

CHALOPIN Quentin
GIRARD Maxime
AUDRERIE Claire
RAILLERE Simon
REPILEZ Kevin

Projet collectif Gestion émargement RFID

Encadrants :

Carl Esswein
Pascal Makris

Sommaire

Introduction

Cahier des charges

Partie Application

Serveur

Plates-formes utilisées

Gestion émargement

Stockages des données

Synchronisation carte étudiante et l'étudiant

Plusieurs boîtiers transportables

Application Smartphone

Partie RFID

Alimentation

Lecture carte RFID

Synchronisation des données vers le serveur

Boîtier transportable

Gestion de projet

Répartition des tâches

Planning prévisionnel

Planning réel

Partie application

Spécification

Choix technologique

Réalisation

Partie RFID

Choix technologique

Réalisation

Introduction

Ce sujet a été proposé dans le cadre du projet collectif système et réseau. Son objectif est de proposer une solution à la gestion des émargements d'un groupe d'individus afin de s'assurer de sa présence dans un local. L'application envisagée pour valider la solution serait de tester le système dans le cadre de la formation par apprentissage et en particulier lors des créneaux travail en autonomie. Plutôt que de faire signer individuellement les personnes et qu'ensuite les responsables de la scolarité saisissent manuellement les informations dans le logiciel de suivi. Nous devions proposer une solution qui mette en œuvre l'exploitation de l'identifiant de la carte « atoutcentre » après passage de celui-ci devant un lecteur RFID qui ensuite alimente la base de données pour in fine fournir mensuellement l'état des lieux des présences et absences de chacun.

Cahier des charges

Utilisateur final

Deux utilisateurs principaux se dessinent :

- Les enseignants “responsables”
- La secrétaire de la formation

Les enseignants “responsables” sont constitués des responsables d’années, le directeur de département, le directeur des études, etc... Ces utilisateurs ne sont pas encore bien définis mais ils auront tous la même utilisation du système. C'est-à-dire qu'ils auront besoin de faire des recherches sur l'absence (ou pas) d'un élève en fonction du cours, du professeur ou encore de la date.

La secrétaire est l'utilisatrice principale du système ou plus précisément de l'application. En effet, les documents générés devront être selon la volonté de cette utilisatrice. Chaque mois, elle doit faire un compte-rendu de tous les élèves absents auprès du CFAI.

Partie Application

Serveur

Hébergé à Polytech', il permettra l'accès à l'application à tout moment si l'utilisateur est connecté au réseau de Polytech'. Sa configuration et les technologies utilisées seront faites en collaboration avec le service informatique pour adapter au mieux son utilisation.

Dans un premier temps, une machine virtuelle en base test sera disponible au service informatique. Elle sera utilisable qu'à partir du réseau Polytech'. Puis, lorsque l'application sera finalisée, la solution sera déployée sur un serveur de production.

Lorsque le boîtier d'émargement aura la possibilité de se connecter au serveur, la connexion sera établie et le serveur récupérera les données récoltées au préalable.

Plates-formes utilisée

L'application devra être accessible par le maximum d'utilisateurs. Pour cela, un site web ou une web application serait un choix judicieux.

L'utilisation d'un site web en temps qu'interface graphique nous permet de simplifier le processus de maintenance de l'application. En effet, dans le cas de l'utilisation d'une application traditionnelle, nous devons installer le logiciel sur chacun des ordinateurs qui en ont besoin, même chose si on veut effectuer une mise à jour. Au contraire, le code d'un site web est centralisé sur le serveur hébergé à Polytech. Les utilisateurs n'ont aucune manipulation à effectuer sur leur machine à part taper l'adresse du serveur dans leur navigateur web. Pour la mise à jour, il suffit que le développeur modifie le code source présent sur le serveur pour que les utilisateurs voient instantanément les modifications apportées.

Gestion émargement

Pour gérer au mieux l'émargement des élèves, le scénario nominal sera le suivant :

1. A chaque cours, l'élève devra passer sa carte sur l'un des boîtiers disponibles dans la salle de classe ou au niveau de la scolarité.
2. Dès que possible, le boîtier se connectera au serveur pour lui envoyer toutes les données récoltées.
3. Lorsque le serveur recevra les données, il devra faire l'association entre la carte utilisée et les informations de l'étudiant.
4. Une fois les informations traitées et stockées, l'utilisateur pourra voyager dans la liste des étudiants absents pendant au moins un cours. Il pourra faire des recherches en saisissant des informations utiles. Par exemple, une recherche par cours, par date, par étudiant, etc...
5. S'il le souhaite, l'utilisateur pourra générer un document listant tous les élèves ayant été absents. Cette fonction sera très utile pour la secrétaire qui se charge de remonter les absences auprès du CFAI.

Pour savoir qui est absent, l'application requiert des informations. En effet, une synchronisation avec l'emploi du temps n'est pas possible ce qui rend la tâche plus difficile.

Pour y remédier, la secrétaire devra rentrer les heures "libres" de chacun des groupes de travail. Ainsi, l'application sera en mesure de savoir si l'élève doit émarger ou non.

Lors de la génération du document, les informations suivantes sont primordiales :

- Créneau où l'élève a été absent
- Nom et prénom de l'élève

Sur le serveur en lui-même, il va de soi que le système connaisse à quel groupe et quelle classe appartient l'élève. Pour cela, le secrétariat a, à sa disposition un fichier Excel associant le nom de l'élève avec sa classe qu'il devra rentrer au moins une fois chaque année (ou durant l'année si la liste change) et qui chargera le tout dans la base de données.

Du côté du secrétariat, la saisie des heures "libres" se fera de la façon suivante :

1. Sélection de l'année
2. Sélection du groupe
3. Sélection de la date et de l'heure (8h15/10h30/14/16h15)
4. Validation du créneau libre

L'application devra aussi prendre en compte les alternances entreprises.

Administration

L'administrateur de l'application aura accès à toutes les données. De plus, il aura accès au paramétrage de l'application. On peut imaginer les champs suivants :

- Paramétrage de la durée de stockage des données sur l'étudiant.
- Paramétrage sur les alertes et les reporting envoyés.
- L'ajout du fichier de synchronisation entre la carte lue et les informations sur l'étudiant. Le type de fichier n'est pas encore défini.

Alertes/Reporting

Le contenu des alertes et de reporting n'est pas encore défini. Une alerte sera faite à un instant t pour signaler l'absence d'un élève par exemple. En revanche, un reporting se fait de façon périodique. On peut imaginer un reporting des absences tous les mois directement envoyé à la secrétaire.

Stockage des données

Pour le stockage des données, tous les émargements seront stockés en base de données et gardés pendant une durée à définir dans le paramétrage.

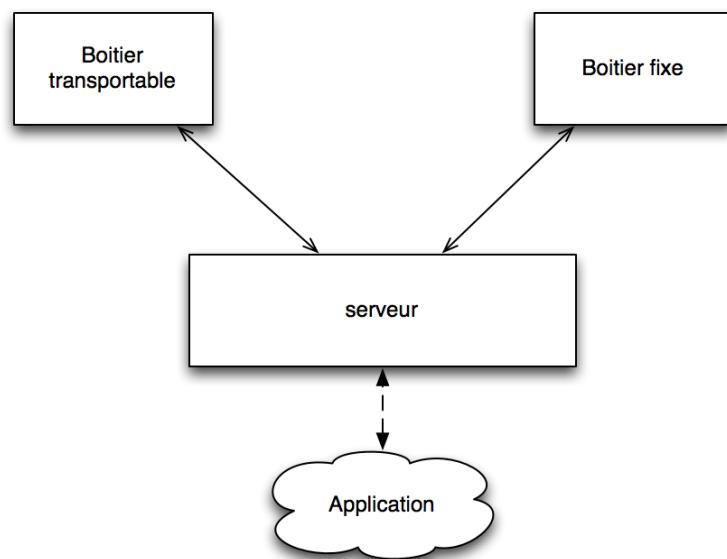
De plus, une gestion des historiques est primordiale. Il est impératif que toutes les données soient sauvegardées et ce, pendant tout le cursus de l'élève.

Synchronisation numéro de carte étudiante avec l'étudiant

La scolarité devra disposer d'un fichier faisant le lien entre le numéro de carte étudiante, le nom, le prénom et le numéro d'étudiant. Ce fichier devra être intégré à l'application ou à la base de donnée pour permettre d'établir le lien entre le numéro de la carte qui a servi à badger et l'étudiant détenteur de cette carte.

Synchronisation avec plusieurs boîtiers transportables

Le schéma du système est le suivant :



Le développement du système sera effectué de manière à utiliser plusieurs boîtiers. En l'occurrence, au moins deux boîtiers seront utilisés. L'un sera posté au secrétariat avec un lecteur visé sur le mur de la scolarité. Le deuxième sera vacant et voyagera avec les professeurs.

Partie RFID

Alimentation

L'alimentation du boîtier ne doit pas être un problème pour son transport. En effet, ce boîtier permettra l'émargement grâce aux cartes étudiantes et sera amené à chaque début de cours. Par exemple, l'alimentation secteur est à oublier car cela rendrait le système trop contraignant.

On peut envisager une alimentation USB qui permet d'avoir un système facilement transportable et nécessite simplement un ordinateur ou un adaptateur secteur vers USB.

Lecture carte RFID

Le boîtier devra lire et récupérer les données d'une carte RFID. Cette carte RFID est celle de l'université nommée "atoutcentre". Le lecteur RFID n'est pas imposé et fera l'objet d'une étude. Celui-ci devra se connecter au boîtier facilement.

Synchronisation des données vers le serveur

Le boîtier transportable doit permettre de stocker les données provenant des badges tant qu'une synchronisation avec le serveur n'a pas pu être effectuée. Ceci fait, les données pourront être supprimées du module transportable.

La méthode de synchronisation devra faire l'objet d'une étude.

Boîtier transportable

Dans un premier temps, nous créerons un boîtier transportable. Les composants internes au boîtier et l'architecture à mettre en place devront faire l'objet d'une étude, notamment la carte, la lecture RFID, l'alimentation, le stockage des données, etc.

Après réalisation des tests fonctionnels sur ce module, nous pourrons réaliser un second boîtier. L'un des boîtiers sera voué à être transporté dans les salles de cours pour permettre aux étudiants de badger à chaque cours. L'autre boîtier sera en libre accès au secrétariat afin de faciliter le badgeage lors des heures de travail personnel.

