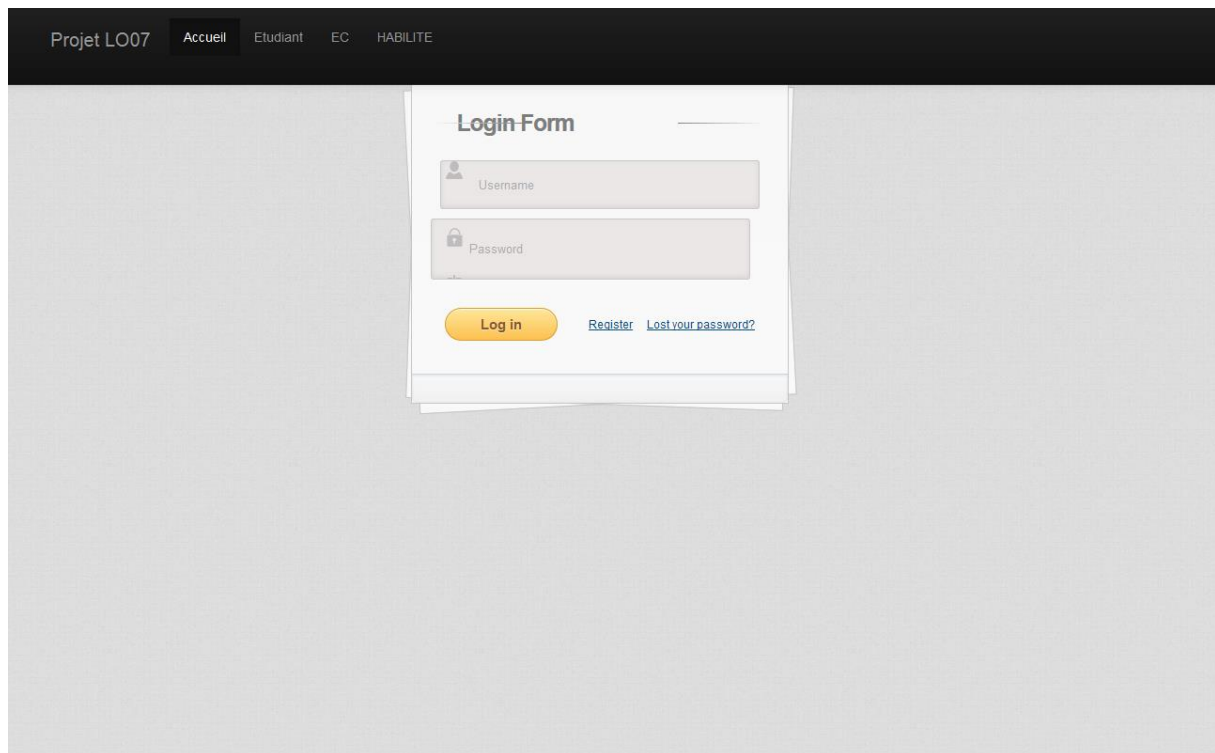


RAPPORT DE PROJET

PROJET LO07 : GESTION DE TUTEURS

Juin 2014



SOMMAIRE

Introduction	3
Présentation du sujet.....	3
Contexte.....	3
Exigences techniques du projet	3
Cahier des charges	4
Fonctionnalite DRH	4
Fonctionnalite scolarite.....	5
fonction Responsable programme	7
Elaboration.....	8
Itération 1 : ajout et modification d'un Etudiant.....	8
Itération 2 : Ajout et modification d'un ec	8
Itération 3 : ajout de tuteur	8
Itération 4 : Gestion du conseiller	8
Itération 5 : Gestion des habilitations	8
ITERATION 6 : AJOUT VIA CSV	8
Itération 7 : Identification	9
Organisation	9
Outils de travail	9
Planification finale du projet	9
Analyse et modèle de données.....	10
Modèle conceptuel de données.....	10
Analyse.....	11
Modele LOGIQUE	12
Modele physique de donnée	12
Choix d'un framework ?	13
Réalisation	14
Base de donnes	14
Aspect graphique	16
AJAX	16
PHP.....	18
Livrable.....	20
Difficultés rencontrées.....	20
Conclusion	21

INTRODUCTION

Ce document constitue le rapport écrit du projet de l'UV LO07 au printemps 2014 à réaliser en binôme.

Le binôme était constitué de :

- Yamine Larbi
- Hamza Obeidat

Le projet s'est étendu du 18 mars 2014 au 20 juin 2014.

PRESENTATION DU SUJET

Le sujet était imposé pour cet UV. Le sujet consiste en l'analyse, la conception et la réalisation d'un outil web pour la gestion des tuteurs des élèves de l'UTT. Cet outil doit être ergonomique, évolutif, et permettre de gérer le tuteur des étudiants tout au long de leurs scolarités.

CONTEXTE

A son entrée à l'UTT, chaque étudiant qu'il soit en Tronc Commun (TC) ou en Branche se voit attribué un conseiller, chargé de le conseiller tout au long de sa scolarité. Ce conseiller est l'un des enseignants-chercheurs de l'UTT. Pour le moment, tous les enseignants chercheurs quel que soit leur domaine de compétences sont habilités à conseiller n'importe quel étudiant. Dans la pratique, il serait préférable que chaque enseignant-chercheur soit habilité à conseiller les étudiants de Tronc Commun, puisque un responsable de programme désigne les enseignants aptes à conseiller les étudiants de son programme.

L'objectif de ce projet est de réaliser un outil web permettant cette gestion des conseillers tout au long de la scolarité.

EXIGENCES TECHNIQUES DU PROJET

Les caractéristiques techniques du projet étaient imposées: nous devons utiliser le langage PHP pour le développement des pages web dynamiques, un serveur Apache, des pages HTML, des feuilles de style CSS, des instructions Javascript et l'utilisation du logiciel MySQL pour le serveur de base de données était grandement conseillé.

L'utilisation d'un Framework était possible et devait être expliqué.

CAHIER DES CHARGES

Le cahier des charges était détaillé dans l'intitulé du projet. Celui-ci était assez clair, et chaque tâche était déjà découpée de manière algorithmique. Nous allons maintenant détailler les actions que doivent pouvoir réaliser les différents acteurs du projet.

