 – Utilisateurs et CU

5.2 Tests fonctionnels

Nous avons pu tester tous nos cas, étant donné que notre site ne met pas en jeu des vies et n'implique pas d'erreurs lourdes de conséquences, le tableau des tests en annexe suffit à couvrir l'étendue des cas problématiques du site. Le fait que nous soyons six pour ce projet nous a permis de nous répartir la tâche de chercher les failles et nous a permis de couvrir plus consciencieusement les différents cas et scénarios.

6 Retrospective

Si le projet était à refaire nous repasserions en revue et avec plus de précision et d'attention les points suivants, qui, à terme auraient pu être mieux traités et auraient pu permettre de rendre le projet plus clair et réutilisable.

Production du site

- ***Les yeux plus gros que le ventre :***
Dans une perspective de produire un produit fini et complet nous aurions pu attacher plus d'importance à l'analyse et à l'étude du sujet pour limiter notre travail à un produit fonctionnel que nous aurions pu par la suite mieux améliorer. Nous avons produit un travail complet fonctionnel et avons pu rajouter quelques fonctions non demandées mais en contre partie, les finitions ne sont pas celle que nous espérions.
- ***Plus de fonctions, moins de scripts :***
Nous avons majoritairement codés des scripts avec peu de fonctions à part entières. Nous pensons qu'avec plus de temps et dans une optique différente il aurait été possible de produire un ensemble de scripts plus organisés (moins nombreux, plus précis dans leurs fonctions).
- ***Plus d'anglais moins de 'fran-glais' ou pourquoi pas en français ? :***
Nous sommes conscients de l'importance que doit être attribuée à la compréhension du code pour permettre sa réutilisation d'une part et une possible implémentation future par d'autres personnes. C'est pour cela que si c'était à refaire, nous nous organiserions pour cadrer dès le début du projet de meilleurs bases en ce qui concerne cette question.
- ***Un deuxième médiateur :***
Cela aurait été pour le mieux, du moins en ce qui concerne un groupe de six.
- ***C'est quoi Git ? :***
Dans ce projet, nous avons d'abord commencés avec l'interface de partage Dropbox, la transition nécessaire à Git ne s'est faite que trop tardivement. L'apprentissage plus précoce des quelques lignes de commandes nécessaires à l'utilisation de GIT aurait pu apporter un plus au projet dès son commencement.

Logistique

- ***On se divise le projet en 6 :***
Pour ce projet nous avons dû travailler à six. Étant donnée nos localisations géographiques il a été difficile de se retrouver pour travailler. Si c'était à refaire nous nous organiserions des séances régulières pour monitorer l'avancement du travail et faciliter la communication dans le groupe.

conception

- ***Qui sont nos étudiants ? :***
Nous avons décidé de vérifier l'identité de nos étudiants en leur demandant de notifier leur numéro de CNI, si c'était à refaire nous tenterions de mettre à terme cette idée et à vérifier l'exactitude des CNI.
- ***Report de l'emploi du temps :***
Nous avons lors de l'écriture des scénarios pensé à mettre un bouton report pour faciliter l'utilisation du site pour les étudiants, comme nous avons travaillé à implémenter un emploi du temps en Java nous n'avons pas eu le temps de chercher à intégrer cette fonctionnalité.
- ***Mot de passe oublié :***
En relisant nos spécifications fonctionnelles nous avons remarqué que la fonction mot de passe oubliée avait ironiquement été oubliée.
- ***Des plages horaires plus lisibles :***
Nous avons décidé de permettre à un utilisateur de choisir ses disponibilités à la demi heure pour faciliter l'accès aux étudiants par les familles. Ce faisant nous nous sommes rendu compte que les créneaux n'étaient pas concaténés et apparaissent demi heure par demi heure. Pour faciliter l'usage du site (validation de profil) nous aurions avec plus de temps peaufiné l'affichage des créneaux 'tout en un'.