Gestion de projet

Définition et répartition des tâches

Lors du début du projet, deux parties distinctes se sont très vite dessinées : la partie transportable et la partie application (serveur). Sur la partie transportable, deux fonctionnalités sont à étudier. Nous avons d'une part la lecture et l'enregistrement des informations de la carte étudiante et d'autre part, l'envoi des données sur le serveur.

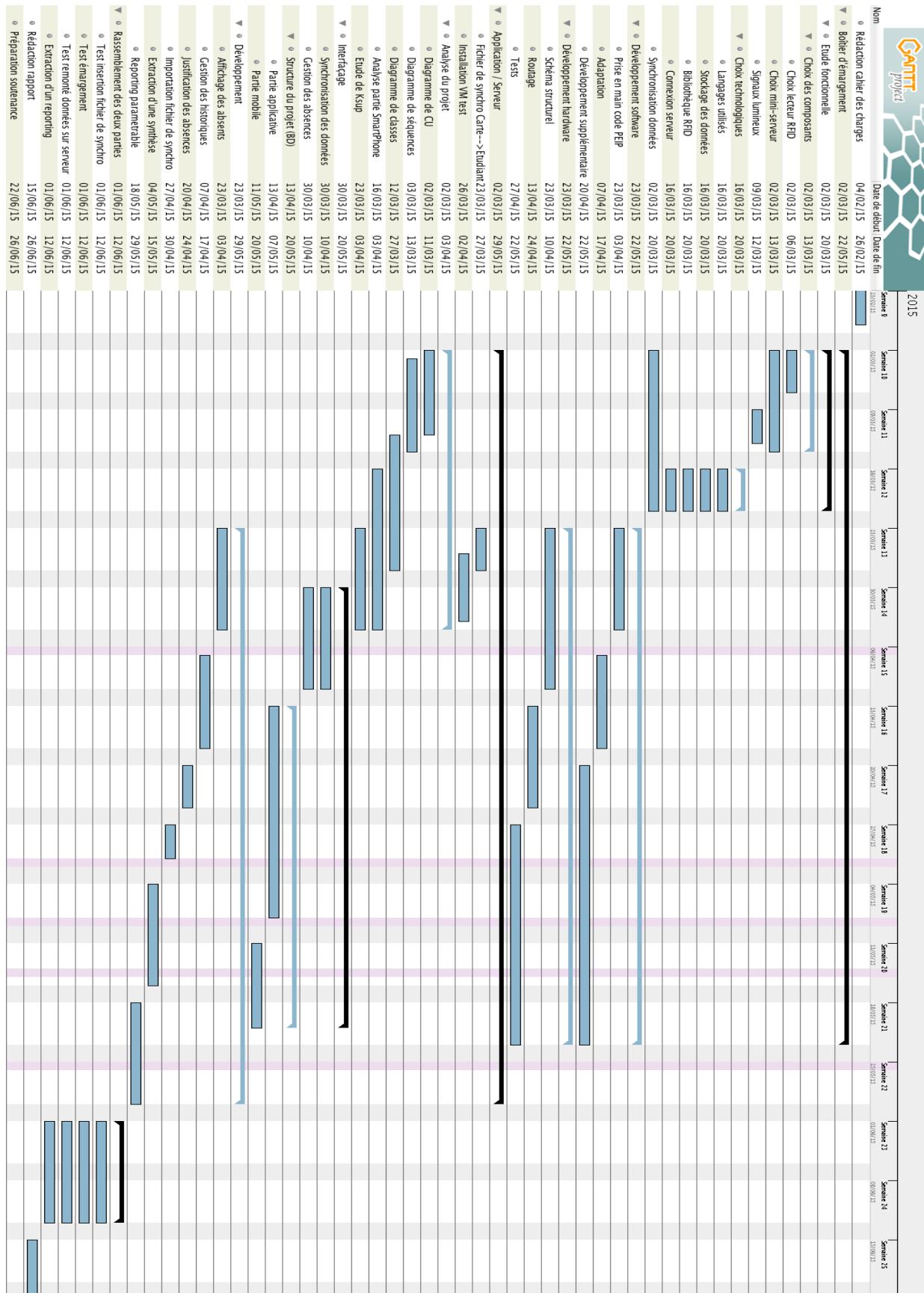
Quant à la partie serveur, nous avons nécessairement besoin d'un développeur pour l'application ainsi que d'un administrateur de base de données.

Claire et Simon s'occuperont de la partie transportable. Simon réalisera la synchronisation des données vers le serveur et Claire du développement de l'application qui lit les informations de la carte.

Maxime et Kevin s'occuperont donc de la partie serveur / application. Maxime travaillant déjà à l'entreprise sur base de données, il s'est naturellement orienté vers cette partie. Kevin ayant déjà travaillé sur un site web grâce aux projets précédents, il a donc pris en charge le développement de l'application.

Planning prévisionnel

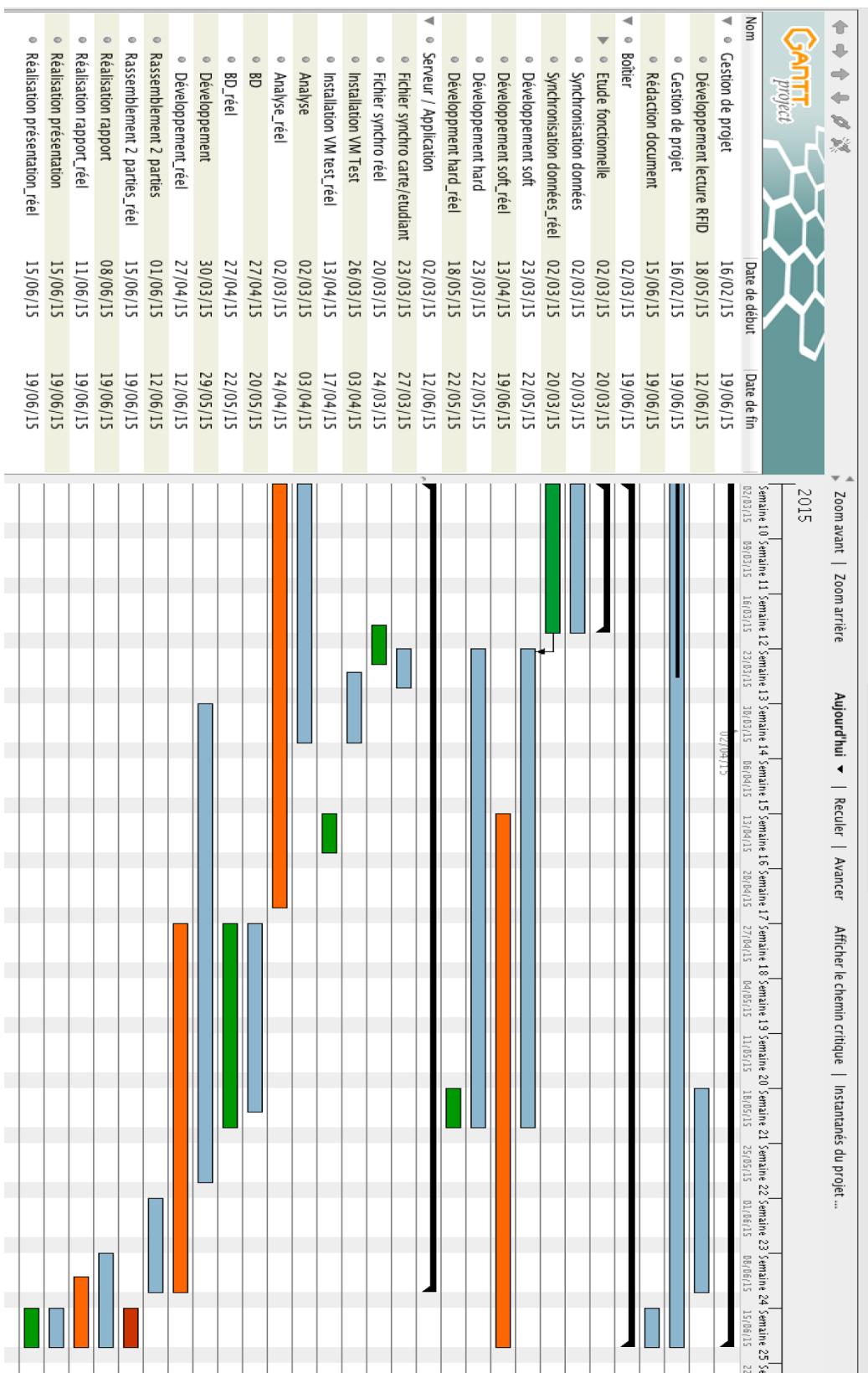
Ci-dessous le planning prévisionnel du projet :



Comme on peut le voir dans chacune des parties, une période d'analyse et de conception a été mise en place. Des choix hardware et software doivent être fait sur la partie transportable. En revanche sur la partie serveur / application les choix technologiques sont déjà établis. En effet, nous devons respecter les choix technologiques du service informatique.

Une fois ces analyses terminées, le développement peut alors commencer pour les deux parties. Comme prévu, Claire s'occupe du développement et réadaptation de l'application et Simon de la partie synchronisation. De même pour la partie application, Maxime s'occupe de la base de données et Kevin du développement software.

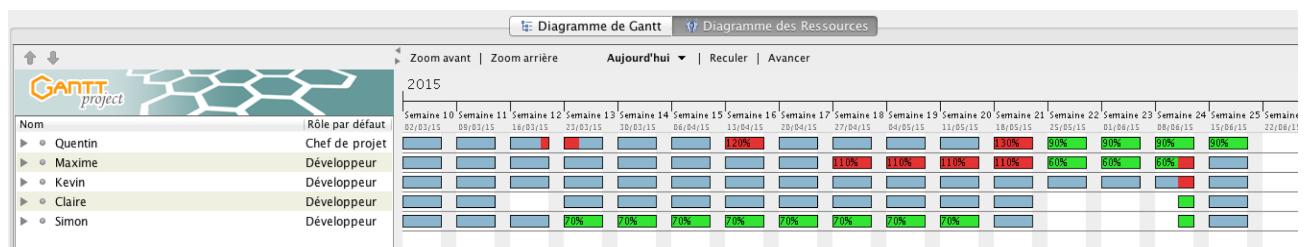
Planning réel



Sur le planning réel, on peut voir que certaines tâches ont pris beaucoup plus de temps que d'autres. Malheureusement, les projets étant sur notre temps libre, il est difficile de tenir des deadlines précises ou encore d'évaluer le temps réel que cela peut nous prendre. Sur la partie application, on peut voir que la partie analyse a pris plus de temps que prévu principalement causé par l'alternance entreprise. Le développement a également pris du retard à cause de problème sur les machines virtuelles de test. Il s'est avéré également que l'apprentissage du Framework CakePHP a pris plus de temps que je ne le pensais notamment pour Kevin qui a quelques difficultés en développement.