FONCTIONNALITE DRH

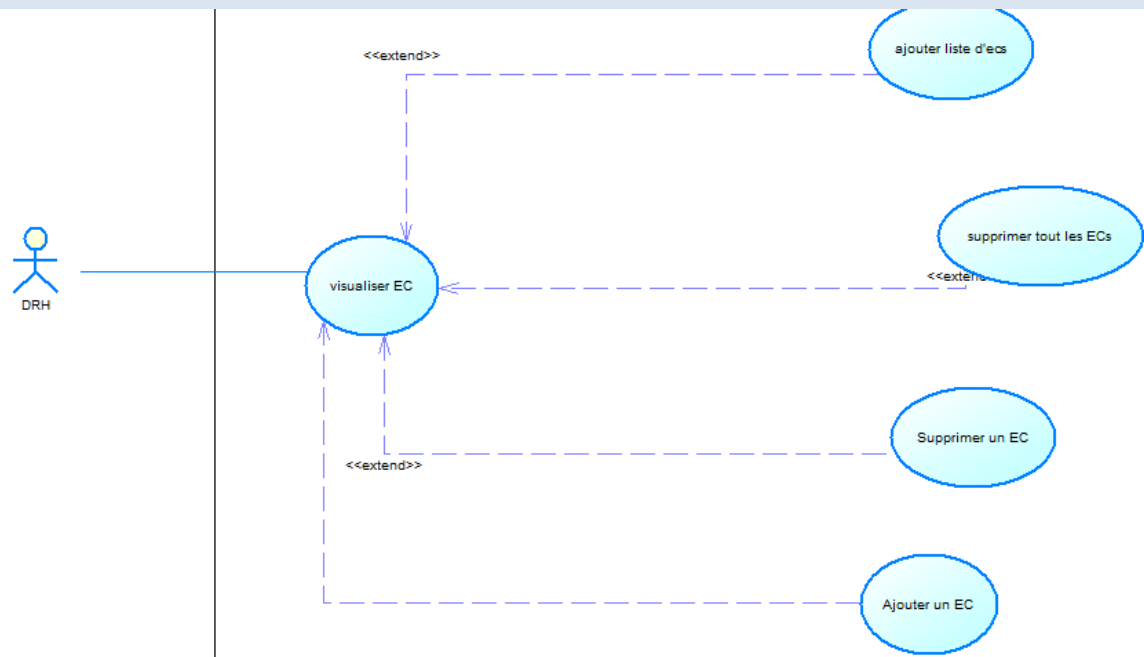


Figure 1 : Cas d'utilisation DRH

- visualiser EC :
 - Le DRH doit pouvoir visualiser l'ensemble des enseignants chercheurs avec le nombre d'étudiant qu'ils conseillent, et la liste de ses étudiants conseillé.
- Ajouter liste CSV :
 - Le DRH doit pouvoir ajouter une liste d'enseignant-chercheurs contenu dans un fichier CSV de ce type :
 - `prenom;nom;bureau;pole`
`marc;lemercier;T122;HETIC`
`alain;corpel;T111;HETIC`
`aurelien;benel;T107;HETIC`
`babiga;birregah;H107;ROSAS`
- Supprimer tous les ECs :
 - Le DRH doit pouvoir supprimer l'ensemble des enseignants chercheurs, ce qui entraine la suppression de leurs liens avec d'éventuels étudiants conseillés.

- Supprimer un EC :
 - Le DRH doit pouvoir supprimer un EC en particulier. Tous les étudiants qui étaient conseillé par cet EC se retrouvent alors sans tuteurs.
- Ajouter un EC :
 - Le DRH doit pouvoir un ajouter un EC à la base de données. Une vérification sur le nom et le prénom de l'EC est effectué.