6.1 Captures Esitting



(a) Accueil

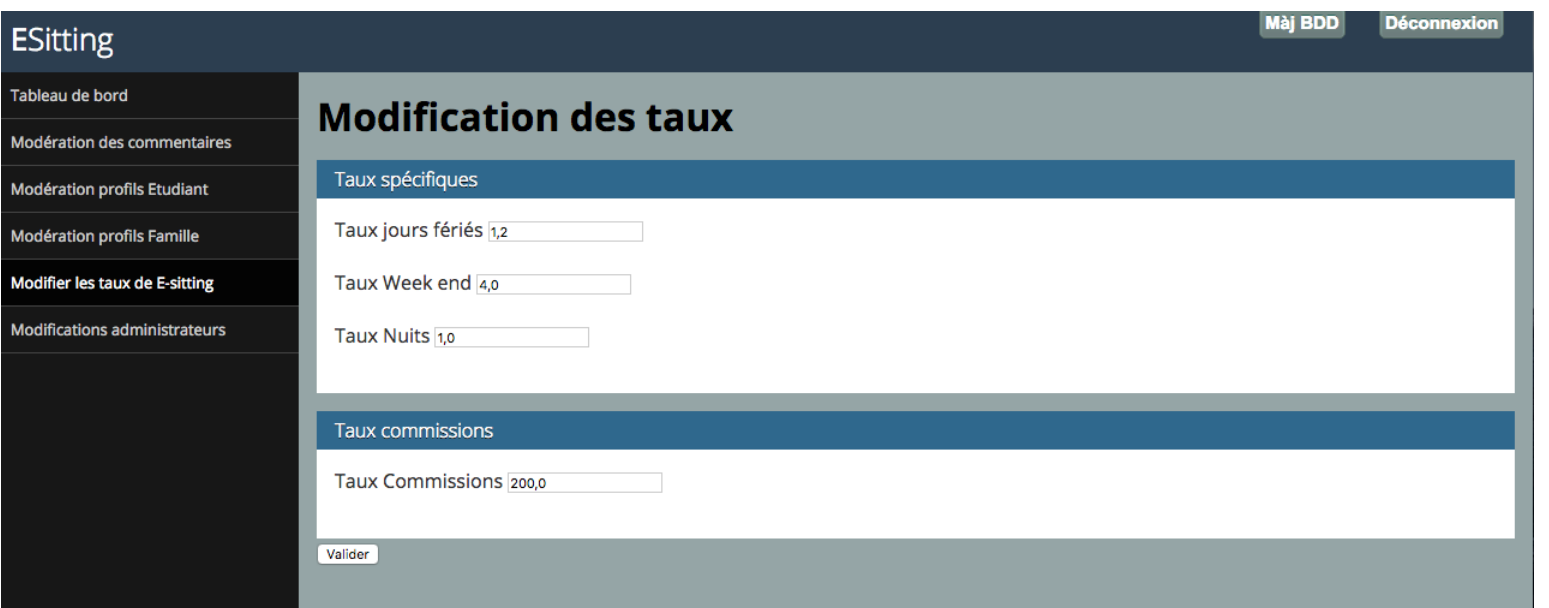


(b) Accueil Déconnecté

FIGURE 2 – Accueil : E-sitting



(a) Tableau de bord des administrateurs : Accueil



(b) Tableau de bord des administrateurs : Gestion des taux

FIGURE 3 – Tableau de bord des administrateurs E-sitting

Profil: 43



Image data 3861127505654601479 User ID 43 User type Etudiant

Nom Pivert Prenom Woody Date de naissance 01 / 01 / 1956

CP 06300 Téléphone 654654

Adresse rue du piquet

Tarif 18

Permis arbre CNI 654654

Etudes test

Mini-bio blabia

Valider Annuler

(a) Tableau de bord des administrateurs : Gestion profils

E-sitting

Tableau de bord

Modération des commentaires

Modération profils Etudiant

Modération profils Famille

Modifier les taux de E-sitting

Modifications administrateurs

Màj BDD Déconnexion

Modification des administrateurs

Ajouter un administrateur

Adresse e-mail: Entrez une adresse mail

Mot de passe: Entrez un mot de passe

Répéter le mot de passe: Retapez le mot de passe

Valider

Supprimer un administrateur

Adresse e-mail: Entrez une adresse mail

Supprimer

Administrateurs enregistrés

huber@huber.com

(b) Tableau de bord des administrateurs : Gestion des administrateurs

FIGURE 4 – Tableau de bord des administrateurs E-sitting - suite



E-Sitting

CONNEXION

tuchesftw@mail.com

.....