Sur la partie transportable, le programme de lecture de carte RFID a pris beaucoup plus de temps que prévu. En effet, Claire a rencontré des difficultés pour la récupération des informations de la carte étudiante. De plus, le choix de fonctionnement pour la partie synchronisation des données n'était pas judicieux et encore moins sécurisé. Après la réunion, nous avons repensé et développé cette partie.

Ci-dessous, le taux de charge de notre équipe :



On peut voir que certaines semaines ont été plus chargées que d'autres. Notamment lorsque la VM de test a été supprimée ou encore lorsque Maxime s'est penché sur l'application web.

Partie application

Gestion de la base de données

Outils utilisés

MySQL



Utilisant une solution LAMP pour la configuration de notre serveur Web, nous travaillons donc avec MySQL pour la gestion de notre base de données. MySQL est un système de gestion de base de données relationnelles sous licence GPL, ce qui nous permet une utilisation libre de l'outil. De plus, c'est un des systèmes les plus utilisés mondialement et donc a de nombreuses ressources sur Internet nous aidant à une bonne mise en place de la base de données.

phpMyAdmin

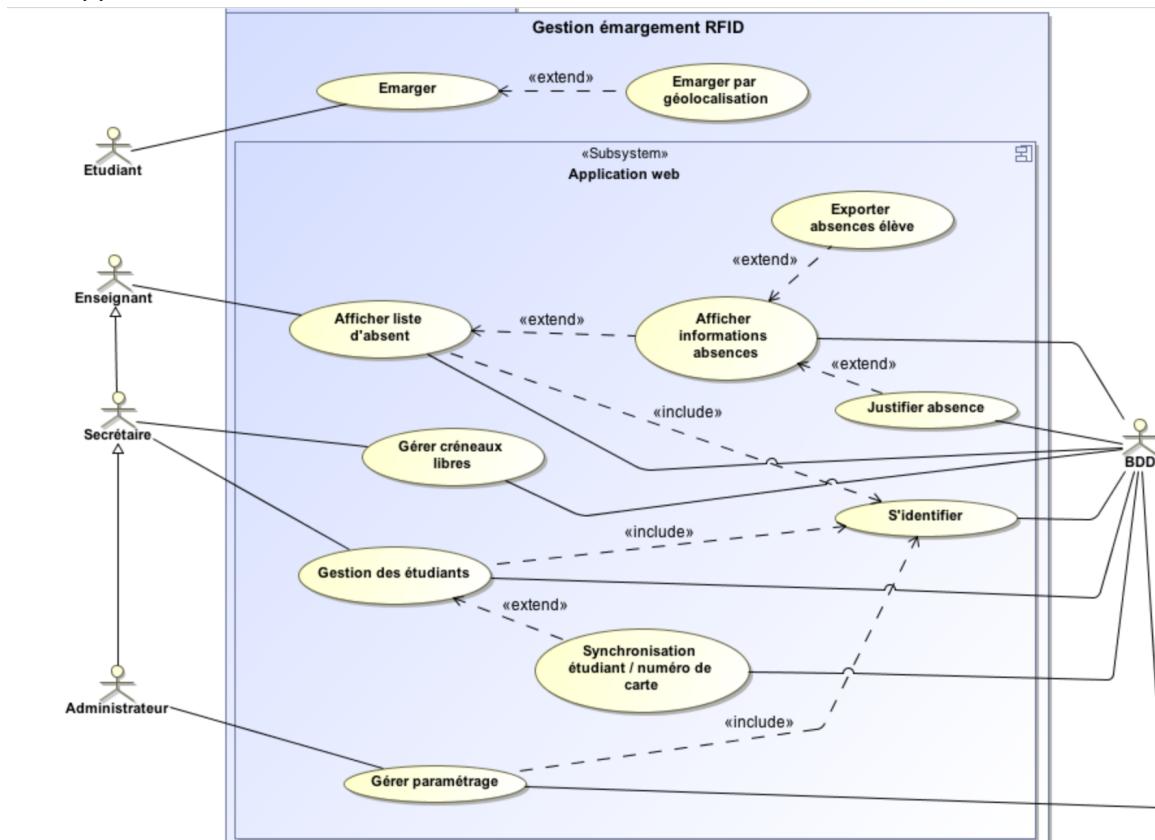


Après installation de MySQL sur le serveur, nous pouvons travailler sur la base de données via le terminal. Malheureusement, cette solution n'est pas des plus pratiques, obligeant à ne passer que par des requêtes SQL pour observer la base de données. Pour rendre notre travail plus facile, nous avons donc décidé d'installer phpMyAdmin.

phpMyAdmin est une application Web réalisée en Php et sous licence GNU GPL permettant de gérer facilement une base de donnée MySQL via une interface graphique. Nous pouvons donc avoir une meilleure vue de la base ainsi que des interactions plus simples.

Interaction avec la base de données

Nous avons réalisé une étude UML de notre application afin de spécifier au mieux toutes les actions à effectuer sur les bases de données. Nous avons créé un diagramme d'utilisation pour structurer les interactions entre les utilisateurs et les différentes fonctions de l'application.



Ce diagramme présente toutes les actions possibles par chaque utilisateur, y compris la base de données. On peut voir que la base de données est à la base de toutes les actions pouvant être réalisées par l'ensemble des utilisateurs.

C'est notamment grâce à ce diagramme que nous avons pu construire la base de données comme décrite ci-après.

Nous avons aussi réalisé des diagrammes de séquences décrivant les interactions applications/base de données, pour la gestion de l'emploi du temps, l'affichage d'une liste d'absence, la liste d'absence d'un étudiant et la synchronisation avec le boîtier transportable. Ces diagrammes sont fournis en annexe.

Définition des tables de la base de données

La base de données telle qu'elle a été conçue est composée de 7 tables distinctes. Conformément à la documentation de CakePhp, chaque nom de table est au pluriel afin de pouvoir être interprété par le framework plus facilement lors des requêtes.

Certaines colonnes décrites ci-dessous ont devant leur nom un symbole désignant une caractéristique particulière :

- * : La colonne est une clé primaire, c'est-à-dire qu'on impose une contrainte d'unicité sur la colonne afin de ne pas pouvoir insérer dans la table deux fois la même valeur. De plus, un index implicite est lié à la clé primaire, ce qui permet d'accélérer la recherche sur cette colonne.
- \$: La colonne est une clé étrangère, c'est-à-dire qu'elle est liée à une colonne dans une autre table. On ne peut donc pas insérer dans la table possédant la clé étrangère une valeur qui n'existe pas dans la table liée.
- ☰ : La colonne est indexée, ce qui permet d'accélérer la recherche d'informations. De plus, elle permet à la colonne d'être liée à une clé étrangère.

De plus, on peut voir que chaque table de la base de données possède une colonne v_statut qui est constituée d'un varchar(1). Cette colonne permet de savoir si la ligne de la table doit être interprétée ou non par l'application. En effet, dans un souci de suivi des données dans la base, nous limitons au maximum la suppression de lignes dans les tables via la requête SQL "delete". On préférera changer la valeur du v_statut de la ligne en "D", ce qui indique à l'application que la ligne est censée être supprimée mais permet de garder les données dans la base.

Etudiants	Type	Description
* v_id_etu	varchar(8)	Identifiant national de l'étudiant
☐ v_id_carte	varchar(14)	Identifiant de la carte Atout-Centre
v_prenom	varchar(50)	Prénom
v_nom	varchar(50)	Nom
\$ v_id_groupe	integer	Identifiant du groupe de l'étudiant
v_statut	varchar(1)	Statut de l'étudiant

Notes sur la table Etudiants : Cette table possède toutes les particularités définies plus haut. En effet, sa clé primaire correspond au numéro national de l'étudiant, constitué de 8 caractères, sa clé étrangère permet de lier la table à la table Groupes, permettant ainsi de s'assurer que l'étudiant appartient bien à un groupe de TP existant. Et enfin sa colonne

v_id_carte, qui correspond au numéro en 14 caractères de sa carte Atout-Centre. Elle est indexée pour mettre une clé étrangère dans une autre table qui sera liée à v_id_carte.

Utilisateurs	Type	Description
* v_id_user	integer	Identifiant de l'utilisateur
v_prenom	varchar(50)	Prénom
v_nom	varchar(50)	Nom
v_type	varchar(1)	Type de l'utilisateur
v_login	varchar(20)	Login
v_mdp	varchar(20)	Mot de passe
v_statut	varchar(1)	Statut de l'utilisateur

Notes sur la table Utilisateurs : v_type correspond au type d'utilisateur de l'application. Il permet ainsi de définir les droits d'accès en vigueur dans l'application. Par exemple, un utilisateur dont le v_type est "A" est considéré de type "Administrateur". Il a donc accès à l'ensemble de l'application, contrairement à utilisateur de type "S" ou "Secrétaire" qui ne pourra pas modifier les paramètres de l'application.

Creneaux	Type	Description
* v_id_creneau	integer	Identifiant du créneau libre
\$ v_id_groupe	integer	Identifiant du groupe concerné
d_debut	datetime	Date de début du créneau
d_fin	datetime	Date de fin du créneau
v_statut	varchar(1)	Statut du créneau

Classes	Type	Description
* v_id_classe	integer	Identifiant de la classe
v_libelle	varchar(100)	Libelle de la classe
v_statut	varchar(1)	Statut de la classe

Groupes	Type	Description
* v_id_groupe	integer	Identifiant du groupe
\$ v_id_classe	integer	Identifiant de la classe englobant le groupe
v_libelle	varchar(100)	Libelle du groupe
v_statut	varchar(1)	Statut du groupe

Emargements	Type	Description
* v_id_emarg	integer	Identifiant de l'émargement
\$ v_id_carte	varchar(14)	Identifiant de la carte Atout-Centre qui émarge
d_date_emarg	datetime	Date de l'émargement sur le boîtier
d_date_synchro	datetime	Date de la synchronisation de l'émargement sur le serveur
v_statut	varchar(1)	Statut de l'émargement

Notes sur la table Emargements : La table Emargements est composée de deux dates distinctes, d_date_emarg qui correspond à la date où l'étudiant a émargé, c'est-à-dire au moment où le boîtier a récupéré les données de sa carte Atout-Centre. La seconde, d_date_synchro correspond à la date où le boîtier a été synchronisé avec la base de données sur le serveur. Cette seconde date n'est pas utilisée dans l'application mais nous permet de mieux tracer les données en cas de problème.