FONCTIONNALITE SCOLARITE

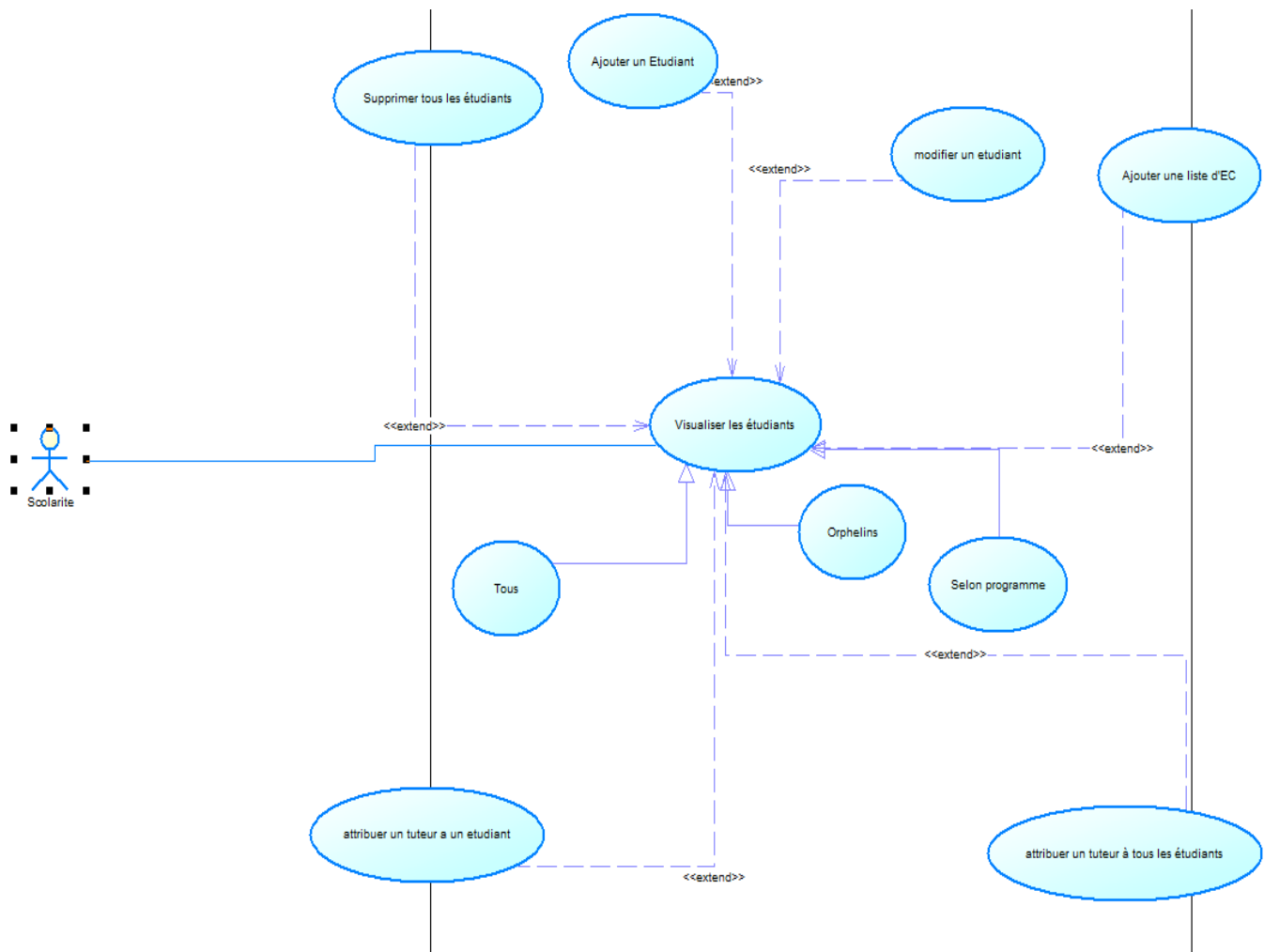


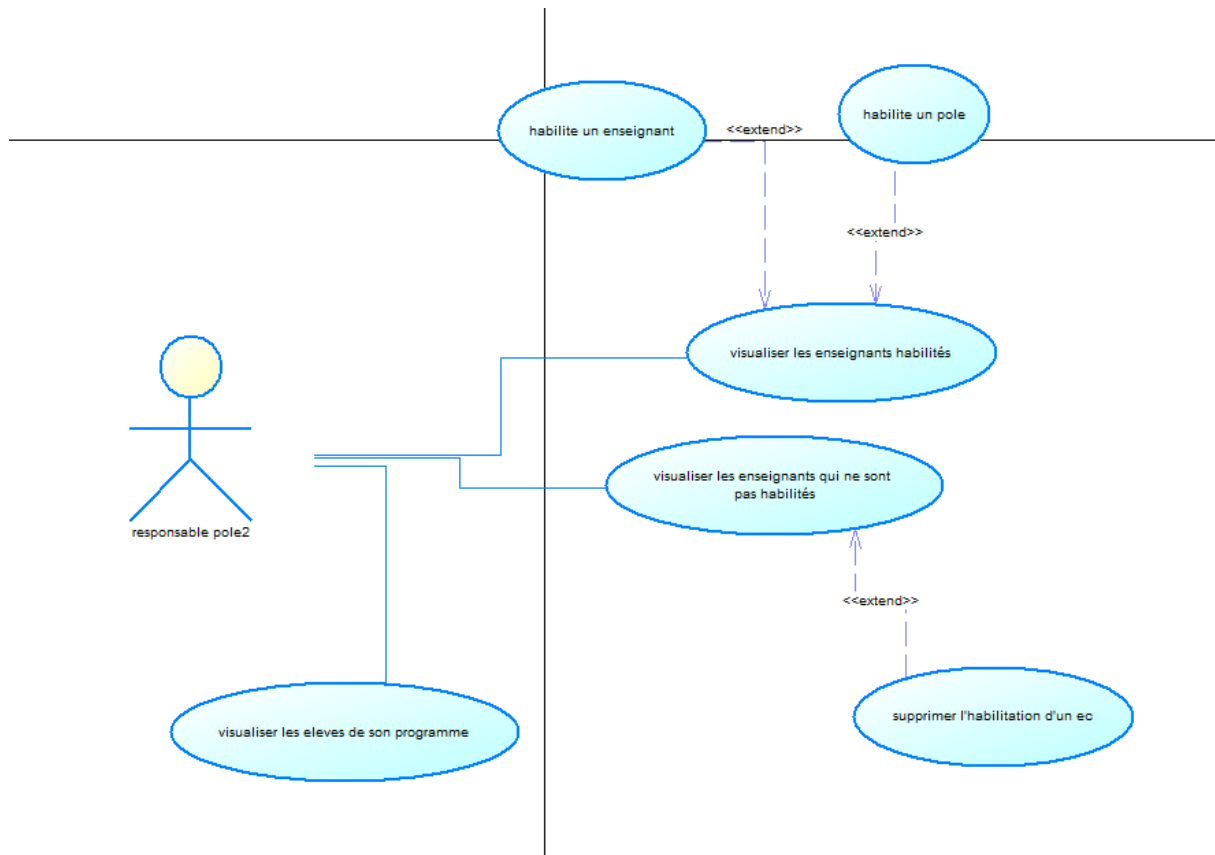
Figure 2 Cas d'utilisation Scolarité

- Visualiser les étudiants :
 - Le service scolarité doit pouvoir visualiser l'ensemble des étudiants avec leurs tuteurs. Le service scolarité doit pouvoir visualiser :
 - Tous les étudiants :
 - Tous les étudiants s'affichent avec leur tuteur respectif.
 - Tous les étudiants d'un programme :

- Tous les étudiants du programme sélectionné s'affichent avec leur tuteur respectif.
- Tous les étudiants orphelins :
 - Tous les étudiants qui n'ont pas de tuteur.
- Supprimer tous les étudiants :
 - Le service scolarité doit pouvoir supprimer l'ensemble des étudiants.
- Supprimer un étudiant :
 - Le service scolarité doit pouvoir supprimer un étudiant spécifique.
- Ajouter un étudiant :
 - Le service scolarité doit pouvoir ajouter un étudiant. Une vérification sera faite sur le nom pour ne pas insérer de doublon.
- Ajouter une liste d'étudiants :
 - Le service scolarité doit pouvoir ajouter une liste d'étudiant au format CSV de ce type :

```
numéro;prénom;nom;programme;semestre  
12345;aurelien;duchemin;ISI;3  
12346;Martin;matin;SM;3  
12347;Jacques;Brel;ISI;3
```
- Modifier un étudiant :
 - Le service scolarité doit pouvoir modifier un étudiant. Si l'étudiant change de branche, une vérification est faite pour vérifier si son tuteur est habilité à conseiller dans cette branche, si ce n'est pas le cas un nouveau conseiller lui est attribué.
- Attribuer un conseiller à un étudiant:
 - Un conseiller est attribué à l'étudiant spécifié. Ce conseiller est habilité à conseiller dans le programme de l'étudiant et est celui conseillant le moins d'étudiant.
- Attribuer un conseiller à tous les étudiants :
 - Un conseiller est attribué à tous les étudiants n'ayant pas de conseiller.

FONCTION RESPONSABLE PROGRAMME



- Visualiser les enseignants habilités :
 - Le responsable du programme doit pouvoir visualiser les enseignants habilités à conseiller pour son programme.
- Supprimer l'habilitation d'un enseignant :
 - Le responsable du programme doit pouvoir supprimer l'habilitation d'un enseignant à conseiller les étudiants de son programme.
- Habilité un pôle :
 - Le responsable du programme doit pouvoir ajouter l'ensemble des enseignants d'un pôle à la liste des enseignants habilités à conseiller dans son programme.
- Visualiser les enseignants non-habilités :
 - Le responsable du programme doit pouvoir visualiser les enseignants qui ne sont pas habilités à conseiller pour son programme.
- Ajouter l'habilitation à un enseignant :
 - Le responsable du programme doit pouvoir ajouter l'habilitation à conseiller pour son programme à un étudiant.

- Visualiser la liste des élèves de son programme :
 - Le responsable du programme doit pouvoir visualiser la liste des élèves de son programme avec leur enseignant chercheur.

ELABORATION

Après avoir élaboré le cahier des charges, nous avons décidé de travailler en mode itératif incrémental, c'est-à-dire qu'à chaque fois, nous créons une application fonctionnelle, et ajoutons petit à petit chacune des fonctionnalités. Les sept itérations prévues sont décrites ci-dessous :

ITERATION 1 : AJOUT ET MODIFICATION D'UN ETUDIANT

Dans cette première itération du projet, l'objectif était de pouvoir ajouter un étudiant dans la base, afficher le formulaire de modification, et supprimer un étudiant.

ITERATION 2 : AJOUT ET MODIFICATION D'UN EC

La deuxième étape du projet consistait en la création et la modification d'un enseignant-chercheur, tout comme pour l'étudiant.

