Projet de fin d'étude

Émargement électronique

Département Informatique, Microélectronique, Automatique

Polytech Lille, Villeneuve D'Ascq

Soutenu par :

Josué Rukata-Maroy

Vincenti Jean-Marie

Nom de l'école :

Polytech Lille

Tuteur Polytech Lille :

M. Thomas Vantroys

M. Alexandre Boé

Responsable Encadrant :

Mme Florence Geoffroy

M. Thomas Rougelot

Année : *2013/2014*

Remerciements

Nous tenons à remercier toutes les personnes qui nous ont permis d'effectuer ce projet de fin d'étude dans les meilleures conditions possibles.

Nous remercions et nous offrons toute notre reconnaissance aux personnes suivantes, pour l'expérience enrichissante et pleine d'intérêt qu'elles nous ont fait partager durant ces quatre mois de projet:

Nous tenons à exprimer notre profond respect et notre gratitude à Monsieur **Thomas Rougelot** et Madame **Florence Geoffroy** pour leur disponibilité et leur soutien tout au long de l'année.

Nous remercions également Monsieur **Thomas Vantroys** et Monsieur **Alexandre Boé** pour nous avoir donné l'opportunité de d'effectuer ce projet dans le cadre de notre enseignement.

Un grand merci, également, à l'ensemble des responsables d'année ainsi que les secrétaires de départements pour leur gentillesse et toute l'aide qu'ils nous ont apportée lors de ce projet.

Sommaire

Remerciements *1*

Introduction *3*

Contexte général *4*

Conception du système *6*

Présentation générale 6

La représentation des données 8

Présentation du fonctionnement *12*

Le fonctionnement de la tablette 12

L'interaction avec la base de donnée du serveur 20

L'interface d'administration 21

Sécurisation du système 26

Bilan de l'exercice pédagogique *27*

Conclusion *29*

Annexes *30*

Annexe 1 : Schéma UML de la base de données 30

Annexe 2 : Fonctionnement d'un « Controller » 31

Annexe 3 : Fonctionnement de la base de donnée Android 32

Annexe 4: Enchainement des activités Android avec les XML associés 33

Annexe 5 : Protocole de synchronisation 34

Annexe 6 : Description de l'API PHP 35

Annexe 7 : Code du fichier index.php 36

Annexe 8 : Fichier js/app.js 38

Annexe 9 : Contrôleur Angular 41

Annexe 10 : une VUE Angular 42

Introduction

Ce rapport présente notre travail effectué dans le cadre du projet de fin d'étude dans le département système communicant de l'école d'ingénieur Polytech'Lille. Ce projet a été proposé par M Thomas Vantroys et M Alexandre Boe. Il s'inscrit dans une démarche d'amélioration des enseignements en alternance grâce au développement d'un outil d'aide à l'émargement. Ce projet répond à un besoin réel. L'outil a été utilisé pendant la période de projet et continuera à l'être par la suite. Ainsi le système est voué à être transmis pour sa maintenance une foi le projet terminé

Ce rapport a donc deux objectifs. Il présente le travail réalisé tout au long du projet de fin d'étude pour un bilan de l'exercice pédagogique mais se présente également comme un rapport technique pour transmettre la maintenance du système aux responsables à venir. Nous nous efforcerons donc de vous présenter de façon claire ce projet et son fonctionnement tout en détaillant de façon précise sa structure.

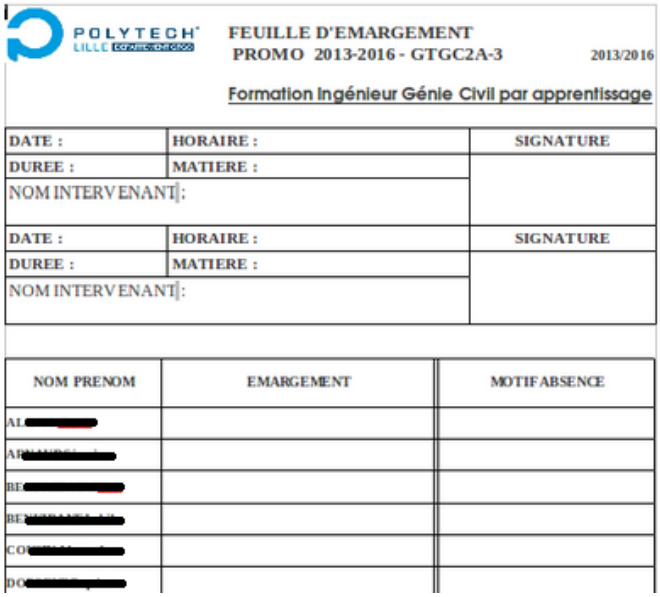
L'ensemble du travail demandé est la réalisation d'un système technique ainsi qu'un ensemble de rapports sous plusieurs formes. L'ensemble de ces documents ainsi que l'historique de développement est présent à cette page :

http://projets-imasc.plil.net/mediawiki/index.php?title=Emargement_%C3%A9lectronique

Contexte général

Les cursus en alternance des écoles d'ingénieur ont des contraintes propres qui traduisent un encadrement au niveau régional. L'une de ces contraintes et de fournir une preuve de la présence à chaque cours des élèves et des encadrants. Cette justification de présence est nécessaire pour justifier la rémunération des élèves tout au long de l'année. C'est donc à cette problématique que le sujet répond.

Actuellement, l'émargement entraine un traitement long et répétitif pour les secrétariats des départements concernés. Aujourd'hui la vérification des absences s'effectue par la signature systématique des élèves à chaque cour. Ainsi, les secrétariats doivent dépouiller chaque feuille d'émargement et ressaisir ces informations dans les outils informatiques existant. De même pour les bilans trimestriels qui doivent être fait « à la main ».



POLYTECH LILLE

FEUILLE D'EMARGEMENT
PROMO 2013-2016 - GTGC2A-3 2013/2016

Formation Ingénieur Génie Civil par apprentissage

DATE :	HORAIRE :	SIGNATURE
DUREE :	MATIERE :	
NOM INTERVENANT :		
DATE :	HORAIRE :	SIGNATURE
DUREE :	MATIERE :	
NOM INTERVENANT :		

NOM PRENOM	EMARGEMENT	MOTIF ABSENCE
AL[redacted]		
AR[redacted]		
BE[redacted]		
BE[redacted]		
CO[redacted]		
DO[redacted]		

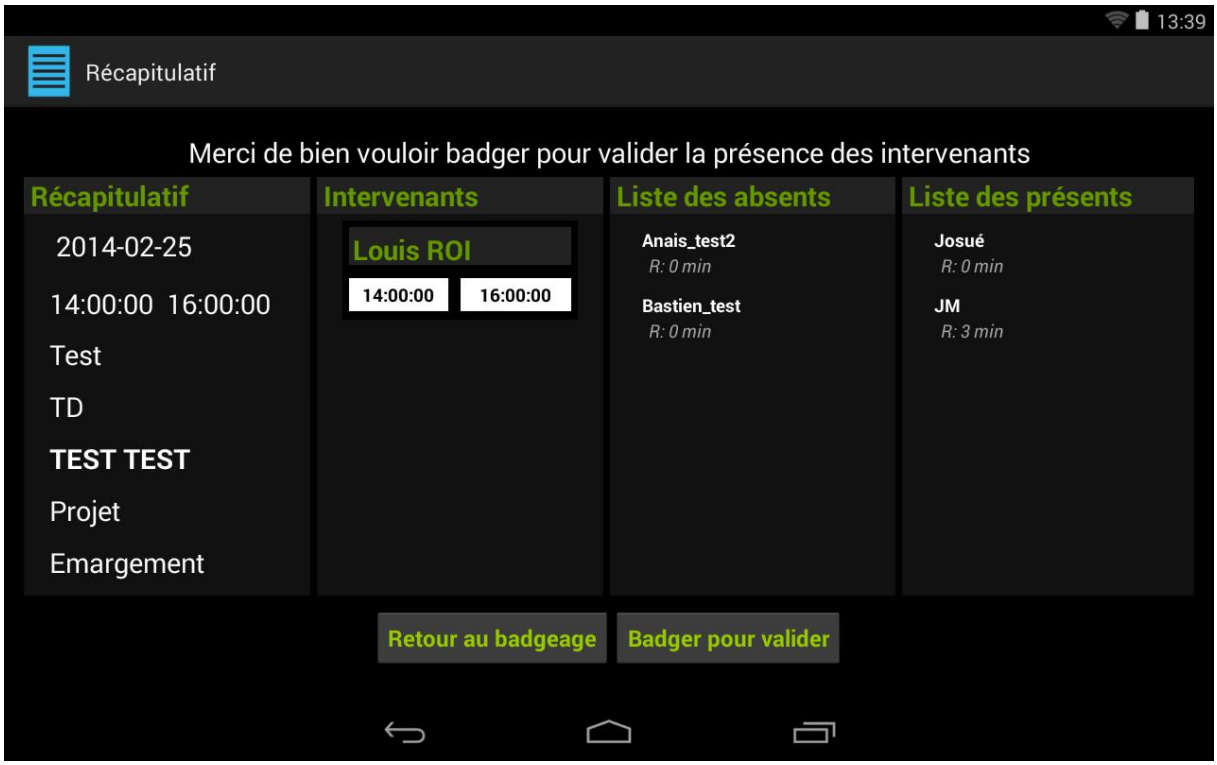
Ce processus est donc long, répétitif et est perçu aujourd'hui comme une perte de temps et d'énergie. C'est pourquoi il nous a été proposé de créer un outil capable d'automatiser ce processus d'émargement et de centraliser ces données pour pouvoir les utiliser avec d'autres outils informatiques existant.