Absences	Type	Description
* v_id_abs	integer	Identifiant de l'absence
\$ v_id_etu	varchar(8)	Identifiant de l'étudiant
d_abs	datetime	Date de l'absence
v_just	varchar(1)	L'absence est-elle justifiée?
v_statut	varchar(1)	Statut de l'absence

Notes sur la table Absences : Cette table possède une colonne v_just qui décrit si l'absence est justifiée ou non. Pour cela, elle est composée d'un varchar(1), "O" ou "N", respectivement pour "Oui" ou "Non". Cette valeur est bien sûr modifiable via l'application.

Automatisation des tâches

Procédures stockées

Dans notre data flow, les données liées à l’émargement sont envoyées depuis le boîtier sur la table Emargements de la base de données du serveur. En revanche l’application n’utilise pas cette table pour savoir si un étudiant était absent mais elle utilise la table Absences. Pour pouvoir créer une ligne dans la table Absences, nous devons vérifier dans plusieurs tables comme Etudiants, Creneaux ou encore Groupes. Cette vérification peut être lourde. Nous avons donc décidé d’utiliser une procédure stockée pour l’effectuer.

Les procédures stockées ont été introduites dans MySQL depuis sa version 5 et permettent d’automatiser certaines tâches en écrivant une série d’instructions SQL. Nous avons donc écrit une procédure “ajouter_absences” qui prends tout les émargements de la journées et qui pour chaque étudiant vérifie les créneaux libres de son groupe de TP. La procédure ajoute au besoin une ligne dans la table Absences. A l’état actuel, la procédure ne permet de traiter que les émargements de la journée.

Programmateur d’évènements (Event Scheduler)

Comme décrit plus haut, notre procédure stockée nous permet d’ajouter automatiquement des lignes dans la table Absences au besoin, mais cette procédure doit être appelée par une commande SQL afin qu’elle s’exécute. Nous utiliserons pour cela l’Event Scheduler.

L’Event Scheduler est apparu dans MySQL à sa version 5.1 et permet d’exécuter automatiquement une commande SQL, une procédure stockée, ou beaucoup d’autres choses. L’évènement peut être programmé pour se déclencher à un intervalle de temps régulier. Pour notre besoin, nous allons donc lui demander d’exécuter notre procédure stockée “ajouter_absences” tous les jours à 20h précise afin de créer l’ensemble des absences de la journée.

Triggers

Grâce à notre procédure stockée et notre event, nous pouvons créer automatiquement des absences pour la journée en fonction des émargements et des créneaux libres présents dans la base. Malheureusement, si un créneau libre n’a pas été renseigné avant ou pendant la journée, une absence sera créée. Pour remédier à ce problème, nous utilisons un trigger SQL.

Les triggers ou “déclencheurs” permettent d’exécuter une séquence d’instructions SQL lorsque l’on insère ou modifie une ligne dans une table donnée. Dans notre cas, notre trigger va s’exécuter à chaque fois que l’on va insérer une ligne dans la table Creneaux. Il va aussi vérifier s’il y a des absences à la date du nouveau créneau créé. Si il y en a, nous allons passer le v_statut de l’absence en “D” afin que la ligne ne soit plus prise en compte par l’application.

Application Web

Choix du langage de programmation

Ayant définie précédemment que nous allions utiliser une interface web pour notre application, nous devons choisir une technologie parmi toutes celles disponibles.

Certains membres de l'équipe ont déjà acquis de l'expérience en développement web grâce à divers projets que ce soit à Polytech ou en entreprise, et notamment le développement PHP.

PHP (Hypertext Preprocessor) est un langage de programmation s'intégrant aux classiques HTML et CSS et qui permet de rendre des pages Web dynamiques. Sa popularité en fait un des langages Web les plus utilisés, et sa communauté très active offre de nombreuses ressources simplifiant l'apprentissage du langage et la résolution des problèmes que l'on peut rencontrer tout au long du développement.

Choix du framework de développement

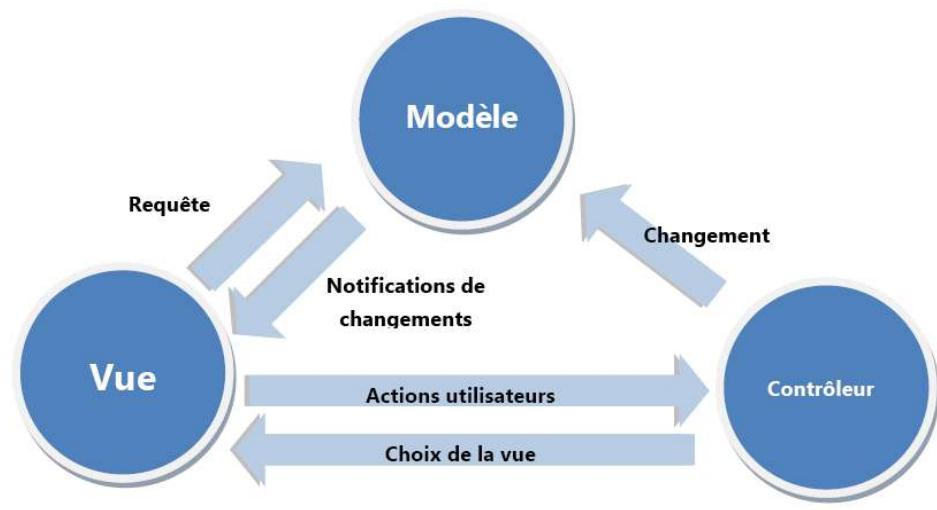
Comme expliqué précédemment, la communauté autour de PHP est très active et le nombre de framework PHP facilitant le développement de sites web est très important.

Certains framework sont particulièrement utilisés par les agences de développement web tels que Symphony ou encore Zend. Malheureusement, La courbe d'apprentissage de ces frameworks est extrêmement dure ce qui peut être un problème pour le bon fonctionnement du projet. Notre choix s'est donc porté sur un framework plus accessible et avec une communauté active, CakePHP. De plus ce framework a déjà été utilisé par un membre de notre groupe lors d'un projet en entreprise.



CakePHP est un framework PHP 5.4+ moderne fournissant un modèle MVC et une gestion des interactions avec la base de données simplifiées. Dans le cadre de notre projet, nous utilisons la version la plus récente de CakePHP, c'est-à-dire la 3.0.

Modèle MVC



Le modèle MVC (Modèle - Vue - Contrôleur) est un patron d'architecture logicielle divisant le code en trois parties distinctes :

- **Modèle** : Le modèle représente le cœur (algorithmique) de l'application : traitements des données, interactions avec la base de données, etc. Il décrit les données manipulées par l'application. Il regroupe la gestion de ces données et est responsable de leur intégrité. La base de données sera l'un de ses composants. Le modèle comporte des méthodes standards pour mettre à jour ces données (insertion, suppression, changement de valeur). Il offre aussi des méthodes pour récupérer ces données. Les résultats renvoyés par le modèle ne s'occupent pas de la présentation.
- **Vue** : C'est avec la vue que l'utilisateur va interagir. Sa première tâche est de présenter les résultats renvoyés par le modèle. Sa seconde tâche est de recevoir toute action de l'utilisateur (clique de souris, sélection d'un bouton radio, cochage d'une case, entrée de texte, etc). Ces différents événements sont envoyés au contrôleur. La vue n'effectue pas de traitement, elle se contente d'afficher les résultats des traitements effectués par le modèle et d'interagir avec l'utilisateur.
- **Contrôleur** : Il prend en charge la gestion des événements de synchronisation pour mettre à jour la vue ou le modèle et les synchroniser. Il reçoit tous les événements de la vue et enclenche les actions à effectuer. Si une action nécessite un changement des données, le contrôleur demande la modification des données au modèle afin que les données affichées se mettent à jour. D'après le patron de conception observateur/observable, la vue est un « observateur » du modèle qui est lui « observable ». Certains événements de l'utilisateur ne concernent pas les données mais la vue. Dans ce cas, le contrôleur demande à la vue de se modifier. Le contrôleur n'effectue aucun traitement, ne modifie aucune donnée. Il analyse la requête du client et se contente d'appeler le modèle adéquat et de renvoyer la vue correspondant à la demande.

Cela simplifie la tâche du développeur qui tenterait d'effectuer une maintenance ou une amélioration sur le projet. En effet, la modification des traitements ne change en rien la vue. Par exemple on peut passer d'une base de données de type SQL à XML en changeant simplement les traitements d'interaction avec la base, et les vues ne s'en trouvent pas affectées.

Choix de l'architecture

Avant de commencer le développement de notre application, nous avons pris le temps de réaliser une première version des IHMs à l'aide d'un logiciel de graphisme Cacoo. Cette première version ayant pour but de définir à chaque membre du projet le résultat souhaité. Cette première version nous a aussi permis de valider l'architecture et le rendu visuel avec la principale concernée par l'utilisation de cette application.

L'application sera composée d'une page d'identification qui après authentification de l'utilisateur, mène à la page principale de l'application. Sur cette page principale, on retrouve une liste d'étudiants absents. Ces paramètres sont, le choix d'une promotion, d'un groupe (ou pas) et d'une période. La vue permet de visualiser les étudiants absents sur une période, si l'on sélectionne un étudiant absent on est redirigé sur une autre page. Cette autre page permet de visualiser toutes les absences de cet étudiant. Elle permettra aussi de justifier une absence.

Sur la page principale on trouve aussi trois boutons. Ces boutons sont des liens vers d'autres pages. La page emploi du temps (utilisée presque exclusivement par le secrétariat), la page gestion des étudiants et la page paramétrage.

La page gestion emploi du temps permet au secrétariat de renseigner la base de données sur les créneaux libres (les créneaux où il n'y a ni cours, ni travail personnel). Sur cette page on trouve tout d'abord trois menus déroulants, pour le choix d'une promotion, d'un groupe et d'une période. Il suffira par la suite d'enregistrer les créneaux libres pour qu'ils soient ajoutés automatiquement à la base de données.