ITERATION 3 : AJOUT DE TUTEUR

La troisième étape du projet concernait la gestion du conseiller de l'étudiant. L'objectif de l'étape est de permettre l'ajout d'un tuteur selon les différentes exigences du cahier des charges et la possibilité de modifier la branche de l'étudiant.

ITERATION 4 : GESTION DU CONSEILLER

Cette étape consistait à finir les différentes fonctionnalités d'un conseiller. Possibilité de visualiser ses étudiants conseillés, suppression du lien des étudiants conseillés en cas de suppression de l'enseignant chercheur etc.

ITERATION 5 : GESTION DES HABILITATIONS

Cette cinquième étape consiste en la gestion des habilitations, avec les possibilités d'ajout/suppression des habilitations selon les exigences du cahier des charges.

ITERATION 6 : AJOUT VIA CSV

Cette étape visait à permettre l'ajout d'enseignant chercheurs et/ou d'élèves via des fichiers csv.

ITERATION 7 : IDENTIFICATION

Cette dernière étape consiste en l'ajout des méthodes d'identification afin de permettre aux différents rôles de n'accéder qu'aux différentes choses auxquelles ils ont le droit d'accéder. Comme exigé, cette fonctionnalité se basera sur des variables de session. Les identifiants seront stockés dans la base de données.

ORGANISATION

OUTILS DE TRAVAIL

GESTIONNAIRE DE VERSIONS

La nécessité de travailler à l'aide d'un outil de gestion de version (VCS) nous est apparue immédiatement. Nous avons fait le choix d'utiliser l'outil décentralisé Git, via le serveur Gitis.

L'intérêt d'un outil de gestion de versions par rapport à un outil de partage comme Dropbox est la possibilité de travailler en simultané sans risquer d'écraser les modifications apportées par son binôme, avoir un historique des différentes modifications etc.

GESTIONNAIRE DE PROJET

Nous avons installé et configuré l'outil Redmine pour planifier et gérer le projet. Ce logiciel en ligne permet d'assigner une ou plusieurs tâches à une ressource, de planifier les différentes étapes d'un projet et d'obtenir de la visibilité sur son avancement.

Si les délais prévus pour l'accomplissement des tâches n'ont pas toujours été respectés, nous considérons malgré tout qu'un tel outil de gestion de projet est fort pratique afin de bien délimiter le projet et gérer son emploi du temps.

PLANIFICATION FINALE DU PROJET

Une fois le prototypage terminé et l'environnement de travail fixé, le développement du projet a été planifié en trois itérations majeures :

- Itération 1 : Gestion des étudiants
- Itération 2 : Gestion des enseignants-chercheurs.
- Itération 3 : Gestion des habilitations.

ANALYSE ET MODELE DE DONNEES

L'analyse et la conception du MCD fût relativement rapide, le nombre de table n'étant pas très élevés et les relations entre elle étant explicités dans l'énoncé.

MODELE CONCEPTUEL DE DONNEES

Le logiciel utilisé pour la modélisation et la génération du script de création de la base de données est PowerDesigner dans sa version 16. Le Modèle Conceptuel de Données définitif du projet contient 5 entités et 4 associations.

Au total, la base de données compte 26 champs répartis dans 6 tables.