Le projet proposé en tant que Projet de Fin d'Etude (PFE) aux étudiants est le suivant :

Aujourd'hui les cartes étudiantes et les cartes multiservices utilisées par les encadrants, sont munies d'une puce « sans contact », ou plus exactement une puce RFiD (de l'anglais radio frequency identification). L'objectif est de réaliser un service utilisant des tablettes Android capable de lire ces cartes RFiD.

Ces tablettes auront pour objectif de remplacer les « fiches papiers d'émargement » et permettront aux étudiants comme aux encadrants de « badger » avec leur carte pour justifier leur présence.

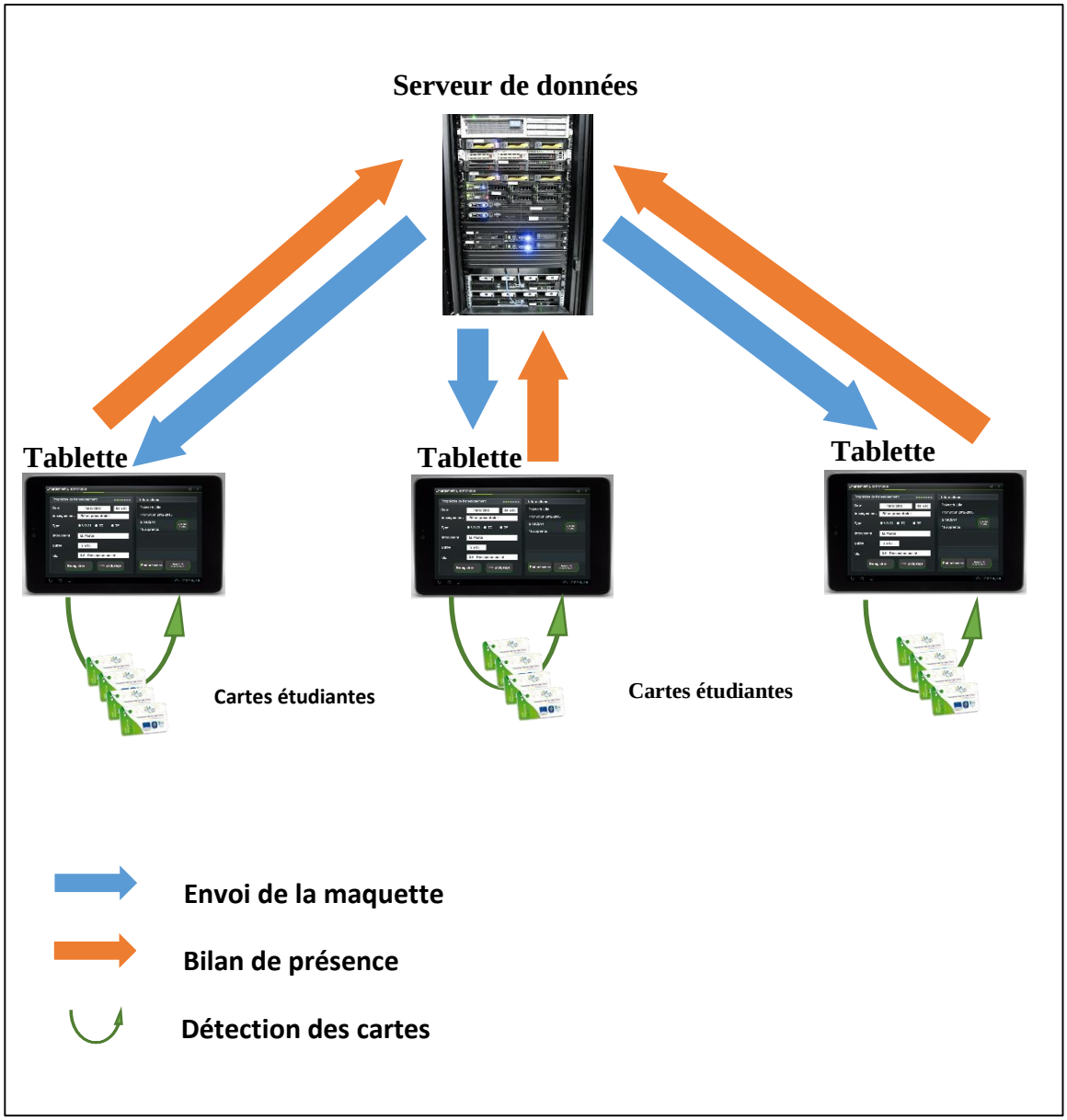
Une fois l'émargement terminé, les données de la tablette doivent être envoyé sur un serveur centralisé. L'objectif est d'offrir une visualisation en temps réel des absences et de regrouper toutes les informations nécessaires pour les utiliser dans d'autres outils existants.



Conception du système

Présentation générale

Le système a pour but d'effectuer un émargement des élèves et d'envoyer les données sur un serveur centralisé. Ainsi le système est composé de trois éléments physiques : un serveur de données, un ensemble de tablettes Android et les cartes étudiantes des élèves.



- Le serveur est l'unité sur lequel sera stocké l'ensemble des informations nécessaires pour effectuer un émargement ainsi que l'ensemble des fiches utilisées et qui seront utilisés. Ce serveur à deux objectifs. Premièrement il doit permettre au responsable d'enseignement de visualiser les données stockées en temps réel et donc de contrôler la présence des étudiants et des intervenants. Mais le serveur permet également de fournir les données nécessaires au bon déroulement de l'émargement sur tablette.

- Les tablettes remplacent les anciennes fiches papiers. Elles communiquent avec le serveur pour récupérer les listes d'étudiants, de cours, d'encadrants et tout ce qui est nécessaires pour effectuer l'émargement pour un cours donné. L'objectif était de proposer une solution au moins aussi rapide que la version papier.

- Les cartes étudiantes sont des cartes dite « sans contact » ou à communication en champ proche (en anglais near field communication, NFC). Elles possèdent, comme toutes les puces NFC, un identifiant unique. C'est cet identifiant unique que les tablettes sont capable de lire et utilisent pour identifier une seule et unique personne.

Ainsi le système fonctionne grâce à un ensemble de tablettes qui communiquent avec un même serveur. Le serveur envoi les données nécessaires pour effectuer le contrôle des présences et les tablettes renvoient les émargements terminés. Voici un schéma de principe illustrant le fonctionnement du système.

La représentation des données

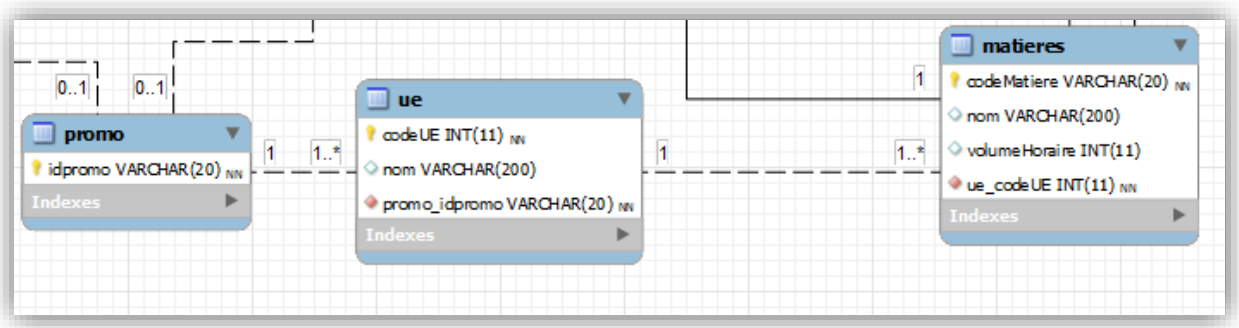
Une grande importance a été attribuée à la conception du système en lui-même. Nous avons effectué un grand nombre de réunions pour comprendre les besoins des futurs utilisateurs et surtout connaitre les informations nécessaires à la fois pour l'émargement mais également pour les divers besoins des secrétariats (présentation de bilan de semestres). Le résultat de cette conception peut être résumé par la présentation de la base de données suivante. Elle se présente en huit entités différentes et quatre tables de lien.

Un schéma UML global de la base de données est disponible en annexe 1.

Maquette des enseignements

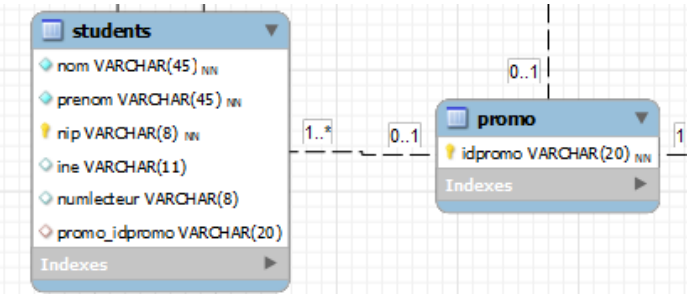
Ces informations correspondent à la maquette des enseignements, elles servent à améliorer l'expérience utilisateur en filtrant les informations en fonction de la promotion et matière adéquat.

- Une promotion est identifiée par une chaîne de caractères unique elle peut contenir plusieurs UE.
- Une UE (ou unité d'enseignement) est identifié par un code (un entier) et un nom unique. Elle appartient à une seule promotion. Une UE peut contenir plusieurs matières.
- Une matière est identifiée par son code (un entier) et une chaine de caractères. Un volume horaire est attribué (un entier) si besoin. Elle appartient à une seule UE.