La page gestion des étudiants est nécessaire à l'application pour assurer une mise à jour constante des données des étudiants. En effet, cette application est basée sur l'identifiant de la carte étudiante, hors il possible qu'un étudiant change de carte (perte, panne, vol,). L'identifiant de la carte associé à un étudiant est donc susceptible d'être modifié. De plus, il est possible qu'un étudiant arrête sa formation. Cela entraînerait donc une absence constante de cet étudiant. Il est donc nécessaire de pouvoir le supprimer de la base de données. Il est également possible qu'un étudiant soit rajouté à une promotion. Cette page permet donc d'ajouter une promotion à l'aide d'un fichier CVS et de modifier les données d'un étudiant.

La page paramétrage permettra de modifier des éléments de l'application.



Choix classe ▾

Choix groupe ▾

Période ▾

Emploi du temps

Gestion étudiant

Paramétrage

LISTE D'ETUDIANT ABSENT

page d'accueil



Retour

Exporter fiche

Justifier absence

LISTE ABSENCE D'UN ETUDIANT

page absence

Retour	Sauvegarde	Choix classe ▾	Choix groupe ▾	Semaine ▾
Lundi	Mardi	Mercredi	Jeudi	Vendredi
8H15-10H15	8H15-10H15	8H15-10H15	8H15-10H15	8H15-10H15
10H30-12H30	10H30-12H30	10H30-12H30	10H30-12H30	10H30-12H30
14H00-16H00	14H00-16H00	14H00-16H00	14H00-16H00	14H00-16H00
16H15-18H15	16H15-18H15	16H15-18H15	16H15-18H15	16H15-18H15

page gestion emploi du temps

Retour	Nouvelle liste étudiant	Ajouter étudiant	Supprimer étudiant	Modifier étudiant
<div style="background-color: #ffffcc; padding: 10px;"> <div style="display: flex; justify-content: space-around; align-items: center;"> <div style="flex: 1;"> <input placeholder="Nom" type="text"/> <input placeholder="Numéro étudiant" type="text"/> </div> <div style="flex: 1; text-align: right;"> Supprimer étudiant Annuler </div> </div> <hr/> <div style="display: flex; justify-content: space-around; align-items: center;"> <div style="flex: 1;"> <input placeholder="Choisir liste" type="text"/> <input placeholder="Classe" type="text"/> </div> <div style="flex: 1; text-align: right;"> Ajouter liste Annuler </div> </div> </div>				
<div style="background-color: #ffffcc; padding: 10px;"> <div style="display: flex; justify-content: space-around; align-items: center;"> <div style="flex: 1;"> <input placeholder="Nom" type="text"/> <input placeholder="Prenom" type="text"/> <input placeholder="Numéro étudiant" type="text"/> <input placeholder="Classe" type="text"/> <input placeholder="Groupe" type="text"/> </div> <div style="flex: 1; text-align: right;"> Ajouter étudiant Annuler </div> </div> <hr/> <div style="display: flex; justify-content: space-around; align-items: center;"> <div style="flex: 1;"> <input placeholder="Nom" type="text"/> <input placeholder="Prenom" type="text"/> <input placeholder="Numéro étudiant" type="text"/> <input placeholder="Classe" type="text"/> <input placeholder="Groupe" type="text"/> </div> <div style="flex: 1; text-align: right;"> Modifier étudiant Annuler </div> </div> </div>				

page gestion étudiant

L'application réalisation

Tout d'abord sur la page d'accueil, on retrouve la quasi totalité des fonctions prévues, la liste des étudiants absents, la possibilité de justifier les absences et les boutons emploi du temps, gestion étudiant et paramétrage. En revanche, nous n'avons pas eu le temps d'intégrer les filtres pour la visualisation de la liste d'absences.

Sur cette page nous avons la possibilité de justifier une absence. Pour justifier une absence il suffit de cliquer sur le bouton justifier et une fenêtre pop-up apparaît pour valider la justification.

Absences

Actions	Nom	Prénom	Promotion	Date	Justification	Actions
Emploi du temps	AUDRERIE	Claire	DII3-G2	10:37 Tue 06 Jan 2015	Non	Détail Edit Justifier
Gestion étudiant	CHALOPIN	Quentin	DII3-G1	14:05 Fri 06 Feb 2015	Oui	Détail Edit Justifier
Paramétrage	RAILLERE	Simon	DII3-G1	08:36 Fri 06 Nov 2015	Non	Détail Edit Justifier

page accueil

Absences

Actions	Nom	Prénom	Promotion	Date	Justification	Actions
Emploi du temps	AUDRERIE	Claire	DII3-G2	10:37 Tue 06 Jan 2015	Non	Détail Edit Justifier
Gestion étudiant	CHALOPIN	Quentin	DII3-G1	14:05 Fri 06 Feb 2015	Oui	Détail Edit Justifier
Paramétrage	RAILLERE	Simon	DII3-G1	08:36 Fri 06 Nov 2015	Non	Détail Edit Justifier

La page à l'adresse 85.69.7.7 indique :
Etes-vous sûr de vouloir justifier l'absence?

Annuler OK

validation justification

Absences

ⓘ L'absence est déjà justifiée

Actions	Nom	Prénom	Promotion	Date	Justification	Actions
Emploi du temps	AUDRERIE	Claire	DII3-G2	10:37 Tue 06 Jan 2015	Non	Détail Edit Justifier
Gestion étudiant	CHALOPIN	Quentin	DII3-G1	14:05 Fri 06 Feb 2015	Oui	Détail Edit Justifier
Paramétrage	RAILLERE	Simon	DII3-G1	08:36 Fri 06 Nov 2015	Oui	Détail Edit Justifier

absence justifier

Pour la page Emploi du temps, nous avons tout d'abord une pages qui liste tous les créneaux libres déjà renseignés. Sur cette page nous avons la possibilité d'ajouter un créneau libre, d'édition un créneau, de le visionner ou de le supprimer. En revanche, nous n'avons pas eu le temps de mettre en place un calendrier avec des créneaux prédéfinis comme prévu lors de la définition des IHMs.

The screenshot shows a table listing three free slots:

Actions	Promotion	Heure De Début	Heure De Fin	Actions
Nouveau Créneau libre	DII3-G1	16/06/15 16:00	16/06/15 18:00	View Edit Delete
	DII4-G1	17/06/15 10:15	17/06/15 12:30	View Edit Delete
	DII5-G1	18/06/15 08:15	18/06/15 10:15	View Edit Delete

liste des créneaux libres

Pour l'ajout d'un créneau libre il suffit de choisir une promotion, la date de début et la date de fin. Un appui sur la touche Submit permet d'enregistrer le créneau libre dans la base de données.

The screenshot shows a form for adding a new slot:

- Groupe ***: A dropdown menu for selecting a group.
- Date de début ***: A date picker set to June 21, 2015.
- Date de fin ***: A date picker set to June 29, 2015.
- SUBMIT**: A brown rectangular button at the bottom right.

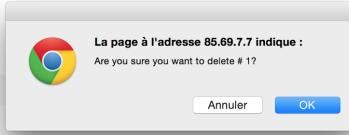
ajout créneaux libre

Pour supprimer un créneau libre, il faut appuyer sur le bouton delete. Une fenêtre pop-up apparaît pour valider la suppression. Il est aussi possible de modifier un créneau à l'aide de la page edit.

Créneaux

Créneaux libres

Actions	Promotion		Actions
Nouveau Crneau libre	DII3-G1	17/06/15 10:15	View Edit Delete
	DII4-G1	18/06/15 08:15	View Edit Delete
	DII5-G1	18/06/15 10:15	View Edit Delete



[suppression créneaux libres](#)

Sur la page de gestion étudiant, nous avons la possibilité d'ajouter un étudiant, de visionner une fiche étudiante, d'éditer une fiche ou de supprimer un étudiant de la liste. En revanche, l'insertion d'une liste d'étudiant via un fichier CSV n'est pas fonctionnelle. En effet, il est très difficile d'ouvrir un explorateur de fichier depuis un site web. Cela implique donc que l'utilisateur de l'application devra remplir manuellement tous les étudiants d'une promotion.

Etudiants

Actions	N° Étudiant	Id Carte	Prénom	Nom	Promotion	Actions
New Etudiant	1	11111111111111	Maxime1	GIRARD1	DII5-G1	View Edit Delete
Importer fichier de synchronisation des cartes étudiantes	2	11111111111112	Quentin	CHALOPIN	DII3-G1	View Edit Delete
	3	11111111111111	Simon	RAILLERE	DII3-G1	View Edit Delete
	30312569	04693D7AA63A80	Simon	RAILL	DII3-G1	View Edit Delete
	4	11111111111111	Kevin	REPILLEZ	DII3-G2	View Edit Delete
	5	11111111111122	Claire	AUDRERIE	DII3-G2	View Edit Delete

< previous next >

1 of 1

[page étudiante](#)

Etudiants

Actions

List Etudiants

Add Etudiant

Id carte *

Prénom *

Nom *

Promotion *

SUBMIT

ajout étudiant

Etudiants

Actions

Edit Etudiant

Delete Etudiant

List Etudiants

New Etudiant

AUDRERIE Claire

Numéro d'étudiant
5

Numéro de carte
111111111112

Promotion
DII3-G2

Liste des absences

Date	Justification	Actions
10:37 Tue 06 Jan 2015	Non	Justifier

visualisation étudiant

Sur la page visualisation nous avons rajouté la visualisation de la liste des absences de l'étudiant. Cela permet donc avoir un récapitulatif complet de l'étudiant.

Etudiants

RAILLERE S

Actions

- Edit Etudiant
- Delete Etudiant**
- List Etudiants
- New Etudiant

Numéro d'étudiant
3

Numéro de carte
11111111111111

Promotion
DII3-G1

La page à l'adresse 85.69.7.7 indique :
Are you sure you want to delete # 3?

Annuler OK

Liste des absences

Date	Justification	Actions
08:36 Fri 06 Nov 2015	Non	Justifier

suppression étudiant

Comme pour les autres actions de suppression présentées précédemment, une fenêtre pop-up apparaît pour confirmation.