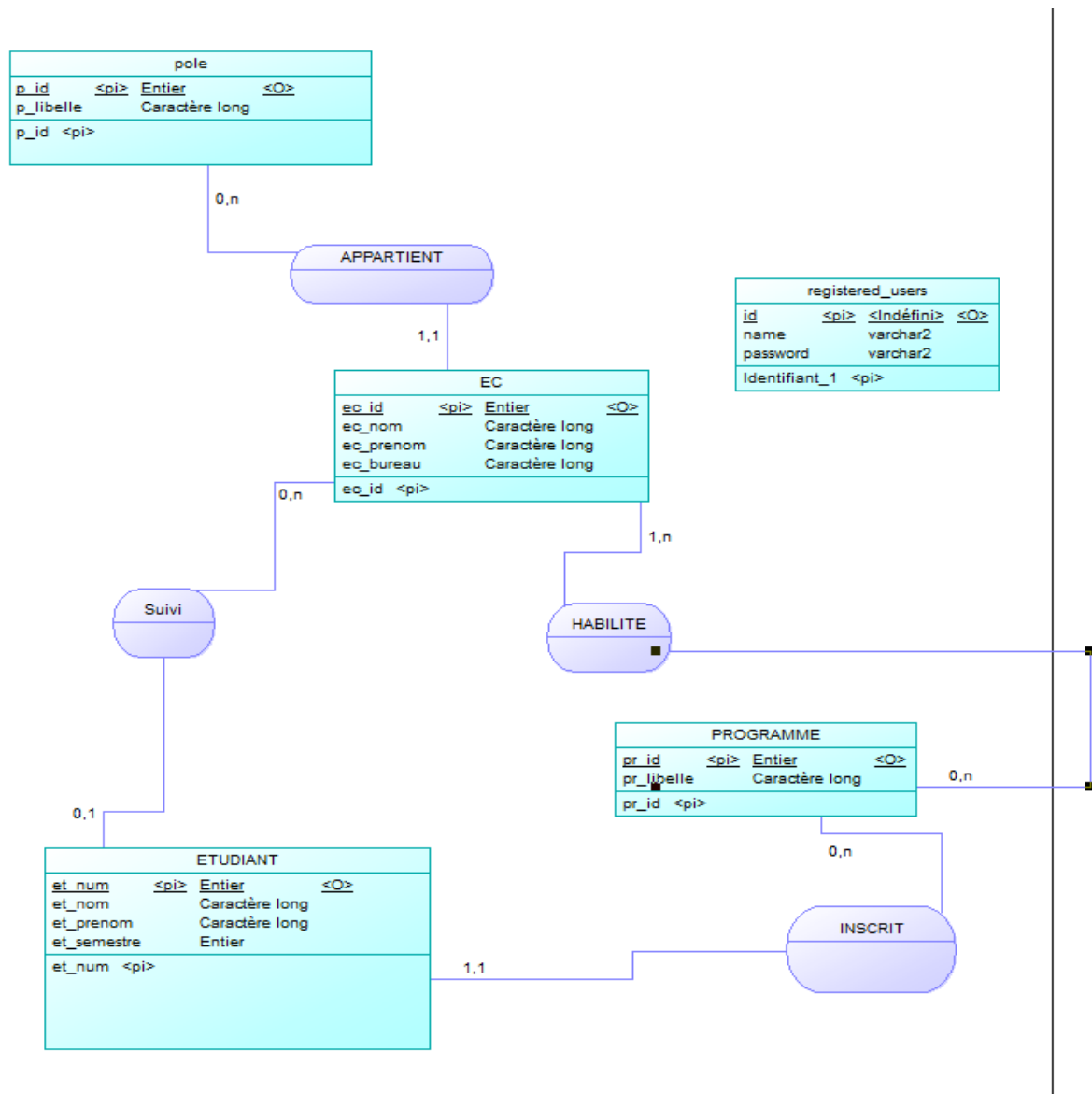


Figure 3 Modèle conceptuel de donnée

ANALYSE

- L'entité EC représente les enseignants chercheurs. Sa clef primaire est ec_id.
- L'entité ETUDIANT représente les étudiants. Sa clef primaire est et_num.
- L'entité PROGRAMME représente les différents programmes de l'UTT (ISI, SI, SM ...). Sa clef primaire est pr_id.
- L'entité POLE représente les pôles auxquels peuvent appartenir un enseignant chercheur.
- L'entité Registered_Users représente les utilisateurs du site. Sa clef primaire est id.
- L'association inscrit entre Etudiant et Programme caractérise le programme de l'étudiant. Un étudiant peut être inscrit qu'à un et un seul programme en même temps. Les programmes peuvent avoir aucun ou plusieurs étudiants.
- L'association habilite entre EC et PROGRAMME caractérise les programmes que l'enseignant chercheur est habilité à conseiller. Un enseignant chercheur est habilité à conseiller soit 1 (puisque tous les ECs sont apte à conseiller les TC) ou plusieurs programmes.
- L'association suivie entre EC et ETUDIANT caractérise le conseillé d'un étudiant. Un étudiant peut avoir 0 (quand il vient d'arriver) et 1 tuteur. L'enseignant chercheur peut lui conseiller 0 ou plusieurs élèves.
- L'association appartient entre EC et POLE caractérise le pôle auquel appartient l'EC. L'EC peut appartenir à un et un seul pôle. Le Pôle peut lui contenir 0 ou plusieurs EC.

MODELE LOGIQUE

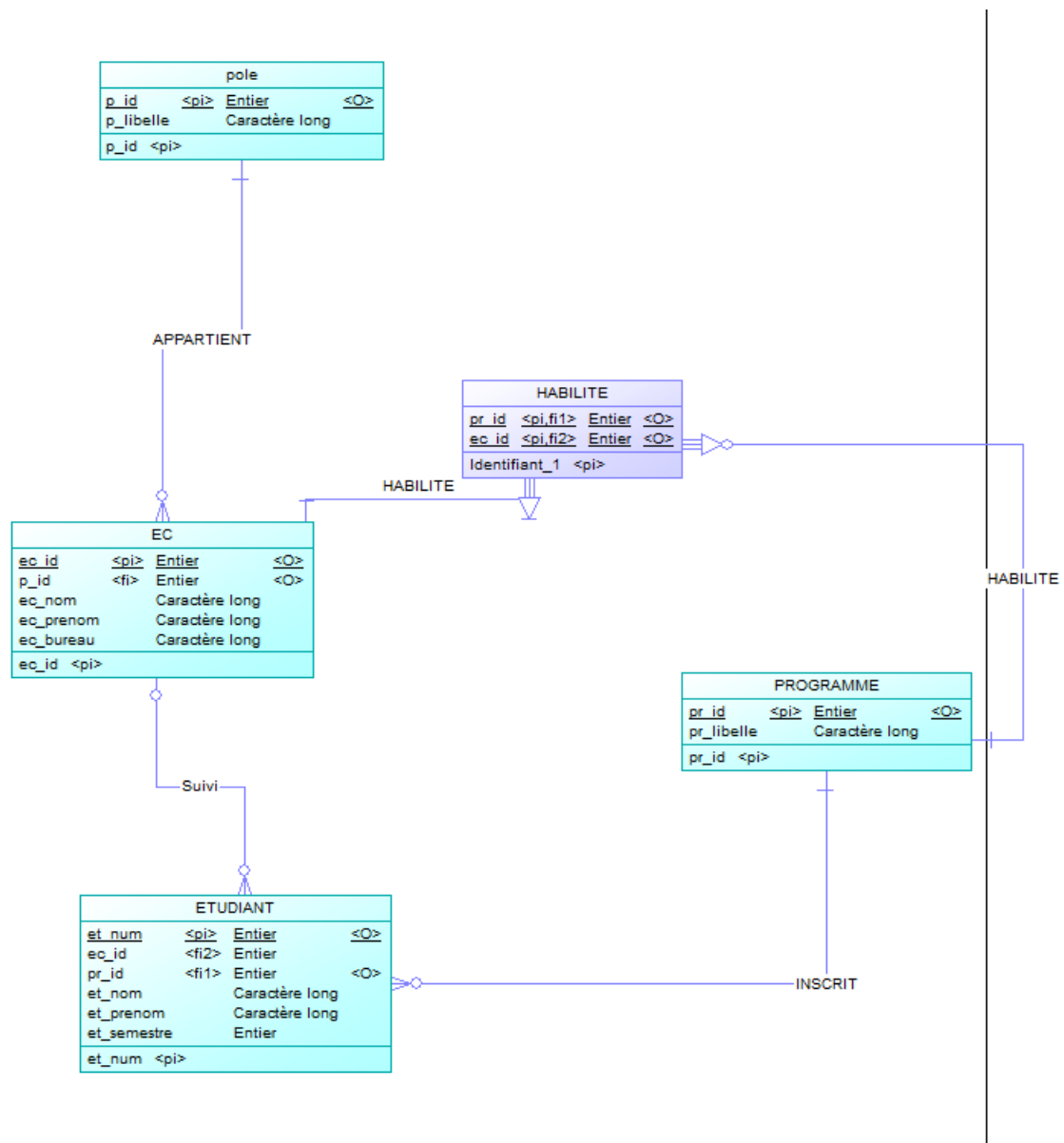
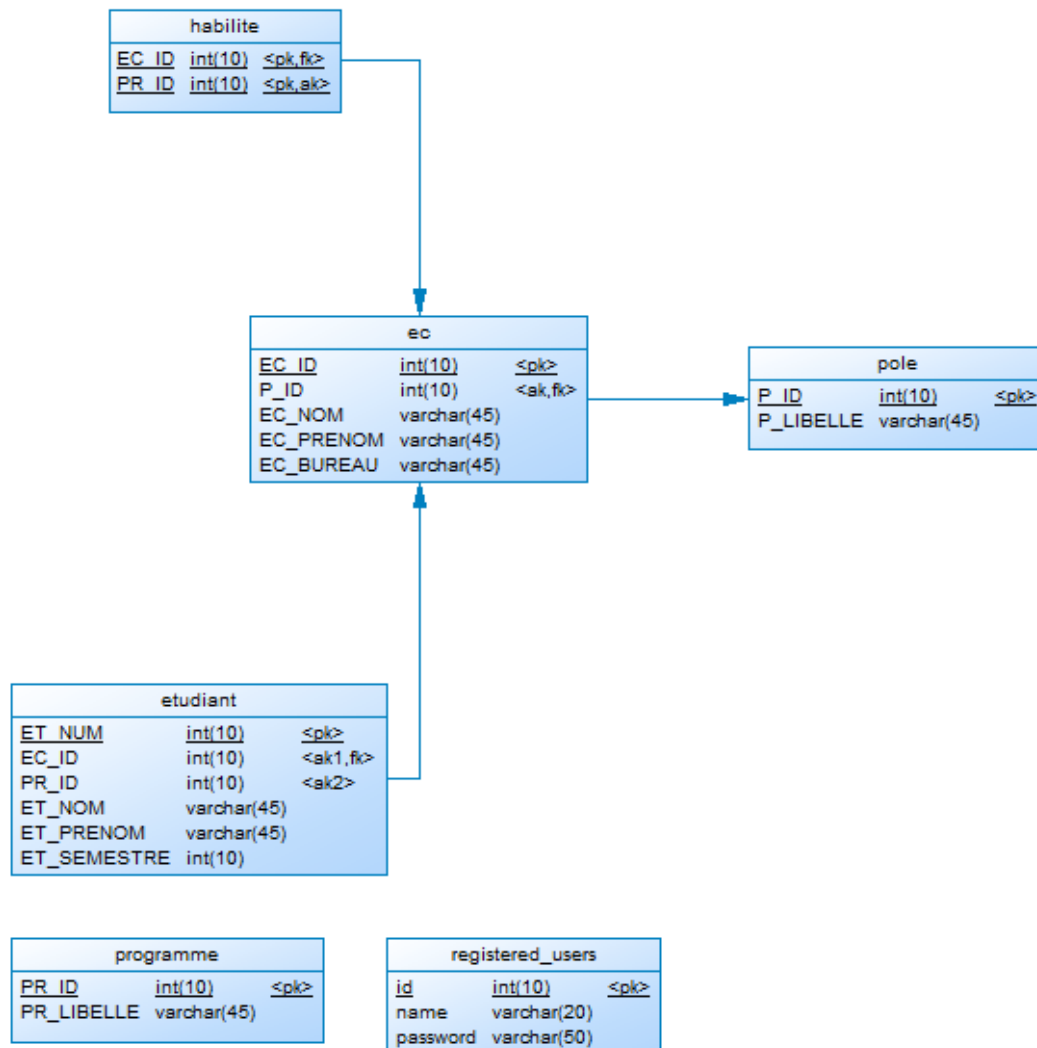