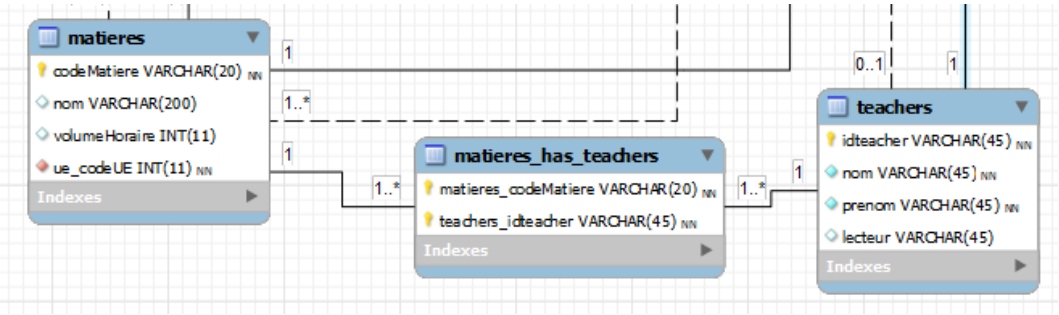


Etudiants et encadrants

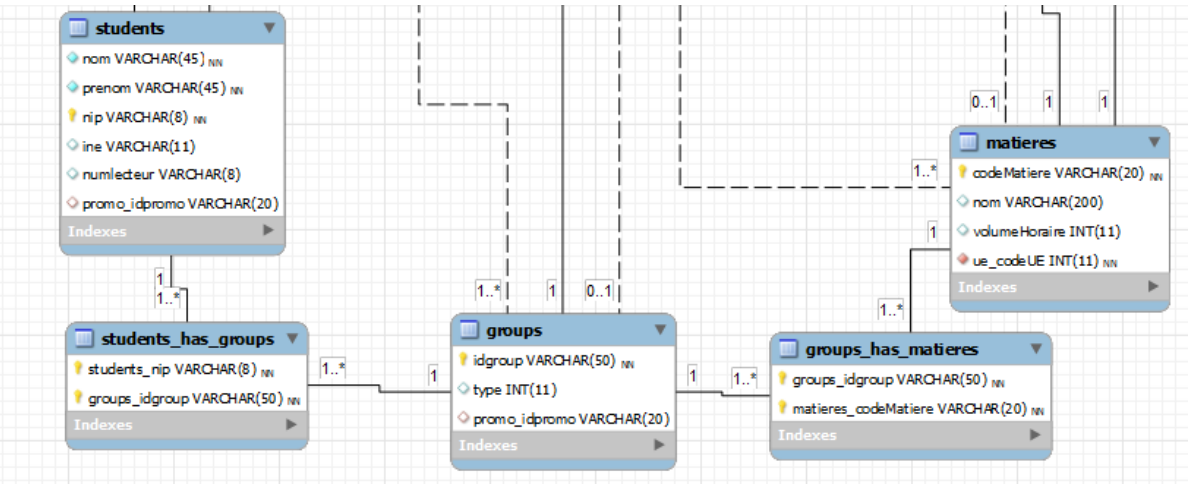
- Les étudiants sont composés de leur nom, prénom, leur numéro étudiant ainsi que le numéro d'identification de la carte qui est l'identifiant unique servant de signature sur la tablette. Les étudiants appartiennent tous à une seule promotion alors qu'une promotion peut contenir aucun ou plusieurs étudiants.



- Tout comme les étudiants, les encadrants sont composés de leur nom et prénom et le numéro de leur carte multiservice. Ils ont tous au moins une matière associée.

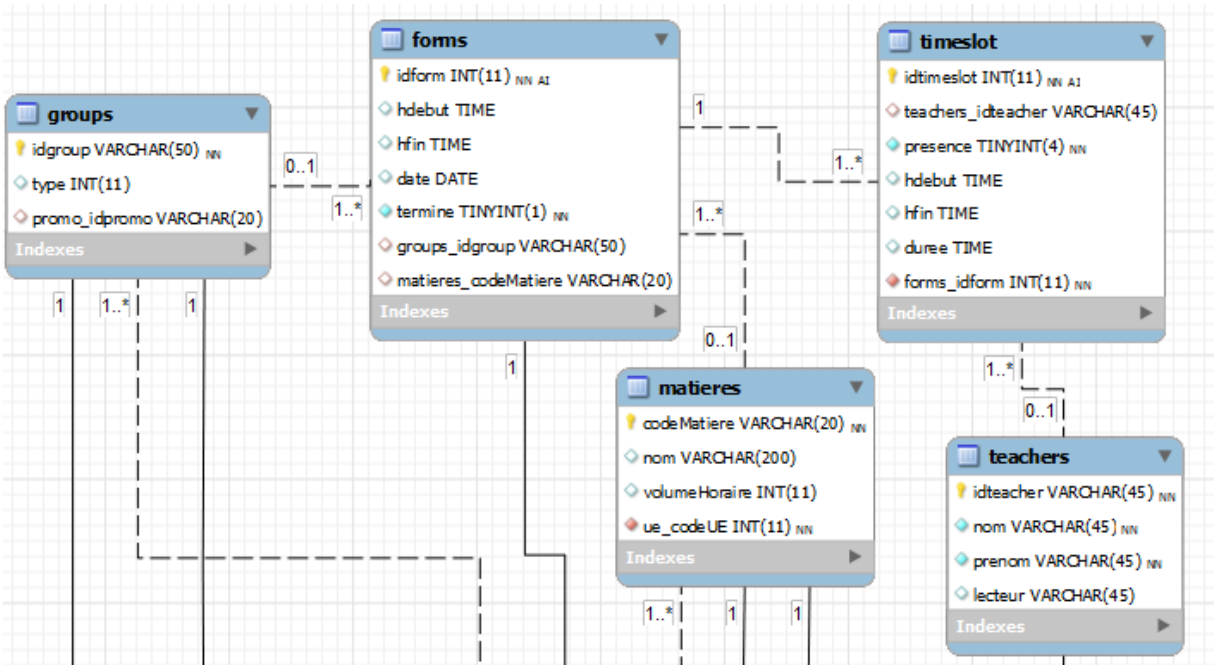


- Les groupes correspondent à des listes prédéfinis d'étudiants (pour les travaux pratiques ou dirigés) et sont liés aux matières associés. Un groupe contient au moins un étudiant et est lié à au moins une matière. Un groupe appartient obligatoirement à une unique promotion. Le type de groupe correspond à une répartition travaux dirigés, travaux pratiques, cours magistraux.



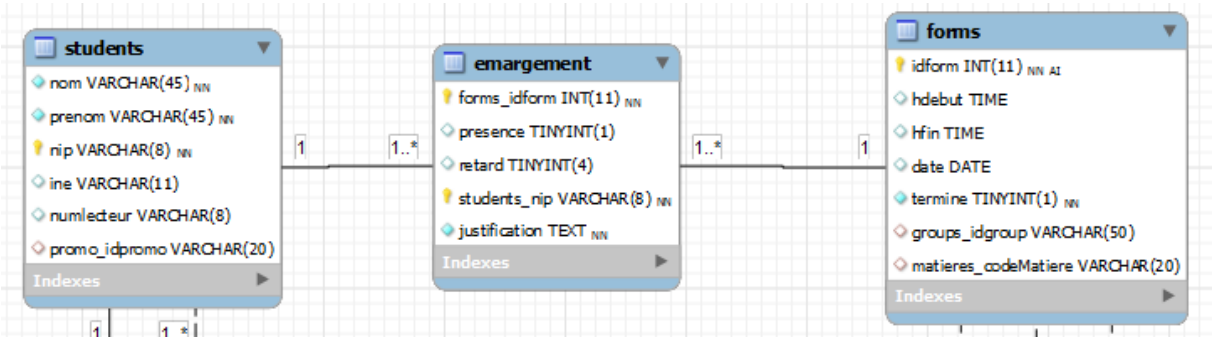
Les fiches d'émargement

- Une fiche rassemble les mêmes informations que les actuelles fiches papier. Elle est liée à une matière, à une date précise, avec une heure de début et une heure de fin. Les fiches sont liées à un groupe d'étudiants, ce lien est uniquement utilisé pour retrouver les fiches d'un même groupe. L'attribue « termine »correspond à son état dans la base de donnée (en attente, terminé, etc.), son utilisation sera présenté dans un autre chapitre.
- Pour répondre au besoin d'avoir plusieurs encadrants pour un même cours et à des heures différentes, les créneaux sont une entité unique. Ainsi les créneaux correspondent à la présence d'un encadrant pour une durée et à un cours donnés. Ainsi plusieurs encadrant peuvent être présent pour une même fiche, et à des heures différentes.



La signature des élèves

- La table d'émargement correspond à la signature des élèves pour une fiche donnée. Elle signale que l'étudiant est inscrit à une fiche et spécifie se présence ou non lors de son déroulement.



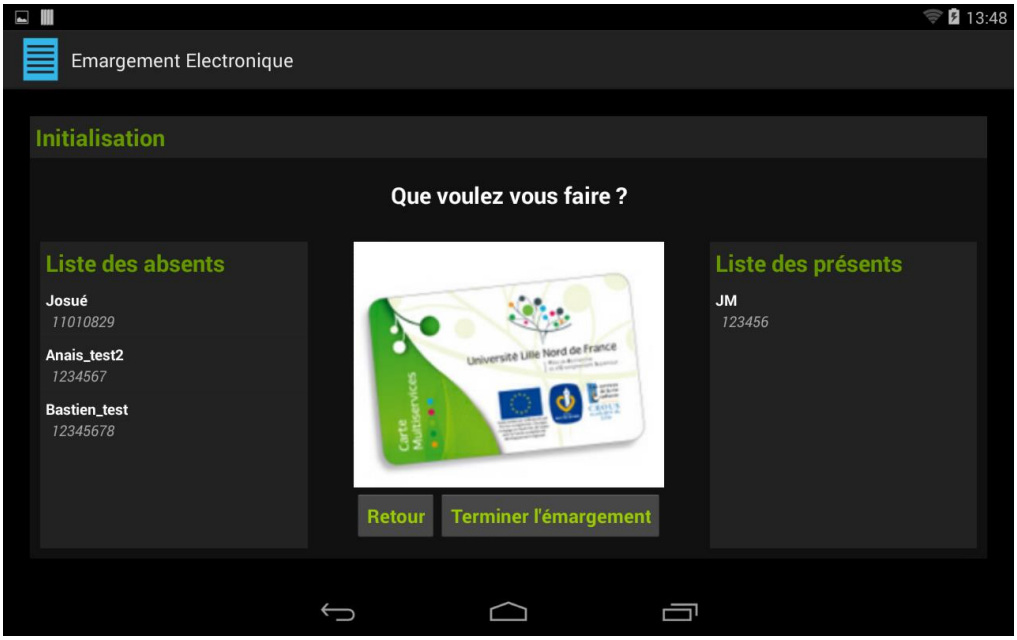
Présentation du fonctionnement

Le fonctionnement de la tablette

Présentation

L'application a été développée pour des tablettes avec le système d'exploitation Android, en technologies natives. Elle est donc conçu dans le langage orienté objet : le JAVA. Cette partie aura pour but de présenter le fonctionnement de l'application en général ainsi que la structure de façon précise.