Valider

Mot de passe incorrect

OK

(a) Connexion Echouée : Mauvais mot de passe



E-Sitting

CONNEXION

mauvais_identifiant

.....

Valider

Membre non reconnu...

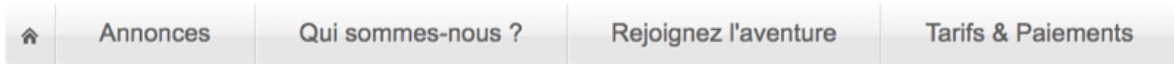
OK

(b) Connexion Echouée : Mauvais identifiant

FIGURE 5 – Connexion E-sitting



E-Sitting



pedro@pedro.com

CONNEXION

Valider

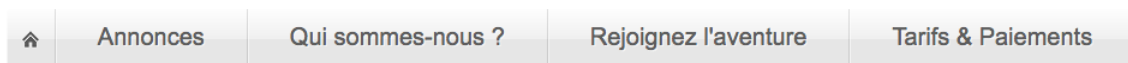
Profil en attente de validation

OK

(a) Connexion Echouée : Profil en attente de modération



E-Sitting



Souhaitez-vous vous inscrire en tant qu'étudiant ou en tant que famille?

Etudiant

Famille

(b) Inscription : Premier choix

FIGURE 6 – Connexion E-sitting - suite

[Annonces](#)
[Qui sommes-nous ?](#)
[Rejoignez l'aventure](#)
[Tarifs & Paiements](#)

[Etudiant](#)
[Famille](#)

INSCRIPTION FAMILLE

LES CHAMPS SUIVIS D'UNE ETOILE (*) SONT OBLIGATOIRES

NOM*
 E-mail*
 Mot de passe*
 Confirmation du mot de passe*
 Code Postal
 Adresse*
 Téléphone*
 Nombre d'enfants

Description personnelle

Image :

[Choisissez un fichier](#)
[Aucun fichier choisi](#)

J'accepte les conditions d'utilisateur.

[Valider](#)

(a) Inscription : Famille

[Annonces](#)
[Qui sommes-nous ?](#)
[Rejoignez l'aventure](#)
[Tarifs & Paiements](#)

[Etudiant](#)
[Famille](#)

INSCRIPTION ETUDIANTS

LES CHAMPS SUIVIS D'UNE ETOILE (*) SONT OBLIGATOIRES

NOM*
 Prénom*
 E-mail*
 Mot de passe*
 Confirmation du mot de passe*
 Date de naissance* jour / mo / année
 Code Postal
 Adresse*
 Téléphone*
 Pib.
 Tax
 Numéro de carte d'is.
 Niveau d'études
 Véhicule

Description personnelle

Image* :

[Choisissez un fichier](#)
[Aucun fichier choisi](#)

J'accepte les conditions d'utilisateur.

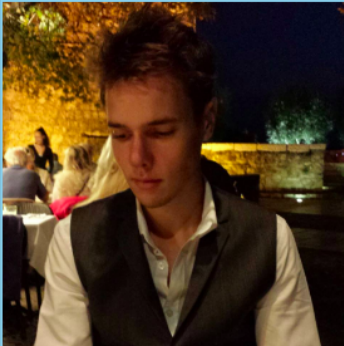
[Valider](#)

(b) Inscription : Étudiant

FIGURE 7 – Inscription E-sitting - suite



MON PROFIL
[Modifier mon profil](#)



TAUX DE VALIDATION :

92%

INFORMATIONS GÉNÉRALES

Nom : pedro
Prénom : laloutre
Date de naissance : 1991-07-29

Email : pedro@pedro.com
Adresse : 25 Bd Carabacel 06000
Téléphone : 0606060606

Etudes : 3D drugs modelisation
Type de permis: B
Tarif : 15€/h

Description

Dans le pain, je préfère la crouste à la mie.