Figure 4 : Modèle Logique de donnée

MODELE PHYSIQUE DE DONNEE

Après avoir réalisé le modèle conceptuelle de donnés nous avons élaboré le modèle physique de donnée suivant :



CHOIX D'UN FRAMEWORK ?

Au début du projet, nous avons longuement hésité sur l'utilisation ou non d'un Framework. Le projet étant assez petit, nous avons d'office éliminé les frameworks les plus lourds comme zend ou symphony. Il existe cependant des Framework moins lourds qui auraient pu être adaptés comme CakePhp, mais le temps d'apprentissage aurait été supérieur aux gains de temps procuré par l'utilisation de Framework.

REALISATION

Après une première phase d'analyse, nous nous sommes mis au développement à proprement parler.

BASE DE DONNES

Voici le script de création des bases que nous avons utilisées pour réaliser ce projet :

```
--  
-- Structure de la table `ec`  
--  
  
CREATE TABLE IF NOT EXISTS `ec` (  
  `EC_ID` int(10) NOT NULL AUTO_INCREMENT,  
  `P_ID` int(10) NOT NULL,  
  `EC_NOM` varchar(45) DEFAULT NULL,  
  `EC_PRENOM` varchar(45) DEFAULT NULL,  
  `EC_BUREAU` varchar(45) DEFAULT NULL,  
  PRIMARY KEY (`EC_ID`),  
  KEY `P_ID` (`P_ID`)  
) ENGINE=InnoDB DEFAULT CHARSET=latin1 AUTO_INCREMENT=19 ;
```

Figure 5 : script de création de la table EC

```
--  
-- Structure de la table `etudiant`  
--  
  
CREATE TABLE IF NOT EXISTS `etudiant` (  
  `ET_NUM` int(10) NOT NULL AUTO_INCREMENT,  
  `EC_ID` int(10) DEFAULT NULL,  
  `PR_ID` int(10) NOT NULL,  
  `ET_NOM` varchar(45) DEFAULT NULL,  
  `ET_PRENOM` varchar(45) DEFAULT NULL,  
  `ET_SEMESTRE` int(10) NOT NULL,  
  PRIMARY KEY (`ET_NUM`),  
  KEY `EC_ID` (`EC_ID`),  
  KEY `PR_ID` (`PR_ID`)  
) ENGINE=InnoDB DEFAULT CHARSET=latin1 AUTO_INCREMENT=12351 ;
```

Figure 6 script de création de la table etudiant

```
--  
-- Structure de la table `habilite`  
--  
  
CREATE TABLE IF NOT EXISTS `habilite` (  
  `EC_ID` int(10) NOT NULL,  
  `PR_ID` int(10) NOT NULL,  
  PRIMARY KEY (`EC_ID`, `PR_ID`),  
  KEY `PR_ID` (`PR_ID`)  
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

Figure 7 script de création de la table habilite

```
--  
-- Structure de la table `programme`  
--  
  
CREATE TABLE IF NOT EXISTS `programme` (  
  `PR_ID` int(10) NOT NULL AUTO_INCREMENT,  
  `PR_LIBELLE` varchar(45) DEFAULT NULL,  
  PRIMARY KEY (`PR_ID`)  
) ENGINE=InnoDB DEFAULT CHARSET=latin1 AUTO_INCREMENT=5 ;
```

Figure 8 script de création de la table programme

```
--  
-- Structure de la table `pole`  
--  
  
CREATE TABLE IF NOT EXISTS `pole` (  
  `P_ID` int(10) NOT NULL AUTO_INCREMENT,  
  `P_LIBELLE` varchar(45) DEFAULT NULL,  
  PRIMARY KEY (`P_ID`)  
) ENGINE=InnoDB DEFAULT CHARSET=latin1 AUTO_INCREMENT=5 ;
```

Figure 9 script de création de la table pole

```
-- Contraintes pour les tables exportées
--

--
-- Contraintes pour la table `ec`
--
ALTER TABLE `ec`
  ADD CONSTRAINT `ec_ibfk_1` FOREIGN KEY (`P_ID`) REFERENCES `pole` (`P_ID`) ON DELETE CASCADE;

--
-- Contraintes pour la table `etudiant`
--
ALTER TABLE `etudiant`
  ADD CONSTRAINT `etudiant_ibfk_1` FOREIGN KEY (`EC_ID`) REFERENCES `ec` (`EC_ID`) ON DELETE CASCADE,
  ADD CONSTRAINT `etudiant_ibfk_2` FOREIGN KEY (`PR_ID`) REFERENCES `programme` (`PR_ID`) ON DELETE CASCADE;