L'objectif de l'application est de permettre d'effectuer un émargement en un temps inférieur sinon égal à un émargement papier. Les principaux avantages de ce système sont avant tout pour la gestion administrative qui suit ce contrôle de présence. Cependant il est important que l'outil reste intuitif et ne rajoute pas de charges et reste intuitif. La figure suivante correspond à un imprimé écran de l'initialisation d'une fiche.

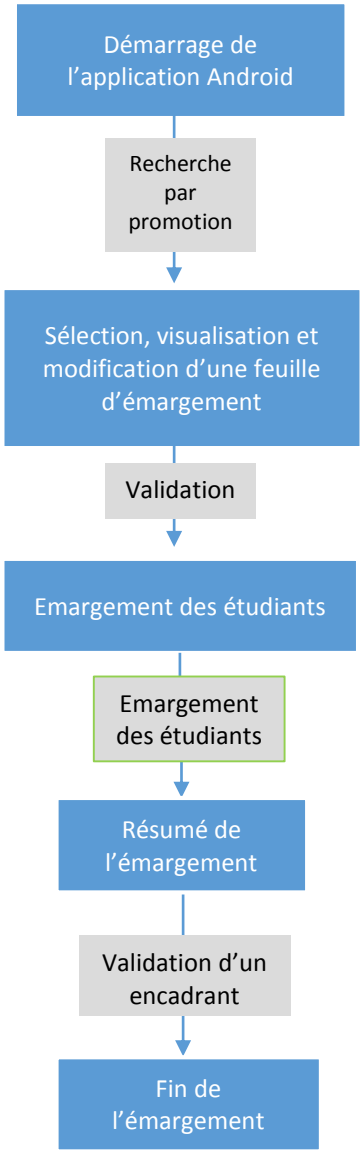


La grande force de l'application face aux fiches papier est que toutes les données sont logiquement proposées à l'utilisateur. Par exemple il n'est proposé que les encadrants correspondant à la matière suivi. Ainsi il est plus évident de remplir la fiche via l'application pour l'utilisateur et la cohérence des données est garanti.

Les étapes du processus d'émargement

Le fonctionnement de la tablette s'effectue en deux temps. Les données du serveur sont synchronisées manuellement par l'utilisateur. Lors de cette étape les informations nécessaires pour effectuer un suivi de présence sont téléchargées et stockées dans une base de données sur la tablette, identique à cette du serveur. Ainsi le deuxième temps est lié à l'émargement même.

Voici les étapes pour effectuer un émargement. Toutes ces étapes sont à répéter pour chaque cours, à n'importe quel moment de celui-ci.



L'utilisateur doit trouver ou créer une fiche d'émargement. Pour cela il doit renseigner le nom de sa promotion, l'UE recherche puis la matière associée pour voir les fiches disponible correspondante.

Une fois la fiche trouvé, l'utilisateur peut encore modifier les informations de la fiche (l'heure, la date, le ou les professeurs présents, etc.). Il peut également en créer une.

Quand ces informations sont validées, les étudiants peuvent bader pour signaler leur présence.

Quand les étudiants finissent de signer un par un, un écran récapitulatif est affiché. Le professeur peut vérifier la bonne présence des élèves. Lors de cette étape de contrôle, les encadrants peuvent modifier les informations de présence, attribuer un retard.

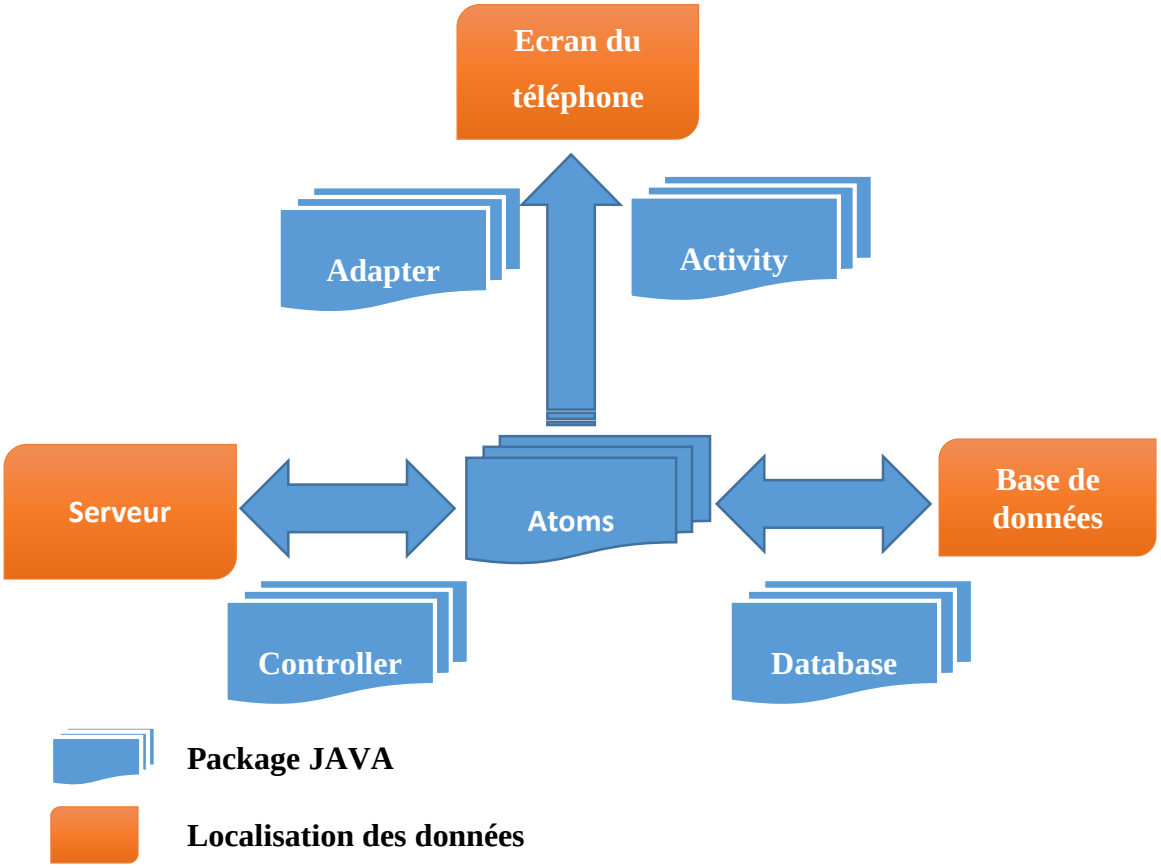
Si ces informations sont correctes, il peut ainsi valider la fin de l'émargement en passant sa carte à son tour et signaler sa présence.

Structures du code

D'un point de vue fonctionnel, le code JAVA est structuré en différents dossier (ou « package ») qui regroupe tous les objets en fonction de leurs rôles dans l'application. Voici comment sont organisés les objets.

- Les premiers dossiers « **Atoms** » et « **Specialsatoms** » rassemblent les 14 objets Java représentant les entités uniques, comme un élève ou une matière ainsi que certaine représentation de données combinés nécessaires pour améliorer l'exécution. Ces éléments correspondent aux **données brutes** que la tablette utilise pour effectuer l'émargement. De façon générale les objets correspondent tous à une table de la base de données.
- Les objets du package « **Database** » respectent le design pattern appelé DAO pour Data Access Object. Ce sont l'ensemble des objets nécessaires pour utiliser la **base de données** de la tablette.
- Le package « **Controller** » concentre les éléments nécessaires pour communiquer avec le **serveur**. Ils ont utilisés pour mettre en forme les données pour les envoyer correctement au serveur centralisé.
- Le suivant, « **Activity** », rassemble les activités Android. Ce sont les objets Java qui correspondent aux « **vues** » ou écrans de l'application.
- Le dernier, « Adapter », contient des objets qui dérivent tous de classes propres à Android. Ils servent à mettre en forme les objets pour les afficher dans des listes sur l'écran de la tablette. Ces objets qui suivent le pattern Adapter, permettent de mettre en forme des éléments d'une liste, par exemple une liste d'élèves.

Le schéma suivant illustre cette organisation par rapport à la localisation des données.



En annexe est présenté le fonctionnement d'un objet de type Controller (Annexe 2) et l'organisation du package Database (Annexe 3). Les objets de type « Activity » sont présenté dans la partie suivante.

Fonctionnement des activités Android

Une activité est la composante principale pour une application Android. Elle représente l'implémentation et les interactions des interfaces. D'un point de vue conceptuel, les activités se basent sur des fichiers XML où est décrite l'interface graphique. Tout comme le HTML avec le Javascript, ces éléments décrits dans les fichiers XML sont enrichies par les activités avec les données de l'application.

Dans la majorité des cas, on retrouvera un fichier XML par activité. On peut donc associer une vue à une activité. En annexe 4 est présenté l'enchainement possible entre les activités.