NOTES ET COMMENTAIRES

Marley :
Étudiant relax, super mon 'pote'
5/5

blabla GIUL :
noms d'oiseaux ***autruche***
3/5

Hubert Grunig :
Super! je vous le recommande
4/5

[Signaler](#)

(a) Visite profil personnel : Informations ; commentaire à signaler

Contrats en attente de réponse :

Vous avez un contrat en attente de validation avec la famille : Grunig à : 10:00:00 le : 2016-12-06

[Accepter](#) [Refuser](#)

[Voir le profil](#)

Mes Contrats en cours :

Vous avez un contrat validé avec la famille : Marley à : 08:00:00 le : 2016-12-04

[Annuler](#)

Mes derniers contrats effectués :

Vous avez un contrat passé avec la famille: Pirate à : 08:30:00 le : 2016-12-04

[Evaluer la prestation](#)

(b) Visite profil personnel - suite : missions

FIGURE 8 – E-sitting - suite



NOTATION D'UN CONTRAT

- 1
- 2
- 3
- ☒ 4
- 5

C'était super, la famille est super sympa

Valider

(a) Notation : Famille, étudiants

December, 2016						
M	T	W	T	F	S	S
			1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30	31	

Available Slots

The following slots are available on 02-12-2016

Les horaires affichés en **bleu** correspondent à ceux que vous avez déjà réservés. Les horaires affichés en **orange** correspondent à ceux réservés par une famille en attente de validation. Pour libérer un créneau il suffit de le cocher et de sauver la modification.

Start	End	Price	Book
08:00:00	08:30:00	€0.00	<input type="checkbox"/>
08:30:00	09:00:00	€0.00	<input type="checkbox"/>
09:00:00	09:30:00	€0.00	<input type="checkbox"/>
09:30:00	10:00:00	€0.00	<input type="checkbox"/>
10:00:00	10:30:00	€0.00	<input checked="" type="checkbox"/>
10:30:00	11:00:00	€0.00	<input checked="" type="checkbox"/>
11:00:00	11:30:00	€0.00	<input checked="" type="checkbox"/>
11:30:00	12:00:00	€0.00	<input checked="" type="checkbox"/>
12:00:00	12:30:00	€0.00	<input checked="" type="checkbox"/>
12:30:00	13:00:00	€0.00	<input type="checkbox"/>
13:00:00	13:30:00	€0.00	<input type="checkbox"/>
13:30:00	14:00:00	€0.00	<input type="checkbox"/>
14:00:00	14:30:00	€0.00	<input type="checkbox"/>
14:30:00	15:00:00	€0.00	<input type="checkbox"/>
15:00:00	15:30:00	€0.00	<input type="checkbox"/>
15:30:00	16:00:00	€0.00	<input type="checkbox"/>

Selected Slots

10:30:00 - 11:00:00
11:00:00 - 11:30:00
11:30:00 - 12:00:00
10:00:00 - 10:30:00
12:00:00 - 12:30:00