--
-- Contraintes pour la table `habilite`
--
ALTER TABLE `habilite`
  ADD CONSTRAINT `habilite_ibfk_1` FOREIGN KEY (`EC_ID`) REFERENCES `ec` (`EC_ID`) ON DELETE CASCADE,
  ADD CONSTRAINT `habilite_ibfk_2` FOREIGN KEY (`PR_ID`) REFERENCES `programme` (`PR_ID`) ON DELETE CASCADE;
```

Figure 10 contraintes d'intégrités des tables

ASPECT GRAPHIQUE

Nous avons choisi d'utiliser le *framework* JavaScript + CSS *Bootstrap* pour gérer l'aspect graphique du site. L'implémentation des styles de *Bootstrap* dans un site est très aisée : il suffit d'insérer quelques lignes de code dans le *layout* par défaut de l'application.

Le *framework* permet entre autres une adaptation de la mise en forme du contenu à plusieurs résolutions d'écran, un style agréable par défaut pour les différents éléments de la page, une compatibilité accrue avec un grand nombre de navigateurs...

AJAX

Nous avons décidé d'utiliser l'architecture AJAX (Asynchronous Javascript and XML). Ce processus permet d'éviter la transmission et l'affichage d'une nouvelle page web à chaque action de l'utilisateur en se basant sur langage de programmation javascript. En Ajax des fonctions javascript sont exécutées par le navigateur du client, et le serveur leur répond, permettant l'affichage des différentes pages.

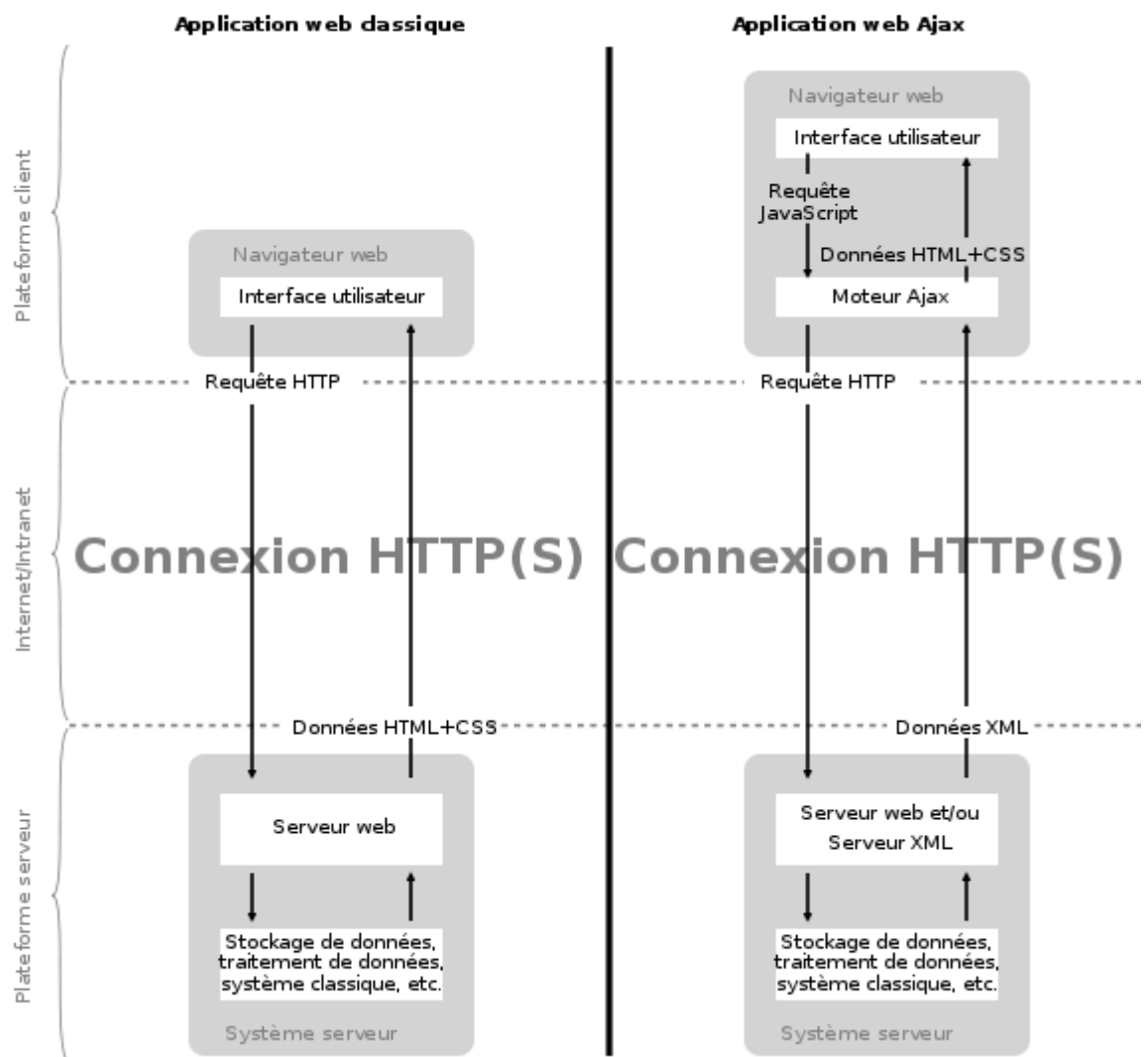
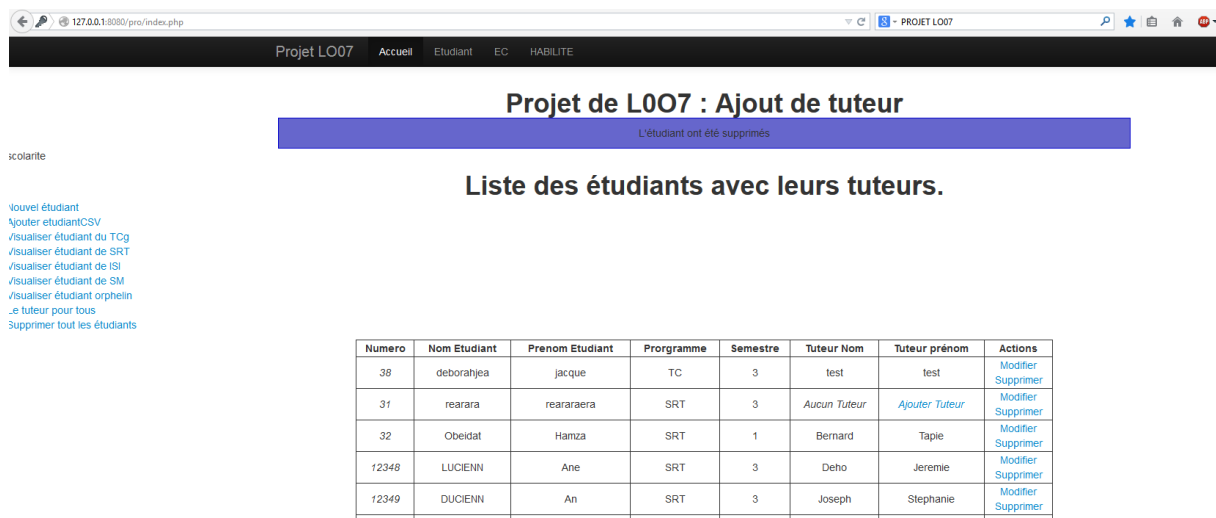


Figure 11 principe de fonctionnement de ajax.

PHP

ASPECT

L'utilisation d'AJAX nous permet d'afficher l'ensemble des informations sur une même page.



Projet LO07 Accueil Etudiant EC HABILITE

Projet de L007 : Ajout de tuteur

L'étudiant ont été supprimés

scolarité

Nouvel étudiant
Ajouter étudiantCSV
Visualiser étudiant du TCg
Visualiser étudiant de SRT
Visualiser étudiant de ISI
Visualiser étudiant de SM
Visualiser étudiant orphelin
Ajouter tuteur pour tous
Supprimer tout les étudiants

Liste des étudiants avec leurs tuteurs.

Numero	Nom Etudiant	Prenom Etudiant	Programme	Semestre	Tuteur Nom	Tuteur prénom	Actions
38	deborahjea	jacque	TC	3	test	test	Modifier Supprimer
31	rearara	reararaera	SRT	3	Aucun Tuteur	Ajouter Tuteur	Modifier Supprimer
32	Obeidat	Hamza	SRT	1	Bernard	Tapie	Modifier Supprimer
12348	LUCIENN	Ane	SRT	3	Deho	Jeremie	Modifier Supprimer
12349	DUCIENN	An	SRT	3	Joseph	Stephanie	Modifier Supprimer

Figure 12 page étudiant pour le service scolarité



Projet LO07 Accueil Etudiant EC HABILITE

Projet de L007 : Ajout de tuteur

drh

Nouvel EC
Ajouter ecCSV
Supprimer tout les EC

Liste des EC

Numero	Nom EC	Prenom EC	Pole	Bureau	Nombre d'élèves	Actions
13	ababa	abababababa	HETIC	3	0	Liste de ses étudiants Supprimer
10	Matin	Daniel	P2MN	T475	0	Liste de ses étudiants Supprimer
18	Ajin	Deker	HETIC	323	0	Liste de ses étudiants Supprimer
16	Benel	Aurelien	HETIC	T107	0	Liste de ses étudiants Supprimer
						Liste de ses

Figure 13 page ec pour le drh

FONCTION PHP

Au cours de ce projet nous nous sommes appliqués à réaliser un maximum de fonction pour chaque action, afin de limiter le copier/coller de code et facilité l'évolutivité du code. Voici les différentes fonctions qui apparaissent dans notre code avec une brève explication.