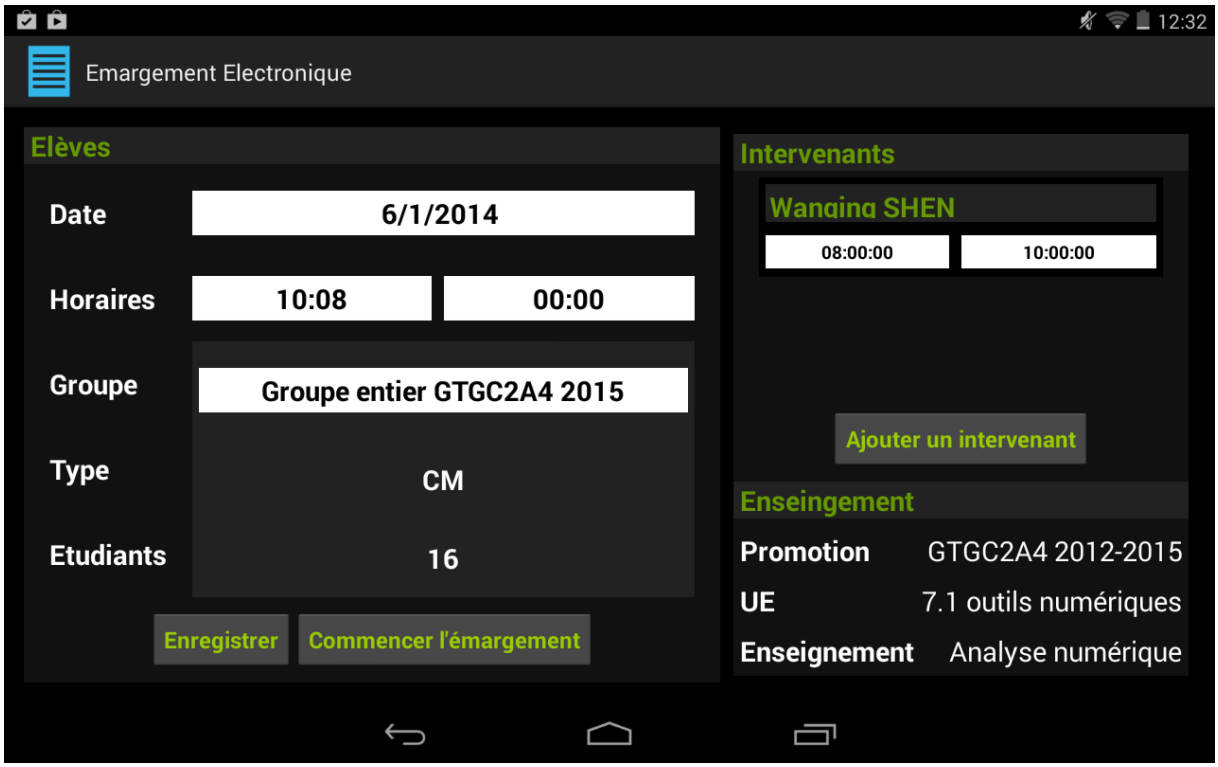
- SelectionActivity

Cette vue est la vue principale, celle qui est affiché lors du lancement de l'application. Elle se comporte comme un tableau de bord. Elle permet d'accéder aux différents menus (paramètre ou synchronisation) ainsi qu'à commencer un émargement. Dans le deuxième cas l'activité propose à la suite : la sélection de la promotion de l'étudiant, puis de l'UE concerné parmi les UE de la promotion et enfin la matière parmi les matières associés à cette UE. Dans chaque cas, dès qu'un élément est sélectionné, une requête dans la base de données est faite pour rechercher la liste d'objets (Ue, matière) en fonction des éléments sélectionnés précédemment.

Une fois la matière sélectionnée, l'application affiche les fiches d'émargement disponible pour cette matière. L'utilisateur peut soit en sélectionner une si elle convient, soit en créer une nouvelle, ce qui mène à l'activité suivante à qui est donné en paramètre l'identifiant de la fiche.

- InitializeActivity

A l'ouverture, cette activité récupère les informations dans la base de données en fonction l'identifiant de la fiche donné en paramètre. Si la fiche est nouvelle, les informations sont prés-rempli en fonction de l'heure et de la date actuelle. Dès que l'utilisateur a sélectionné ou créé une fiche il peut modifier ses informations. La figure suivante montre l'affichage des informations d'une fiche. On remarque que lors de la sélection du groupe, le type et le nombre d'étudiants sont automatiquement affiché.



Dans le cas d'un ajout d'un intervenant, l'utilisateur est invité à renseigner l'heure d'arrivée et de départ de la personne puis sélectionne son nom dans la liste généré en fonction de la matière (ou enseignement) associé à la fiche.

Les informations sont mises à jours dans la base de données quand l'utilisateur l'enregistre. Si une modification est faite, il ne peut commencer un émargement tant qu'il n'a pas sauvegardé ses modifications. De plus, l'utilisateur ne peut commencer tant que toutes les informations n'ont pas été renseignées.

Pour commencer un émargement, l'activité suivante est lancée avec l'identifiant de la fiche en paramètre.

- DetectionActivity

Cette activité a pour but d'effectuer l'émargement des étudiants. A son initialisation, elle assigne absent tous les étudiants. Dès qu'une carte NFC est détectée, l'application compare son identifiant avec les étudiants présents dans la base de donnée. Si l'étudiant existe, elle tente d'assigner présent cet étudiant pour cette fiche. Si l'étudiant est bien inscrit dans ce cours, il est marqué présent. Ainsi l'application peut distinguer le cas où l'identifiant de la carte ne correspond à aucun étudiant connu et celui de la détection d'un étudiant non inscrit au cours.

- ResultsActivity

Cette activité a trois objectifs distincts. Premièrement elle permet d'afficher un résumé de l'émargement terminé, avec les informations de la fiche (date, heure, matière, etc.). Elle permet également à l'intervenant de modifier à son appréciation la liste des personnes présentes (si un élève a oublié sa carte), attribuer un retard si nécessaire mais également signaler sa propre présence en utilisant sa propre carte.

Ainsi, cette vue est uniquement destiné aux encadrants qui doivent vérifier les informations, les modifier si nécessaire et d'émarger.

Pour valider définitivement cette fiche, et terminer le processus, l'encadrant doit impérativement valider avec sa carte. Ainsi une fiche est déclarée valide sous la responsabilité d'un encadrant.

- DownloadDatabaseActivity

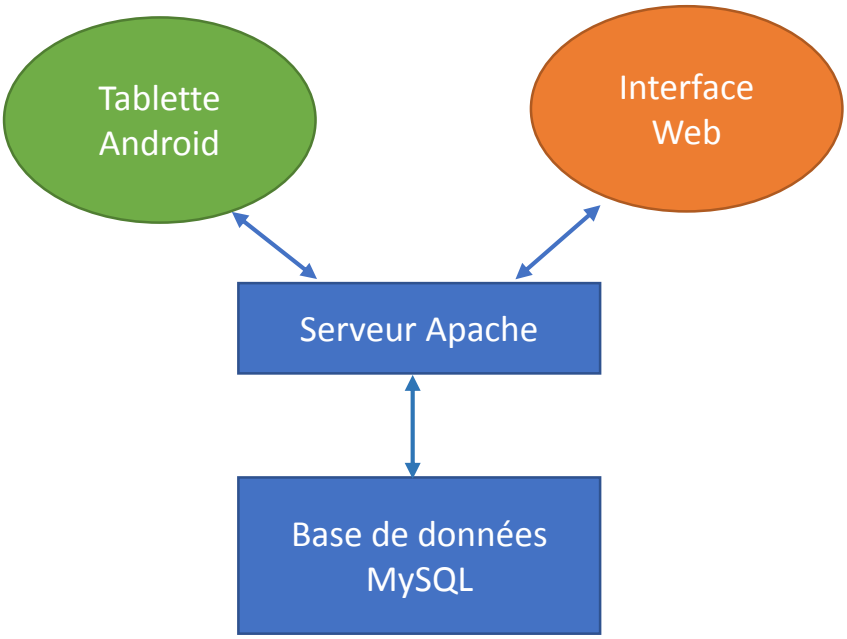
Cette activité ne peut être lancée que par le menu principal. Son objectif est de transmettre au serveur les fiches d'émargement terminées ainsi que de télécharger les informations nécessaires aux futurs contrôles de présence. D'un point de vue conceptuel, elle n'a pour unique but que de lancer une tâche asynchrone qui utilise l'ensemble des contrôleurs (voir l'annexe 2 sur le fonctionnement des « controller ») afin de mettre à jour table par table les données contenues sur la tablette.

Pour assurer la pérennité des données un protocole a été mis en place pour assurer que toutes les données ont bien été envoyées au serveur et assurer que les données présentes sur les tablettes sont complètes.

Le schéma en annexe 5 explicite ce protocole.

L'interaction avec la base de donnée du serveur

Que ce soit la tablette ou l'interface web, pour accéder à la base de données nous avons développé une interface de programmation (abr. API pour Application Programming Interface) qui est un ensemble de pages PHP qui convertissent les données de la base en objets JSON (un format de donnée standard).



Le rôle de ces pages PHP est donc de permettre à la tablette de communiquer avec la base de données. L'Annexe 6 décrit l'ensemble des pages PHP utilisées ainsi que les paramètres et résultats de chacune.

L’interface d’administration

Présentation

La saisie de la maquette (liste des élèves, des matières, etc.) ainsi que la préparation des fiches d’émargement s’effectue via une application WEB réalisée en Javascript.

L’interface a été pensée pour afficher de façon claire les étapes de remplissage des données sur le serveur. Cette étape d’importation de la maquette doit être faite en début d’année car tant qu’elle n’est pas terminée, l’émargement sur tablette ne sera pas possible.