Login

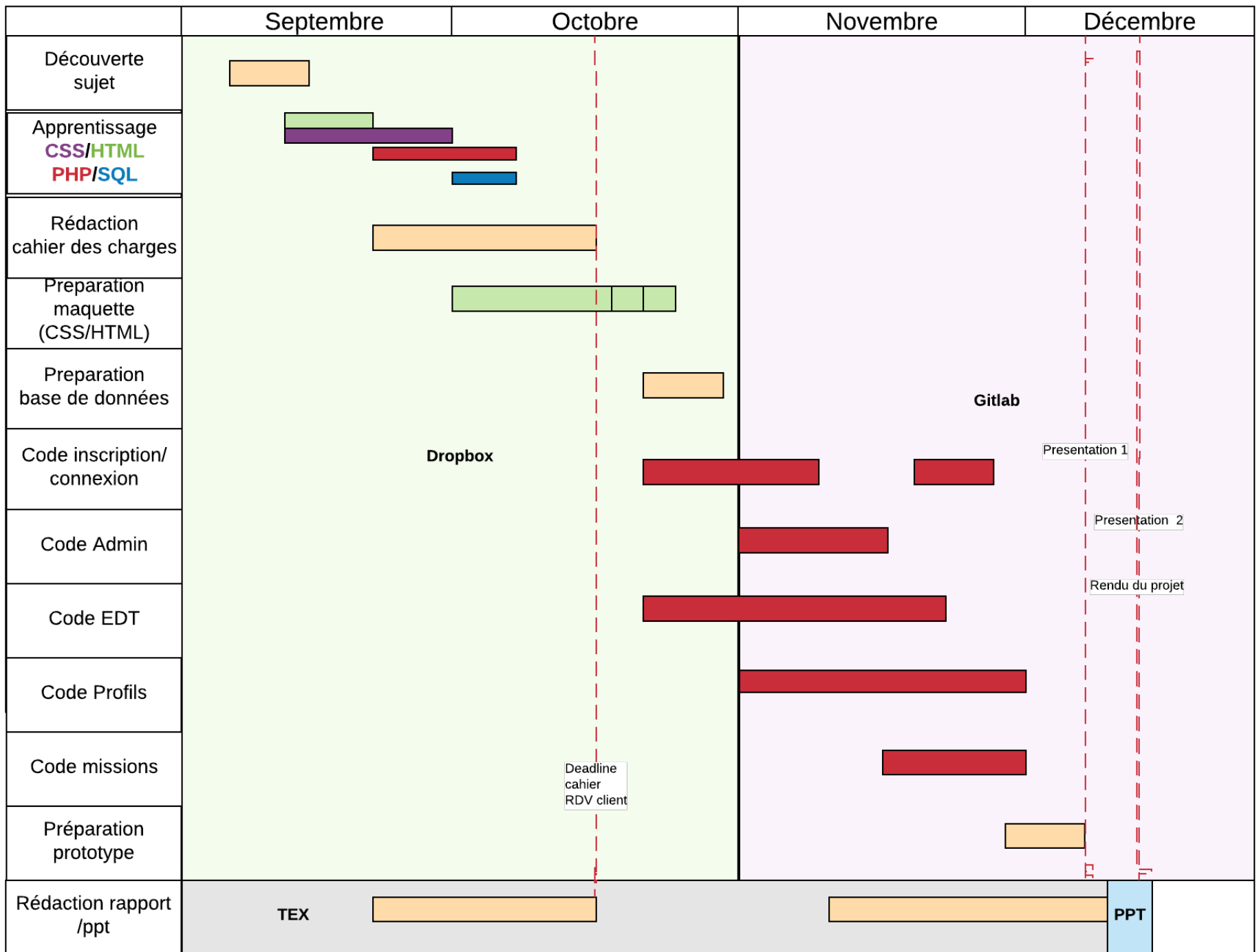
mark@mark.com

Password

Sauver

(b) Calendrier de disponibilité étudiant

FIGURE 9 – E-sitting - suite



B Annexe Code php : Édition du profil

```
<!-- On récupère les anciennes données enregistrées dans la base de données correspondantes à l'utilisateur -->
<?php include("BDProfilUtilisateur.php");
autologout();
?>
<DOCTYPE html>
<html>

    <head>
        <meta charset="utf-8" />
        <link rel="stylesheet" href="style_inscription.css" />
        <link rel="stylesheet" href="style_general.css" />
        <link rel="stylesheet" href="style_general_connect.css" />
        <title>E-Sitting - Le babysitting sans bouger de chez vous</title>
    </head>
    <body>

        <div id = "bloc_page">
            <header>
                <div id="titre_principal">
                    <div id="logo">
                        
                    </div>
                </div>
            </div>
            <div id="deconnexion">
                <a href="logout.php" class="myButtonConnected">Déconnexion</a>
            </div>
            <div id="bonjour"><?php
                if ($_SESSION['type']=='Super' or $_SESSION['type']=='Admin')
                {
                    echo( " <a href='dash_admin.php' class='myButtonConnected'>Dashboard". " </a>" );
                }
                else
                {
                    echo("<a href='#' class='myButtonConnected'>Bonjour ". $_SESSION['prenom'] . " ". $_SESSION['nom'] . " </a>");
                }?></div>
            </div>
        </header>
        <div id = "Menu_Principal">
            <ul class="fancyNav" id="myTopnav">
                <li id="home"><a href="AccueilConnecte.php" class="homeIcon">Accueil</a></li>
                <li><a href="#">Annonces</a></li>
                <li><a href="AccueilConnecte.php#Whoarewe">Qui sommes-nous ?</a></li>
                <li><a href="AccueilConnecte.php#Esitter">Rejoignez l'aventure</a></li>
                <li><a href="AccueilConnecte.php#Paiements">Tarifs & Paiements</a></li>
                <li><a href="ConsultationProfilUtilisateur.php">Profil</a></li>
                <li class="icon">
                    <a href="javascript:void(0);" style="font-size:15px;" onclick="myFunction()">?</a>
                    <!-- Initialisation du bouton qui apparait en cas de trop petite image (voir media query) -->
                </li>
            </ul>
        </div>
        <script>
            function myFunction() {
                var x = document.getElementById("myTopnav");
                if (x.className === "fancyNav") {
                    x.className += " responsive";
                } else {
                    x.className = "fancyNav";
                }
            }
        </script>
    </body>
</html>
```

```

    }
}
</script>
<section>
    <article>

        <!-- On gère deux cas différents en fonction du type d'utilisateur car les infos