Partie RFID

Choix technologiques

Etude comparative des nano-ordinateurs

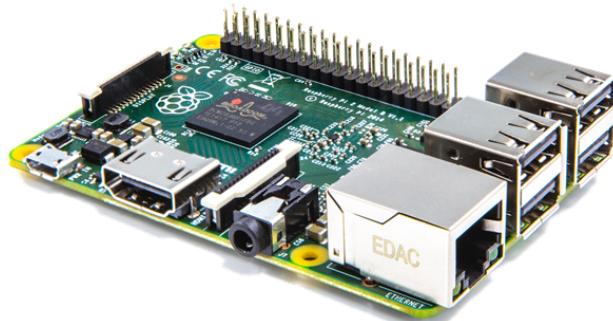
Pour la réalisation du boîtier transportable, nous avons pensé à utiliser un nano-ordinateur. Nous avons donc étudié les principaux nano-ordinateurs présents sur le marché. Nous les avons sélectionnés en fonctions de nos besoins. En effet, il nous fallait une prise Jack ou un port USB afin de pouvoir alimenter le système. Nous avions besoin aussi d'un USB afin de pouvoir connecter le lecteur de carte RFID. Nous souhaitions aussi un équipement qui dispose de quelques ports d'entrées-sorties afin de pouvoir faire évoluer le système en y ajoutant par exemple des leds pour indiquer l'état du système ou interrupteur pour éventuellement assurer la synchronisation avec la base de données.

	Rasp. Pi 1 A+	Rasp. Pi 1 B+	Rasp. Pi 2 B	ODROID U3	ODROID -XU3	Cubie- Board2	Beagle- Bone black	Pc Duino3 (arduino)	Banana pi
Ports USB	1	4	4	3	6	2	1	2	3
Pins GPIO	40	40	40	12	30	67	92	22	26
Port HDMI	-	1	1	1		1	1	1	1
Port Ethernet	-	1	1	1	1	1	1	1	1
Jack 3.5mm	Oui	Oui	Oui	Oui	Oui	Oui	Oui	Oui	Oui
Carte SD	Micro SD	Micro SD	Micro SD	Micro SD	Micro SD	Micro SD	Micro SD	Micro SD	Micro SD
Alim	5V- 1,2A	5V - 1,2A	5V - 1,2A	5V - 2A	5V - 4A	5V-2A	5V - 210-460 mA	5V - 2A	5V - 2A
OS	Raspbian	Raspbian	Raspbian	Ubuntu, Android 4	Ubuntu, Android 4	Debian, Android	Linux, Android	Linux, Android	Linux
Taille	66 x 56 mm	85 x 56 mm	85 x 56 mm	83 x 48 mm	94 x 70 mm	100 x 60 mm	86 x 53 mm	121 x 65 mm	92 x 60 mm
Prix	20,70 €	21,36 €	35,30 €	61,37 €	159,20 €	69,90 €	51,99 €	65,55 €	41 €

Nous avons finalement choisi de travailler avec la famille Raspberry. En effet, ceux sont les nano-ordinateurs les plus courants, ils disposent d'une grande communauté d'utilisateurs. Ce sont également les moins onéreux.

Plusieurs Raspberry Pi existent, la Raspberry Pi 1 A+, la Raspberry Pi 1 B+ et la Raspberry Pi 2 B.

Notre choix s'est alors porté sur l'utilisation d'une Raspberry Pi 1 B+. Cependant, compte-tenu des nano-ordinateur disponibles à Polytech nous avons travaillé sur une Raspberry Pi 2 B (voir figure ci-dessous).



Etude comparative des lecteurs RFID

Nous avons recherché les lecteurs de cartes RFID qui soient compatibles avec les cartes Atout'Centre. Nous avions donc besoin d'un lecteur RFID hautes-fréquences (13,56 Hz).

	Springcard Prox'N'Roll	String link SL500	MIFARE UID	PcProx (RFIDEAS)	MIFARE eReader
Type RFID	HF	HF	HF	HF - BF	HF
Normes Tags compatibles	ISO 14443 A ISO 15693	ISO 14443 A ISO 14443 B ISO 15693	ISO 14443 A	ISO 15693	ISO 14443 A
OS compatible	Windows Linux Mac (configuration sous windows)	Windows Vista et antérieurs	Windows Mac Linux	Windows Mac Linux	Windows Mac Linux
Interface	USB	USB	RS232	USB RS232 Ethernet	USB
Distance de lecture	< 5 cm	< 1cm	< 5 cm	< 7cm	< 10 cm

Prix	95,99 €	26,62 à 53,24€ suivant le modèle	75 €	156 €	46,60 €
------	---------	----------------------------------	------	-------	---------

Finalement, nous avons choisi de travailler avec le Springcard Prox'n'Roll car il est multi-plates-formes et qu'il peut s'interfacer en USB avec notre Raspberry. Sa distance de lecture est largement suffisante pour notre besoin (la carte Atout'Centre sera quasi au contact du lecteur). De plus, Polytech possède déjà ce lecteur, nous n'avions donc pas d'investissement à prévoir.



Réalisation

Réalisation logicielle du boîtier transportable

Choix du langage

Nous avons choisi de développer l'application en Java pour plusieurs raisons. Premièrement, l'existant qui nous a été donné (projet réalisé par des étudiants de PEIP) était codé en Java tout comme les programmes de tests proposés par le fournisseur.

D'autre part, Java est un langage orienté objets multi-plates-formes et gratuit. De plus, nous disposions déjà d'environnement de développements intégrés permettant de développer dans ce langage.

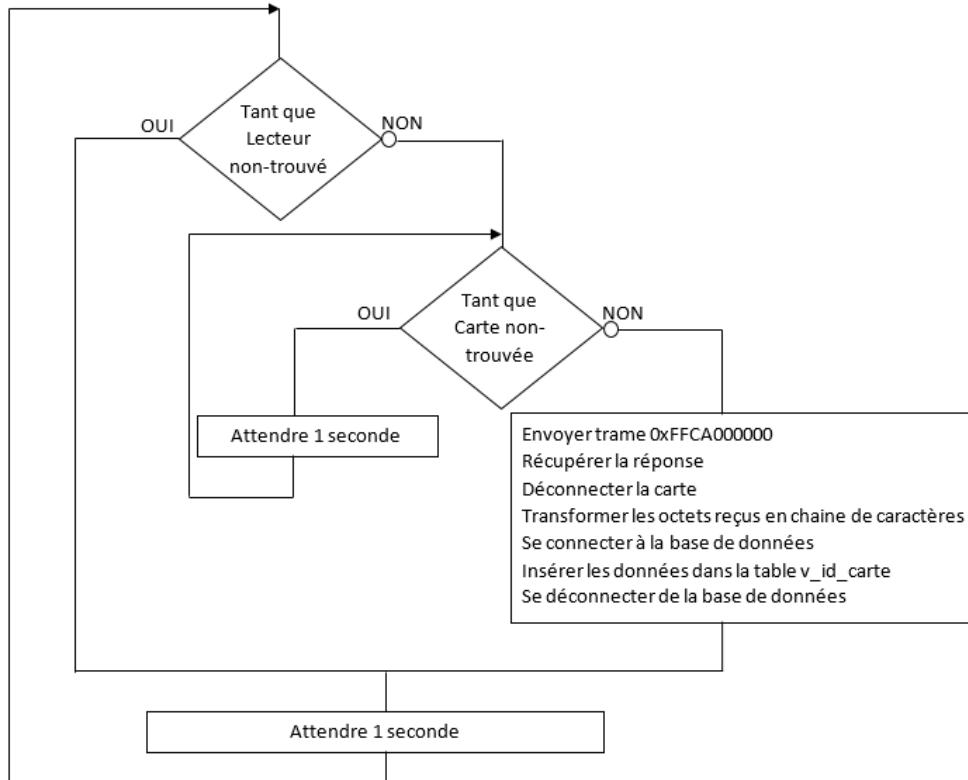
Principe de fonctionnement

Lors de l'ouverture de l'application, celle-ci est dans l'attente de la connexion d'un lecteur RFID. Pour cela, elle vérifie toutes les secondes s'il a eu connexion, ou non, d'un lecteur.

Une fois le lecteur connecté, l'application scrute toutes les secondes pour savoir si le lecteur a détecté une carte. Lorsque le lecteur peut lire une carte, une requête est envoyée dont la réponse permettra de récupérer l'identifiant de la carte.

L'identifiant, récupéré en byte, est alors transformé en chaîne de caractères.

L'application se connecte alors à la base de données et insère l'identifiant de la carte dans la table relative à l'émargement. L'application se déconnecte alors de la base de données.



Partie transmission de donnée

L'une des problématiques de notre projet est la transmission des données. Comment faire pour envoyer les informations d'émargements stockées dans la base de données du Raspberry vers le serveur contenant la base de données complète de notre application. Nous avions d'abord pensé à faire un script qui permet de se connecter à la base de données du Raspberry, faire une requête pour récupérer les ID des cartes qui ont émargé ainsi que la date d'émaragement. Puis de se connecter directement à la base de données du serveur et de lui insérer les données recueillies sur le raspberry.

Malheureusement, cette solution pose un problème de sécurité. En effet, comme nous l'a fait remarquer M.Esswein, quelqu'un pourrait très facilement réaliser une attaque de "man in the middle". C'est à dire que si quelqu'un se place au milieu de notre transmission et capture les paquets, il trouvera écrit en dur les informations de connexions à notre base de données, et ainsi en avoir complètement l'accès.

Pour contrer ce problème M.Esswein nous a proposé d'utiliser un serveur de sockets TCP.

Afin de développer le programme il n'y avait pas vraiment de restriction concernant le langage à utiliser. Le choix a donc été laissé à Simon, qui avait en charge cette partie du projet. Étant donné qu'il était plus à l'aise en langage C, le programme sera fait dans ce langage.