Par ordre, voici l'état de remplissage des données :

1. **Promotions** : Dans cette rubrique, l'application donne à l'utilisateur la possibilité de créer une nouvelle promotion. Et au fur et à mesure que les informations relatives à la promotion seront ajoutées (élèves, matières, ….), il sera possible de visualiser les détails d'une promotion à partir de cette vue.

2. **Ue** : Sur cette page, l'utilisateur peut créer une nouvelle U.E. qui sera obligatoirement associée à une promotion. Il est alors nécessaire de se rassurer qu'une promotion est déjà créée avant de créer une nouvelle U.E.
   Cette vue liste toutes les U.E. de la base de données et offre la possibilité de supprimer ou de modifier une U.E. en cas d'erreur de frappe.

3. **Matière** : Cette rubrique offre la possibilité de créer une nouvelle matière associée à une U.E.
   On peut y visualiser toutes les matières de la base de données et réaliser l'action supprimer ou modifier sur les matières.

4. **Enseignants** : A ce niveau, on peut créer un enseignant et lui associer les matières qu'il dispense. On pourra également voir sur cette plage, la liste de tous les créneaux de l'enseignant pendant l'année.

5. **Étudiants** : C'est dans cette rubrique que l'application offre la possibilité d'ajouter un étudiant dans une promotion. Comme dans les autres vues, on pourra visualiser tous les étudiants de la base de données, et il sera possible de supprimer ou de modifier les données d'un étudiant.

6. **Groupe** : Une promotion sera divisée en plusieurs groupes. Soit les groupes de TP, TD ou langue. Cette page offre alors la possibilité de créer ces groupes, et d'y ajouter les étudiants de la promotion liée au groupe, et les matières associées à ce groupe. Sur cette page, on pourra visualiser toutes les fiches d'émargement associées à un groupe.

7. **Fiche** : A ce niveau, on peut créer, modifier et supprimer les fiches d'émargement.

Développement

L'interface d'administration a été développée en HTML/CSS et Javascript grâce à la librairie graphique AngularJS.
AngularJS est un Framework javascript libre et open-source JavaScript. Il adapte et étend le HTML grâce à des directives qu'on y inclut.

Il a pour but de simplifier la syntaxe Javascript, et de combler les faiblesses de Javascript en lui ajoutant de nouvelles fonctionnalités. Et ainsi faciliter la réalisation d'applications web monopages.

Nous avons fait le choix de ce framework car il permet d'organiser les données à l'écran de façon fluide, il accélère la vitesse de développement tout en gardant une grande liberté de conception et surtout car il traduit automatiquement les données JSON en objets Javascript. Ceci nous a permis d'avoir la même représentation de données JSON du côté WEB et Tablette.

Quelques Vocabulaires AngularJS

1. **Scope** : Les scopes d'AngularJS sont des objets qui servent de contexte d'évaluation des expressions contenues dans les templates (vue). Ils forment un arbre, dont la racine est le seul scope de l'application qui est aussi publié comme un service, sous le nom ***$rootScope***. Ce *$rootScope* est associé à l'élément contenant toute l'application AngularJS, celui sur lequel on met la directive *ngApp* : ça peut être l'élément <html> lui-même, ou le <body>, ou un à l'intérieur, peu importe.
   L'Annexe 8 et 9 expliquent la façon dont le scope est configuré sur AngularJS.

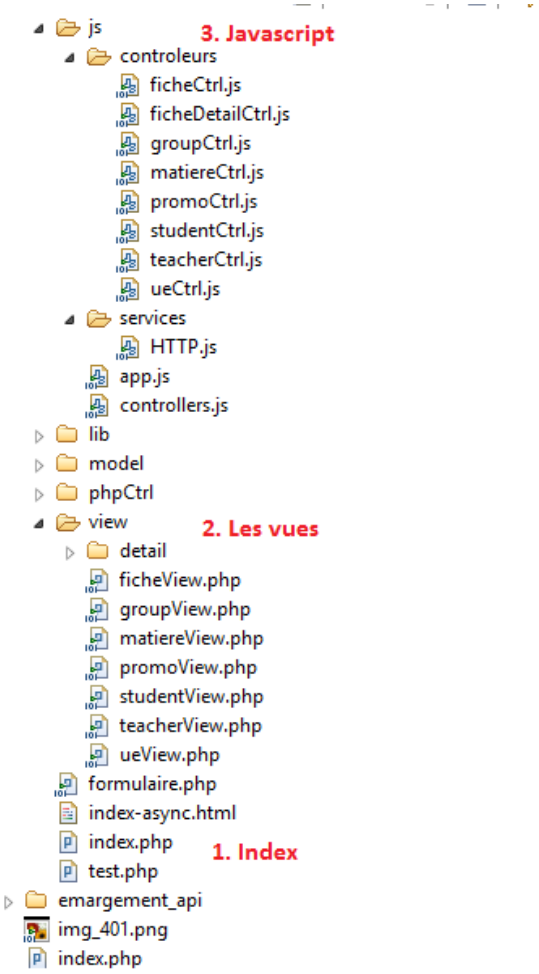
2. **Directive** : A un haut niveau, les directives AngularJS sont des marqueurs Angular contenus dans le code HTML sous forme d'attribut, nom d'un élément, ou encore d'une classe CSS. Une directive permet de signaler au compilateur AngularJS d'accorder un certain comportement à une balise HTML et ses fils (sous-balise). Pour AngularJS, la compilation signifie relier une balise HTML à un évènement. C'est grâce aux directives que ces liaisons sont réalisées.
   AngularJS est composé d'un ensemble de directive, parmi lesquelles on peut citer :
   - **Ng-model** : Cette directive lie un objet, javascript contenu dans le « scope » courant, à un élément du formulaire (balise <INPUT>).
   - **Ng-view** : elle permet d'inclure du code HTML dans une balise.
   - **Ng-repeat** : permet de faire une boucle pour afficher les éléments contenus dans un tableau déclaré dans le scope de la page.

   Vous trouverez un plus sur les directives à l'Annexe 10.

3. **Controller** : en AngularJS, un contrôleur est un constructeur javascript qui permet d'étendre les fonctionnalités et le comportement d'un scope. Un contrôleur est attaché au DOM grâce à la directive « **ng-controller** » d'angularJS. Le contrôleur est utilisé pour initialiser le contexte de la page, ou d'une partie de la page HTML à partir de la balise où il est défini. La description du contrôleur se trouve en Annexe 9.

4. **Services** : les services sont des objets singletons ou des fonctions qu'AngularJS met en place pour fournir des tâches communes à tous les contrôleurs. Tous les services Angular commence par un « $ ».
   Par exemple, $http est un service AngularJS qui fournit l'accès à l'objet XMLHttpRequest du navigateur pour réaliser des requêtes AJAX.

5. **Filter** : Les filtres sont des fonctions AngularJS qui permettent de formater la valeur d'une expression à afficher à l'utilisateur. Ils peuvent être utilisés directement dans une vue HTML, dans un contrôleur javascript ou encore dans un service.
   Voir les détails sur les filtres à l'Annexe 8



Cette image réprensente l'arborescence du site.

1. L'index : l'index, la page principale du site, se trouve à la racine de l'application. Ce fichier intègre tous les fichiers Javascript (contrôleur, services, librairie Angular) et CSS, et sont tous chargés dans le navigateur au lancement de l'application.
Voir l'Annexe 7 pour plus de détail.

2. Les vues : le dossier « view » contient toutes les vues de l'application. Ces fichiers ne sont pas directement affichés dans le navigateur, ils sont d'abord insérés dans le fichier index.php avant d'être visualisé. Ces insertions sont réalisées grâce à la technique d'injection qu'offre angularJS par la directive «**ng-view** » disponible dans la librairie « angular-route.js » et dont l'ordre d'insertion de page est configuré dans le fichier js/app.js (Annexe 8)

3. Javascript : Le dossier Js contient tous les fichiers javascript utilisés dans l'application.

   Son sous-dossier « controlers » contient les fichiers javascrips qui assurent les interactions possibles entre l'utilisateur et les vues (les interfaces). Chaque vue est associée à un contrôleur javascript. Par exemple, comme le nom l'indique, le contrôleur ficheCtrl.js sera lié à l'interface ficheView.php. Ces liaisons sont faites dans chaque vue grâce à la directive «ng-controller » d'angularJS.

   Le fichier js/app.js contient la configuration principale de l'application. (Annexe 8)

Sécurisation du système

Les tablettes communiquent avec le serveur par Wifi, et l'interface d'administration est accessible depuis n'importe quel ordinateur connecté à internet.

Ainsi la protection du système des intrusions extérieures est une problématique importante qui a été étudié tout au long du développement. Nous avons étudié les différentes options et les systèmes existant au sein de l'école pour assurer une protection par compte d'utilisateur (identifiant et mot de passe).

- L'outil d'administration

Pour le cas de l'interface d'administration, le système utilise le service d'annuaire de Polytech via le protocole LDAP (*Lightweight Directory Access Protocol)* qui permet d'interroger le service et de vérifier les informations de connexion.

Ainsi le système est paramétré pour autoriser l'accès à une liste de comptes de l'école, seule les personnes de cette liste peuvent voir et utilisé l'outil.

- Les tablettes

Dans le cas de la connexion avec les tablettes. La sécurisation est faite par l'utilisation du ficher « .htaccess » utilisé par le serveur. Des comptes y sont créés pour chaque tablette. Le principe est de paramétrer ces comptes coté serveur puis coté tablette pour restreindre l'accès aux pages PHP.

Ces comptes étant paramétrés par l'administrateur du système, les identifiants de connexions sont donc inconnus des utilisateurs et les comptes peuvent être révoqués à tout moment en cas de problème. Ainsi l'outil est assuré d'être protégé des éléments externes au système.