        ne sont pas les mêmes -->
        <?php
        if($type_utilisateur=='Etudiant')
        { ?>

            <h1>Modification de profil étudiant</h1>
            <a class="EmploiDuTemps" href="Calendrier/calendar_etudiant.php">Mon emploi du temps</a>

<div id="formDiv">
    <form method="post" action="ModificationBDDProfilEtudiant.php" enctype="multipart/form-data"><br/>
    <h2>Les champs suivis d'une étoile (*) sont obligatoires</h2>
    <input type="text" name="NomEtudiant" placeholder="NOM*" <?php echo 'value=' . '\'' .
    $ancien_nom . '\'' ;?> size="30" maxlength="20"/> <br/><br/>
    <input type="text" name="PrenomEtudiant" placeholder="Prenom*" <?php echo 'value=' . '\'' .
    $ancien_prenom . '\'' ;?> size="30" maxlength="20"/> <br/><br/>
    <input type="text" name="MailEtudiant" placeholder="E-mail*" <?php echo 'value=' . '\'' .
    $ancien_email . '\'' ;?> size="30" maxlength="30"/> <br/><br/>
    <input type="password" name="PasswordEtudiant" placeholder="Mot de passe*"
    size="30" maxlength="30"/> <br/><br/>
    <input type="password" name="PasswordConfirmationEtudiant"
    placeholder="Confirmation du mot de passe*" size="30" maxlength="30"/> <br/><br/>
    <label>Date de naissance*</label>
    <input type="number" name="JourNaissance" placeholder="Jour" <?php echo 'value=' . '\'' .
    $ancien_jourNaissance . '\'' ;?> min="1" max="31"/>
    <label></label>
    <input type="number" name="MoisNaissance" placeholder="Mois" <?php echo 'value=' . '\'' .
    $ancien_moisNaissance . '\'' ;?> min="1" max="12"/>
    <label></label>
    <input type="number" name="AnneeNaissance" placeholder="Année" <?php echo 'value=' . '\'' .
    $ancien_anneeNaissance . '\'' ;?> min="1900" max="2016"/> <br/> <br/>
    <input type="text" name="CodePostal" placeholder="Code Postal*" <?php echo 'value=' . '\'' .
    $ancien_cp . '\'' ;?> size="10" maxlength="5"/> <br/><br/>
    <input type="text" name="AdresseEtudiant" placeholder="Adresse*" <?php echo 'value=' . '\'' .
    $ancien_adresse . '\'' ;?> size="30" maxlength="50"/> <br/><br/>
    <input type="number" name="TelephoneEtudiant" placeholder="Téléphone*" <?php echo 'value=' .
    $ancien_telephone ;?> max="9999999999"/> <br/><br/>
    <input type="number" name="NumeroCNIetudiant" placeholder="Numéro de carte d'identité*"
    <?php echo 'value=' . '\'' . $ancien_cni . '\'' ;?> max="99999999999999"/> <br/><br/>
    <input type="text" name="EtudesEtudiant" placeholder="Niveau d'études" <?php
    //ce champs peut rester vide. On vérifie qu'il ne le soit pas avant de concaténer
    if($ancien_etudes) {
        echo 'value=' . '\'' . $ancien_etudes . '\'' ;
    }

    ?> size="30" maxlength="30"/> <br/><br/>
    <input type="text" name="VehiculeEtudiant" placeholder="Véhicule" <?php
    //ce champs peut rester vide. On vérifie qu'il ne le soit pas avant de concaténer
    if($ancien_permis) {
        echo 'value=' . '\'' . $ancien_permis . '\'' ;
    }
}