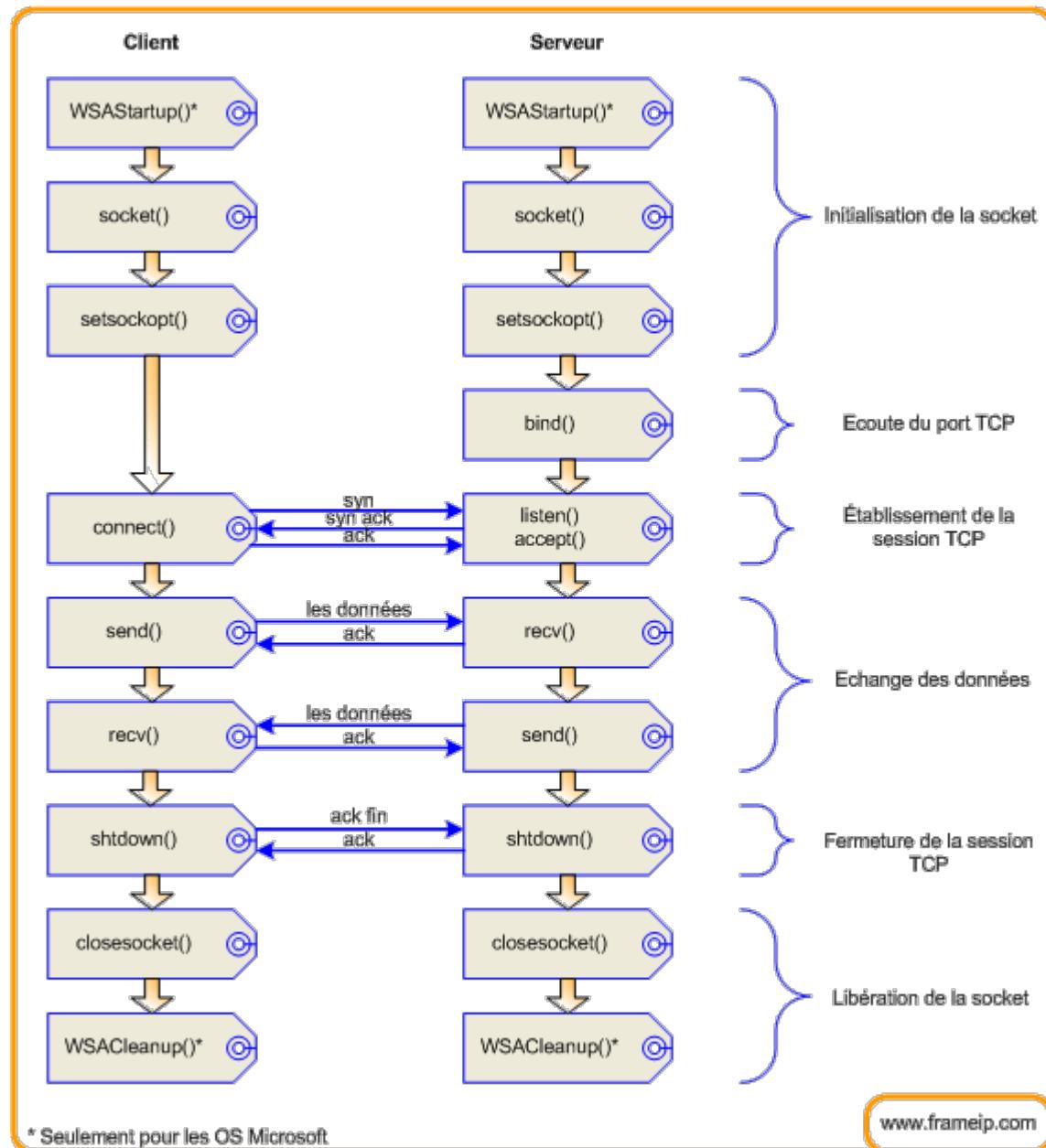
Les sockets fonctionnent comme expliqués sur le schéma ci-dessous, il y a un serveur et un client.

Sur le serveur il faut d'abord créer une socket, on l'initialise pour lui dire quelle connexions accepter sur quel port. Puis la socket attend qu'une connexion se fasse.

Sur le client, on crée aussi une socket et on l'initialise en lui disant à quel serveur se connecter et sur quel port. Puis on lui dit de se connecter.

Une fois le client connecté on peut envoyer et recevoir des informations d'un côté comme de l'autre.

Une fois le transfert de données terminé il faut fermer la connexion et détruire la socket.



Une fois la connexion entre le client et le serveur établie, il faut récupérer les données qui sont sur la base de données pour les envoyer.

Étant donné que nous utilisons une base de données MySQL nous avons utilisé l'API de MySQL.

Une fois la connexion initialisée et établie, il est assez facile d'envoyer nos requêtes SQL grâce à la commande "mysql_query" qui prend en paramètre la connexion que l'on veut utiliser puis la requête SQL.

La partie la plus compliquée à été de récupérer les données après la requête SELECT qui se retrouvent stockée dans un objet MYSQL_RES.

Pour cela nous avons créé une structure qui nous est propre, composée de deux tableaux de char contenant l'heure de l'émargement et l'ID de la carte. Puis nous avons créé un tableau de notre structure où chaque ligne correspond à une ligne de la base de données.

A partir du moment où nous avons réussi à stocker nos informations il ne nous restait plus qu'à les envoyer vers le serveur grâce à la commande "send" des sockets.

Dès que le serveur reçoit notre tableau de structure il le parcourt et crée la requête SQL à envoyer.

Pour simplifier la concaténation de la requête, nous n'envoyons pas une requête avec toutes les lignes du tableau mais une requête par ligne.

Pour finir, une fois que les données ont été envoyées nous effaçons les données du client afin de ne pas renvoyer plusieurs fois les mêmes informations d'émarginement au serveur.

Il nous a aussi fallu trouver un événement déclencheur pour lancer notre programme. Plusieurs choix s'offraient à nous : l'envoi des données lorsque le boîtier se connecte à internet, un envoi à intervalles réguliers ou lorsque que l'on appui sur un bouton.

Nous avons décidé de retenir cette dernière solution car cela laisse le choix à l'enseignant du moment auquel il veut transmettre la liste des étudiants qui ont émargés. Il peut de ce fait sanctionner les étudiants qui sont trop en retard.

Outils d'aides au projet



Pour pouvoir synchroniser l'ensemble des documents et du code qui a été produit tout au long du projet, nous avons utilisé le site web GitHub. GitHub est un réseau social basé sur le logiciel de gestion de versions décentralisées Git. Grâce à ce site, nous avons gardé l'ensemble des ressources à jour.

Bilans

Bilan personnel de Quentin Chalopin

C'était ma première expérience en tant que chef de projet et je dois dire que c'est une expérience très formatrice. J'ai dû me planifier et organiser le travail de mes camarades tout en travaillant sur les livrables.

De plus, j'ai beaucoup apprécié le contact relationnel avec toutes les personnes aussi bien internes qu'externes au projet. Je cite par exemple le service informatique de l'université avec lequel j'avais déjà eu des contacts. Ou encore les encadrants de notre projet dont l'échange d'idées a été constructif.

Malheureusement, certains échanges au sein de l'équipe ont été un peu plus soutenus à mon regret. Cet aspect fait aussi parti de la responsabilité d'un chef de projet.

J'ai également apprécié travailler sur le côté technique tout en restant chef de projet, c'est une attitude que j'aime adopter pour rester vraiment dans le concret et comprendre au mieux les fonctionnalités du projet. Ainsi, je peux voir les limites du projet et rebondir au mieux sur les bugs et les mauvais choix adoptés. Je suis conscient qu'agir comme cela n'est plus envisageable à un certain niveau de management.

Certes, le contrat n'a pas été rempli dans sa totalité mais je suis content de ce que l'on a construit. Nous avons une application avec une architecture solide et modulable. Par exemple, l'ajout d'une application mobile pour l'émargement est tout à fait envisageable. L'architecture de l'application web est telle qu'il est simple et peu chronophage d'ajouter de nouvelles fonctionnalités. C'est d'ailleurs là qu'un Framework PHP prend tout son intérêt.

Le temps passé sur le projet engloberait environ 80 heures.

Bilan personnel de Maxime Girard

Travaillant quotidiennement en entreprise sur une base de données Oracle SQL, j'ai été naturellement affecté par Quentin sur la partie base de données du projet. Il a aussi été convenu que j'aiderai Kévin à la réalisation de l'application si le besoin était.

Après avoir travaillé avec Kévin et Quentin à l'élaboration de l'analyse de l'application Web, je me suis chargé de toute la configuration de serveur qui nous a été mis à disposition par le service informatique de Polytech. Malheureusement, après avoir installé l'ensemble des composants nécessaire au bon fonctionnement du serveur, celui-ci a été supprimé par erreur par des sous-traitants du service informatiques, nous forçant à reconfigurer de nouveau l'ensemble de serveur. Par un soucis de sécurité pour la bonne tenue du projet, nous n'avons pas ré-hébergé sur un serveur de Polytech mais sur mon Raspberry Pi.

Une fois ces problèmes techniques passés, j'ai travaillé sur l'architecture de la base de données. Ce n'a pas été une partie extrêmement complexe étant donné mon expérience en entreprise mais l'utilisation de CakePHP m'a forcé à pratiquer une certaine rigueur sur l'appellation des tables.

Kévin, qui était préposé au développement de l'application, ayant des difficultés avec la compréhension de CakePHP, j'ai donc passé un temps important sur la compréhension du framework et sur le développement de l'application.

Sur la fin du projet, j'ai mis en place l'automatisation des tâches sur la base de données comme expliqué dans le dossier.

Ce projet a été très bénéfique pour moi, car il m'a permis de travailler sur beaucoup d'aspects différents comme la mise en place d'un serveur, la conception d'une base de données MySQL (Même si MySQL et Oracle SQL ont des ressemblances, il y a quelques différences que j'ai dû apprivoiser) ou encore le développement d'un site web, expérience totalement nouvelle pour moi.

Sur l'ensemble du projet, j'estime mon temps de travail à environ 60-80 heures.

Bilan personnel de Kevin Repillez

Ayant déjà un peu travaillé sur une application web lors du projet développement et souhaitant acquérir de nouvelles connaissances en web, je me suis proposé de travailler sur la partie application avec Maxime.

Avec l'aide de Quentin, Maxime et moi-même avons commencé la réalisation de ce projet par une analyse UML. Cela nous a permis tout d'abord, de mettre en application les connaissances acquises lors des cours d'UML. J'ai en suite, réalisé une première version des IHMs pour nous permettre de visualiser l'objectif à atteindre pour la réalisation de l'application.

Suite aux problèmes de serveur, nous avons pris un peu retard sur le démarrage du développement de l'application. De plus, lors de notre projet développement nous n'avions pas utilisé de framework et l'utilisation d'un framework était pour moi totalement nouveau. J'ai eu beaucoup de difficultés à me familiariser avec ce logiciel et j'ai donc demandé un peu d'aide à Quentin et Maxime, pour ne pas perdre trop de temps sur la réalisation du projet. Pour compensé le temps passé par Quentin et Maxime sur la partie qui m'était dédiée, j'ai passé un peu de temps sur la leur partie.

Ce projet m'a permis d'utiliser pour la première fois un framework et devoir comparer cette utilisation avec la réalisation d'une application web sans. J'ai aussi eu l'occasion d'utiliser phpmyadmin pour modifier et effectuer quelques tests sur la base de données.

Sur l'ensemble du projet, je pense avoir travaillé environs soixante heures.