Bilan de l'exercice pédagogique

Ce projet de fin d'étude a été un grand défis et sa réalisation a été suivi par de nombreuses personne et était très attendu. Cette expérience nous a énormément apporté, autant sur le plan humain que technique.

D'un point de vue humain, nous avons eu la chance de vivre une expérience type de suivi de projet à toutes les étapes. Nous avons organisé un mois de conception de l'outil en s'appuyant sur des réunions et des rencontres avec les clients et les futurs utilisateurs de l'outil. Dès le début nous avons donné une grande importance à cette étape qui nous a permis d'anticiper les besoins et de préparer la gestion du développement.

Puis nous avons pu travailler avec des outils de gestion de projets pour organiser notre développement en suivant une des méthodes Agiles : la méthode SCRUM. Cette méthode qui s'appuie sur une organisation par jalons (ou par « Sprint ») nous a permis de mieux présenter notre avancement tout au long de projet et de respecter les délais.

L'objectif était de fournir un outil partiellement terminé mais utilisable pour le mois de Janvier, soit à la moitié du projet. Cette expérience particulière nous a permis de voir et de connaître la période d'utilisation de notre système. D'un point de vue professionnel nous pouvons donc prétendre avoir connu une expérience de mise en production et de livraison au client. Expérience qui est exceptionnel pour un projet de fin d'étude.

Toutes ces expériences sont un atout et une grande mise en valeur de notre formation.

D'un point de vue technique, nous avons travaillé sur un projet autonome qui comprend une partie embarqué et une partie serveur. Cette dualité nous a permis d'étudier le

fonctionnement de la plupart des systèmes dans leur globalité. Les compétences techniques développées sont nombreuses. La technologie Java-Android, le langage PHP orienté objet et la programmation particulière de la librairie Javascript AnglularJS sont d'autant de points qui ont pu être approfondi et que nous pouvons mettre en avant pour nos futures recherches de stages et d'emplois.

Le projet, dans son ensemble, est donc une expérience très enrichissante et permet de bien mettre en avant notre formation au sein de l'école d'ingénieur de Polytech Lille.

Conclusion

Ce rapport présente notre travail effectué dans le cadre du projet de fin d'étude dans le département système communicant de l'école d'ingénieur Polytech'Lille. Nous avons présenté notre travail réalisé à la foi comme bilan pédagogique et un rapport technique à destination des futurs responsable du système.

Lors de ce projet de six mois, nous avons pu mettre en pratique nos connaissances théoriques acquises durant la formation, tout en acquérant de nouvelles.

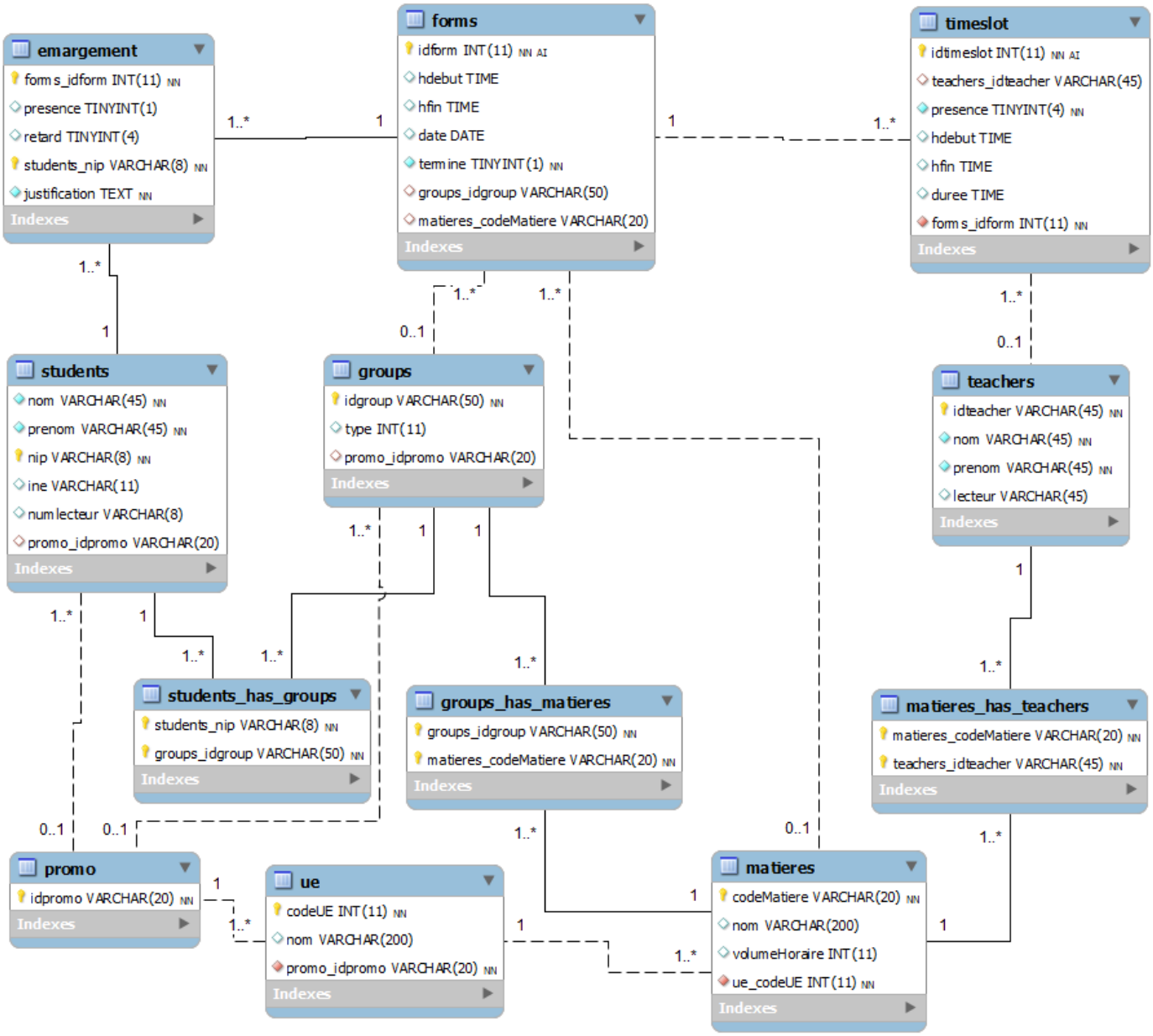
Notre mission était de réaliser un système d'émargement électronique à destination du pôle alternance de notre école. Les objectifs ont été atteint et le système a été adopté et utilisé pendant le projet. Aujourd'hui l'outil est fonctionnel et a été présenter pour une utilisation possible dans d'autres établissement de la région.

Cette expérience nous a permis de vivre toutes les étapes d'un projet professionnel et restera une grande force de notre formation.

Ce projet de fin d'étude est une bonne préparation à notre insertion professionnelle car il présente les caractéristiques d'un projet en entreprise avec la réponse à un appel d'offre, une phase de conception, une phase de développement et également la phase de livraison et de mise en production.

Annexes

Annexe 1 : Schéma UML de la base de données



Annexe 2 : Fonctionnement d'un « Controller »

D'un point de vue fonctionnel, tous les contrôleurs du projet héritent tous d'un même objet java : **projectController.**

Les paramètres à envoyer au serveur sont passé au moment de l'initialisation du contrôleur. Voici les étapes réalisés lors de son exécution :

1. L'objet formate les données à envoyer en JSON. Chaque contrôleur implémente sa façon de formater en fonction de l'existence ou non de données et leur type.
2. L'objet construit l'URL de la requête en fonction de la page PHP ciblée.
3. Le contrôleur exécute la requête et stocke l'objet JSON renvoyé par le serveur
4. Les données sont reconstruites à partir du JSON récupéré.
5. Le code de retour de la requête http est stocké pour vérifier le bon fonctionnement.

Tous les contrôleurs respectent ces étapes. Le nom des méthodes correspondant à ces étapes sont les suivantes.