```

```

}
?> size="30" maxlength="30"/> <br/><br/>
<textarea name="DescriptionEtudiant" rows="10" cols="50" maxlength="750"><?php
//ce champs peut rester vide. On vérifie qu'il ne le soit pas avant de le remplir
if($ancien_description)
{
    echo $ancien_description ;
}
else
{
    echo 'Description personnelle';

} ?></textarea><br/><br/>
<label>Image :</label><br/><br/>
<input type="hidden" name="oldImage" <?php echo 'value=' . '\'' . $ancien_image . '\'' ;?>
<input type="file" name="image" /><br/><br/>
<input type="submit" name="submit" /> <br/><br/>
</form>
</div>

<?php
}
elseif($type_utilisateur=='Famille')
{ ?>
    <h1>Modification de profil famille</h1>

<div id="formDiv">
    <form method="post" action="ModificationBDDProfilFamille.php" enctype="multipart/form-data"><br/>
        <h2>Les champs suivis d'une étoile (*) sont obligatoires</h2>
        <input type="text" name="NomFamille" placeholder="NOM*" <?php echo 'value=' . '\'' . $ancien_nom . '\'' ;?> size="30" maxlength="20"/> <br/><br/>
        <input type="text" name="MailFamille" placeholder="E-mail*" <?php echo 'value=' . '\'' . $ancien_email . '\'' ;?> size="30" maxlength="30"/> <br/><br/>
        <input type="password" name="PasswordFamille"
placeholder="Mot de passe*" size="30" maxlength="30"/> <br/><br/>
        <input type="password" name="PasswordConfirmationFamille"
placeholder="Confirmation du mot de passe*" size="30" maxlength="30"/> <br/><br/>
        <input type="text" name="CodePostal" placeholder="Code Postal*" <?php echo 'value=' . '\'' . $ancien_cp . '\'' ;?> size="10" maxlength="5"/> <br/><br/>
        <input type="text" name="AdresseFamille" placeholder="Adresse*" <?php echo 'value=' . '\'' . $ancien_adresse . '\'' ;?> size="30" maxlength="50"/> <br/><br/>
        <input type="text" name="TelephoneFamille" placeholder="Téléphone*" <?php echo 'value=' . '\'' . $ancien_telephone . '\'' ;?> size="30" maxlength="10"/> <br/><br/>
        <input type="number" name="NombreEnfantsFamille" placeholder="Nombre d'enfants" <?php
//ce champs peut rester vide. On vérifie qu'il ne le soit pas avant de concaténer
if($ancien_enfants) {
            echo 'value=' . $ancien_enfants ;
        }
        ?> max="30"/> <br/><br/> <!-- A SURVEILLER -->
        <textarea name="DescriptionFamille" rows="10" cols="50" maxlength="750"><?php
//ce champs peut rester vide. On vérifie qu'il ne le soit pas avant de le remplir
if($ancien_description)
{
            echo $ancien_description ;
        }
        else
        {
            echo 'Description personnelle';

```



```
    } ?></textarea><br/><br/>
    <label>Image :</label><br/><br/>
    <input type="hidden" name="oldImage" <?php echo 'value=' . '\'' . $ancien_image . '\'' ;?>>
    <input type="file" name="image" /><br/><br/>
    <input type="submit" name="submit" /> <br/><br/>
</form>
</div>

    <?php
    }
    else
    {
        echo 'problème de type d\'utilisateur' ;
    }
    ?>

</article>
</section>

<footer>
</footer>
</div>
</body>
</html>
```