```

/***** Fonctions ec *****/

function displayec($db) {/** Afficher l'ensemble des ec dans un tableau avec le nombre des étudiants conseillés **/}
function displayec_etudiant($db, $ec) {/** Afficher les étudiants conseillés par l'ec indentifié par l'id ($ec) **/}
function pole($id) {/** Retourner le pôle d'un EC en fonction de l'id ($id) **/}
function LabeltoKeyEC($label) {/** Retourner la clé associé à un label($label) **/}
function verifdoubleEC($nom,$prenom,$db) {/** Verifier le nombre de doublons de l'EC($nom,$prenom) et le renvoie **/}
function ECAjout($nom,$prenom,$pole,$bureau,$db) {/** Ajouter un ec($nom,$prenom,$pole,$bureau) à la base **/}
function supprimerEC($db) {/** Supprimer les EC, Les liens de tutorats de ces EC avec les étudiants et les habilitations **/}
function idEC($nom,$prenom,$db) {/** Renvoie l'id de l'EC($nom,$prenom) qu'on lui passe en argument **/}
function habilitationTC($id,$db) {/** // Habilitation de l'EC au programme TC en utilisant son id
**/}

/***** Fonctions habilitation *****/

function habilitation($id,$pr,$db) {/** Habilitation de l'EC ($id) au programme ($pr) **/}
function displayecHab($db,$pr) {/** Afficher l'ensemble des EC habilités au programme ($pr) **/}
function displayecnotHab($db,$pr) {/** Afficher l'ensemble des EC non-habilités au programme ($pr) **/}
function supprimerlienEC($db,$enc) {/** Supprimer l'habilitation d'un EC ($enc) **/}
function ajoutehabilitation($db,$enc,$pr) {/** Habilitier d'un EC ($enc) au programme ($pr) **/}
function ajoutehabilitationpole($db,$pole,$pr) {/** Habilitier tous les EC du pole ($pole) au programme ($pr) **/}

/***** Fonctions étudiant *****/

function displayAll($db) {/** Afficher l'ensemble des étudiants dans un tableau avec leur tuteurs **/}
function displayetudiant($db) {/** Afficher uniquement la liste des étudiants **/}
function displayetudiant_sanstuteur($db) {/** Afficher uniquement la liste des étudiants qui n'ont pas de tuteurs **/}
function displayetudiant_programme($db, $programme) {/** Afficher l'ensemble des étudiants selon un programme ($programme) **/}
function ajouterTuteur($id, $db) {/** Ajouter un tuteur à l'étudiant ($id) **/}
function tuteurAll($db) {/** Ajouter un tuteur aux étudiants n'en ayant pas **/}
function programme($id) {/** Retourner le programme en fonction de l'id ($id) **/}
function EtudiantAjout($nom,$prenom,$semestre,$programme,$db) {/** Ajouter un etudiant($nom,$prenom,$semestre,$programme,$db) à la base **/}
function supprimerEC($db) {/** Supprimer tous les étudiants **/}
function verif($db,$prog,$ec) {/** Vérifier si l'EC ($ec) est habilité au programme($prog) **/}
function tuteur($db,$etudiant) {/** Retourner l'id du tuteur de l'étudiant ($etudiant) **/}

formulaire_csv.php ; {/** Upload est traitement du fichier CSV et insertion de ces données dans la base de donnée **/}

```

LIVRABLE

Le projet est maintenant terminé, et l'ensemble de fonctions exigées a été réalisé.

DIFFICULTES RENCONTREES

PROBLEMES MATERIELS

A une semaine de la date de rendu, le serveur dédié hébergé chez online a rencontré des problèmes de disque dur qui ont entraîné son remplacement sans la possibilité de récupérer les différents fichiers qui y étaient stockés.

UPLOAD DE FICHIER EN AJAX

Nous avons réalisés que l'upload de fichier directement en AJAX était impossible. A la place, l'utilisation de frames pour simuler le comportement était conseillée, cependant cela était bien trop lourd à mettre en place. Nous avons donc du proposer des formulaires d'upload plus traditionnels.

CONCLUSION

Ce projet fût globalement très enrichissant, permettant de mettre en œuvre les différentes choses apprises au cours de l'UV LO07. La relative liberté nous ayant été laissée quant au moyen à utiliser pour le réaliser nous ont permis de découvrir un peu plus les outils mis à la disposition des développeurs web, qu'il soit technique comme les frameworks ou managérial comme un gestionnaire de version ou de projet.