Bilan personnel de Simon Raillere

Quentin Chalopin, qui avait le rôle de chef projet, m'a assigné la partie de la synchronisation des données.

Ayant été le premier à avoir le raspberry entre les mains, j'ai aussi eu la charge de le paramétrier. Ayant une bonne expérience personnelle sur linux, cela ne m'a pas posé de soucis particuliers.

Concernant ma véritable partie, j'ai un peu appréhender qu'on il a été décidé d'utiliser des sockets TCP pour communiquer entre le serveur et le raspberry. En effet nous avions bien sur vu le principe des sockets en cours de réseau, mais nous les avions jamais utilisé en TP.

Mais suite à une bonne recherche documentaire sur le web, j'ai pu me rassurer sur leur fonctionnement.

Le plus dur à été au final le stockage des données dans une structure qui nous été propre. En effet, comme expliqué dans le rapport il a été difficile de comprendre comment été retournées les données par l'API de MySQL. Heureusement, avec l'aide Quentin nous sommes finalement arrivé à nos fins.

Ce projet m'a permis d'appréhender l'utilisation d'un serveur de sockets TCP dont je n'avais aucune idée du fonctionnement avant. J'ai aussi pu utiliser l'API MySQL que je ne connaissais pas, n'ayant utilisé que des bases de données sous Oracle durant une expérience professionnelle ainsi qu'en cours l'an dernier.

Je regrette cependant de ne pas avoir eu plus de rapport avec les autres membres du projet. En effet, je récupère des données qui sont sur une base de données pour les insérer sur une autre.

Au final, je pense avoir passé sur ce projet une bonne cinquantaine d'heures.

Bilan personnel de Claire Audrerie

Il m'a été demandé de réaliser des études comparatives pour savoir quelles technologies nous allions utiliser. J'ai tout d'abord commencé par effectuer une étude comparative sur les nano-ordinateurs et ensuite une étude comparative sur les différents types de lecteurs RFID. Ces deux études m'ont permis de découvrir qu'il existe de nombreux équivalents au Raspberry sur le marché mais également d'enrichir mes connaissances sur les normes RFID, ce qui pourra m'être utile sur l'un de mes projets industriels.

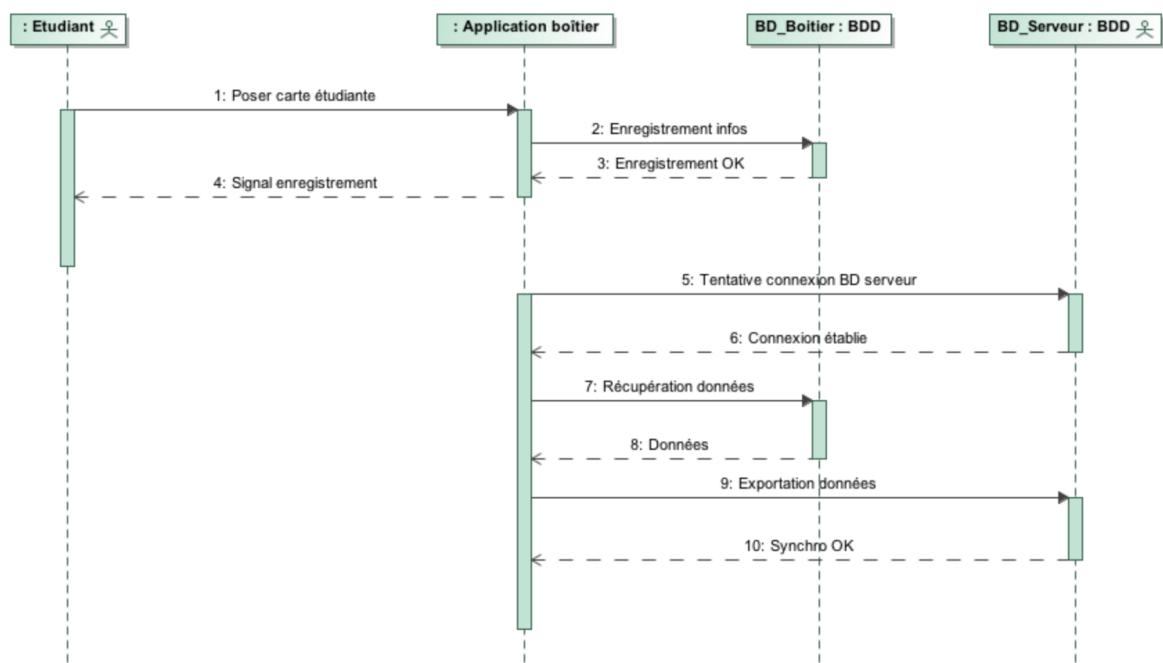
Ensuite, j'ai été amenée à comprendre le code fourni par les étudiants de PEIP et à étudier le code proposé par le fournisseur du lecteur RFID. J'ai ainsi pu me rendre compte que ce code, annoncé comme trivial par le fournisseur, n'était pas fonctionnel. Après de nombreuses heures passées à chercher la cause de ce non-fonctionnement, j'ai demandé à Quentin de prendre du temps pour m'aider à comprendre l'erreur et améliorer mes connaissances en Java.

Je regrette quelques difficultés de communication au sein de l'équipe qui résulte de la différence entre les manières de travailler de chacun des membres.

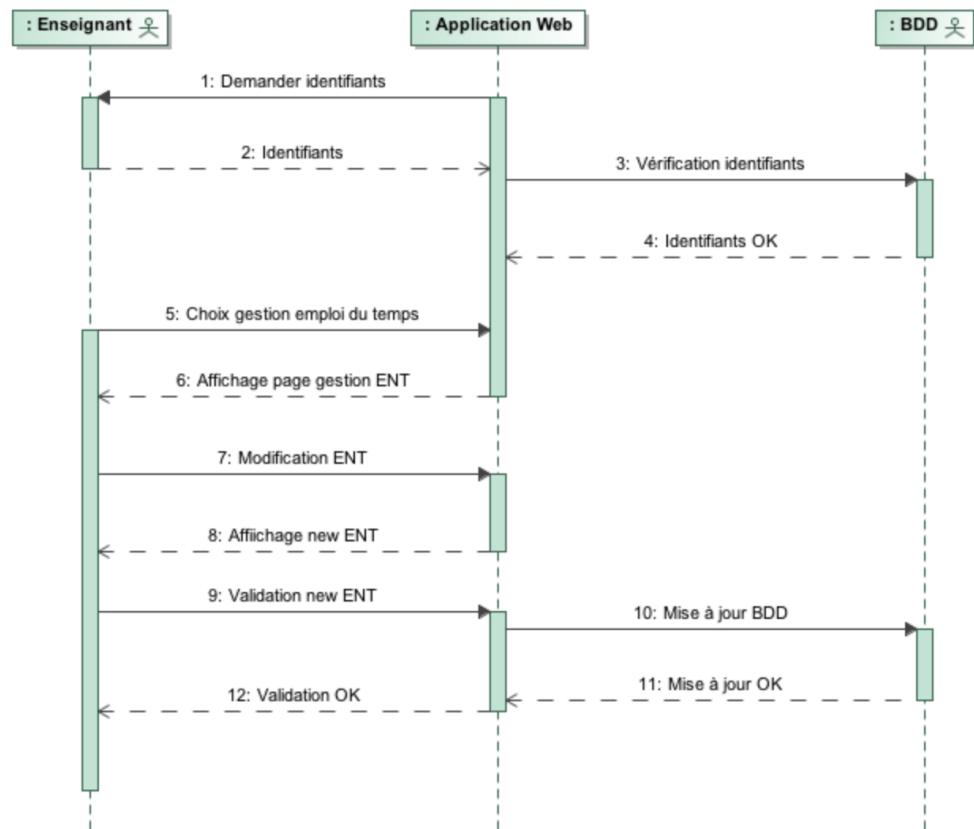
J'estime avoir travaillé une quarantaine d'heures sur le projet.

ANNEXES

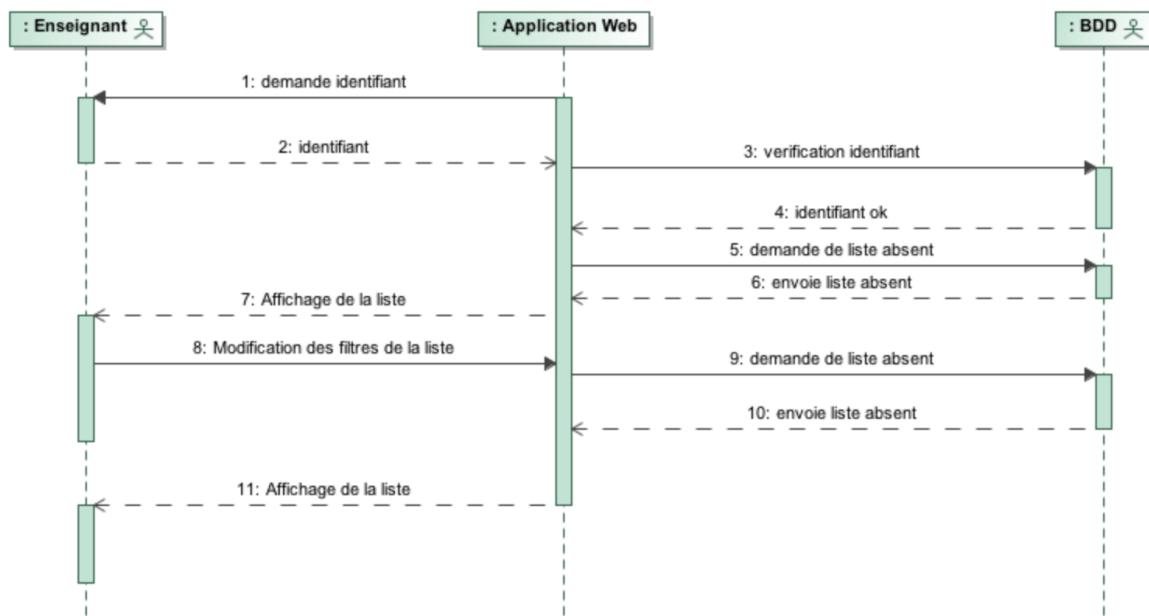
Emarger via boitier [Emarger via boitier]



Gestion emploi du temps [ Gestion emploi du temps]



Afficher liste absent [Afficher liste absent]



interaction Exporter absences d'un élève [ Exporter absences d'un élève]