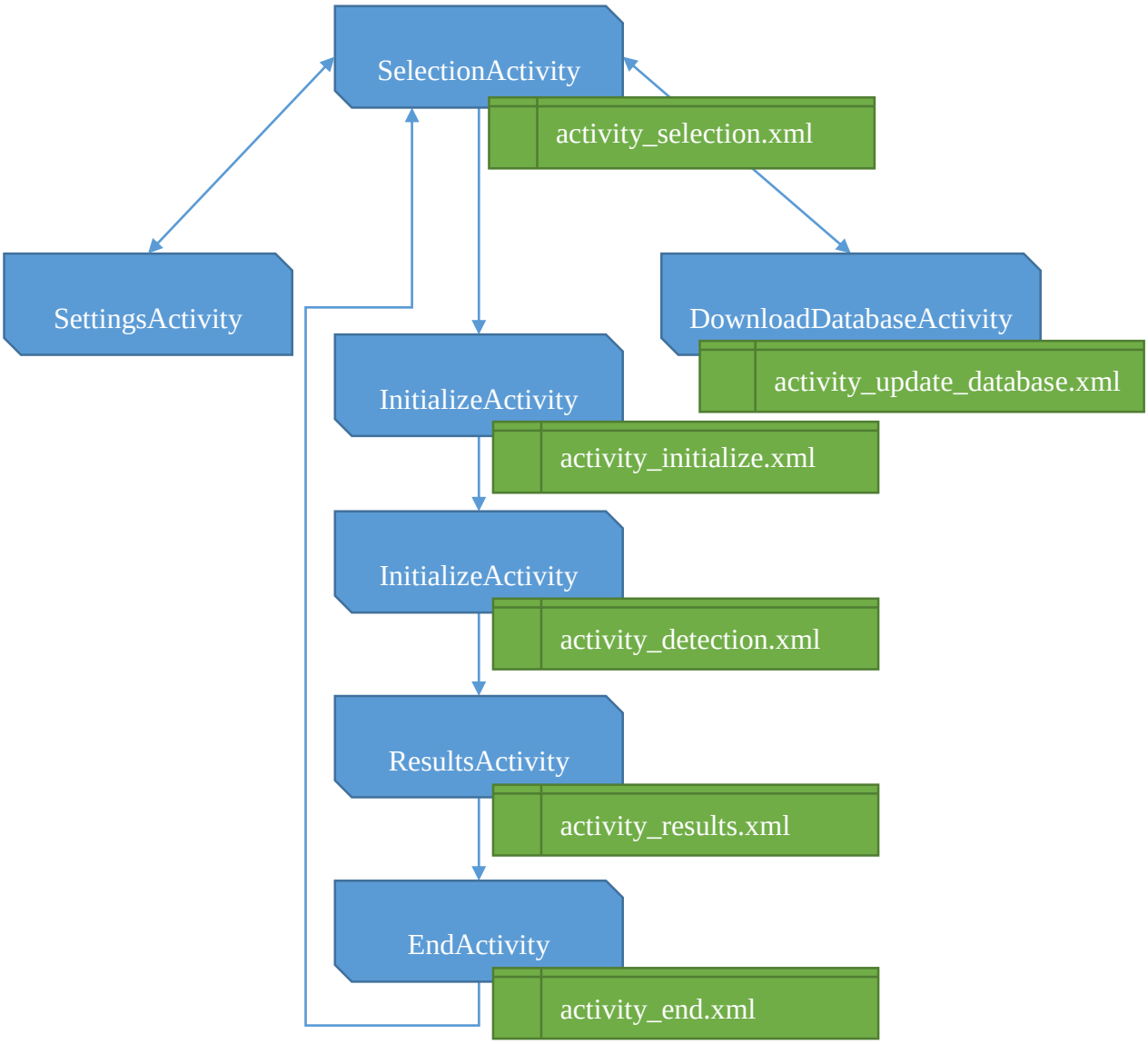
Etape	Nom	Type de méthode et type de retour
1	buildParamsJson	abstract String
2	buildUrl	abstract String
3	sendRequest	void
4	parseJson	abstract void
5	checkresult	abstract void

Annexe 3 : Fonctionnement de la base de donnée Android

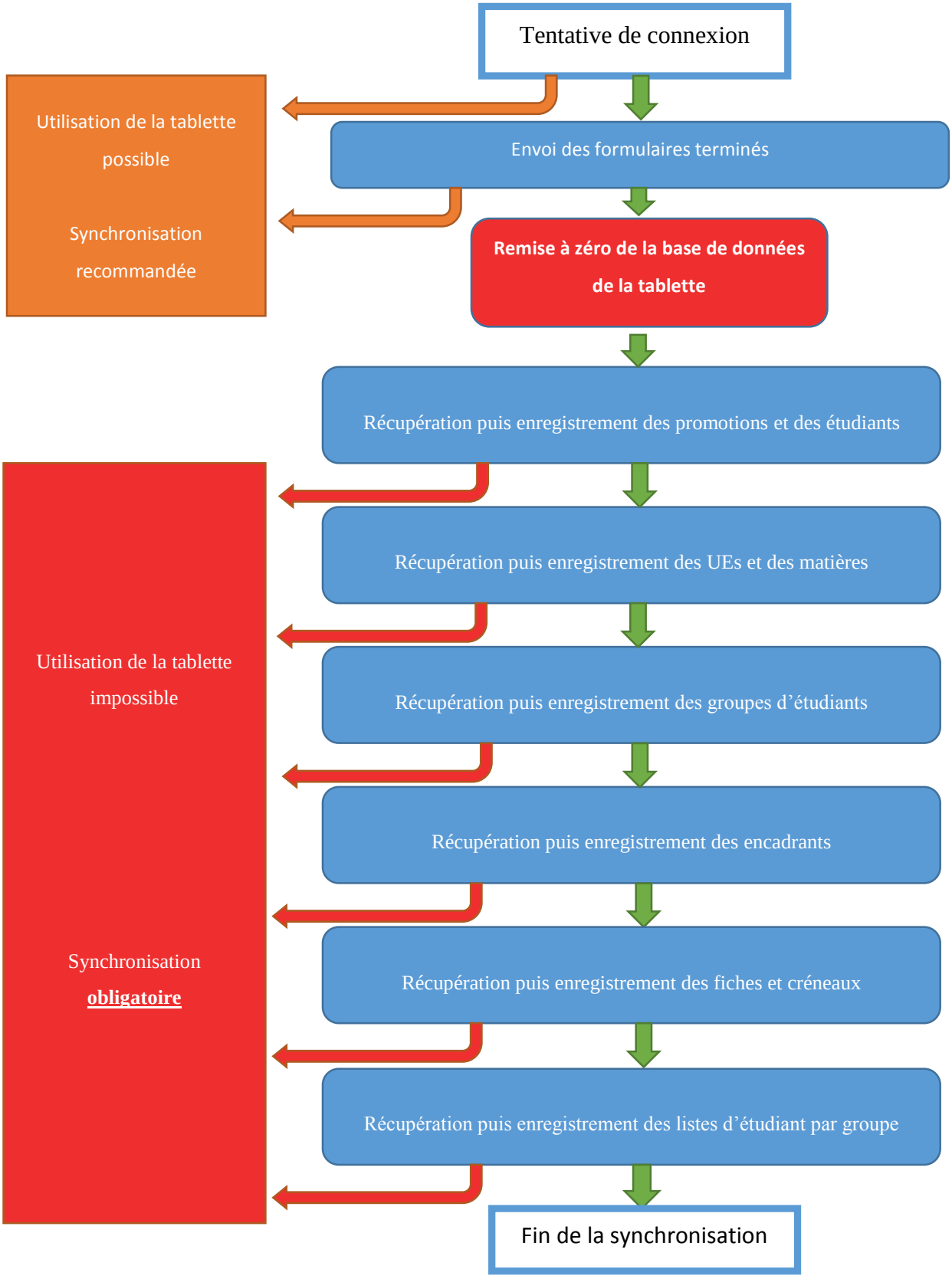
La base de données est utilisée via trois objets Java distincts :

- **DAOBase** sert à gérer la création, suppression et mise à jour de la base en cas d'installation, mise à jour ou désinstallation de l'application.
- **DBHand** regroupe s'ensemble des méthodes de déclarations ainsi que les noms des tables de la base de données et le nom de leurs colonnes. Elle est appelée lors de la création de la base de données et le nom des tables et de leurs colonnes sont centralisé dans cet objet.
- **MaquetteReceivedDAO** hérite de DAOBase. C'est dans cet objet que ce trouve toutes les méthodes d'interaction avec la base de données. Toutes les requêtes SQL utilisés sont décrites et utilisé dans ce même objet.

Annexe 4: Enchainement des activités Android avec les XML associés



Annexe 5 : Protocole de synchronisation



Annexe 6 : Description de l'API PHP

Nom de la page	Rôle	Paramètres	Résultats
check.html	Tester la connexion de la tablette	Aucun	Aucun
updated.php	Avertir de la fin de synchronisation d'une tablette. Sert à stocker la date de dernière mise à jour des tablettes.	tablette : chaine de caractère correspondant au nom de la tablette	Aucun
elevesPromos.php	Permet de récupérer la liste des promotions et des élèves.	Aucun	**promos** : liste de promotion **students** : liste des étudiants
ueMatiere.php	Permet de récupérer la liste des UEs et des matières.	Aucun	**ues** : liste des UEs **matieres** : liste des matières
profsMatieres.php	Permet de récupérer la liste des encadrants et de la table de lien encadrant/matière.	Aucun	**teachers** : liste des encadrants **matieres_has_teachers** : lien ecadrant/matiere
groupsMatieres.php	Permet de récupérer la liste des groupes et de la table de lien groupe/matiere.	Aucun	**groups** : liste des groupes **groups_has_matieres** : liens groupe/matiere
fichesCreneauxMatieres.php	Permet de récupérer la liste des fiches et la liste des créneaux.	Aucun	**forms** : liste des formulaires **timeslots** : liste des créneaux
groupsFormsStudent.php	Permet de récupérer les liens entre groupes et étudiants.	Aucun	**students_has_groups** : liens étudiants/groupes
insertCtrl.php	Permet d'envoyer des fiches terminées au serveur.	**Liste d'objets de structure :** **form** : un formulaire **timeslots** : liste de lien entre les créneaux et le formulaire **emmargements** : table de lien entre les élèves et ce formulaire	Aucun

Annexe 7 : Code du fichier index.php

```
9     <script src="lib/angular/angular.js"></script>
10    <script src="lib/angular/angular.min.js"></script>
11    <script src="lib/angular/angular-route.js"></script>
12    <script src="lib/angular/angular-resource.js"></script>
13    <script src="http://code.angularjs.org/1.2.4/angular.min.js"></script>
14    <script src="http://code.angularjs.org/1.2.4/angular-animate.min.js"></script>
15    <script src="http://code.jquery.com/jquery-2.0.3.min.js"></script>
16    <script src="js/services/HTTP.js"></script>
17    <script src="js/controllers.js"></script>
18    <script src="js/controleurs/promoCtrl.js"></script>
19    <script src="js/controleurs/studentCtrl.js"></script>
20    <script src="js/controleurs/matiereCtrl.js"></script>
21    <script src="js/controleurs/ficheCtrl.js"></script>
22    <script src="js/controleurs/ueCtrl.js"></script>
23    <script src="js/controleurs/groupCtrl.js"></script>
24    <script src="js/controleurs/teacherCtrl.js"></script>
25    <script src="js/controleurs/ficheDetailCtrl.js"></script>
26    <script src="js/app.js"></script>
27 </head>
28 <body ng-app="emargemeApp">1
29    <div ng-controller="MenuDataCtrl">
30       <div id="bottommenu">
31          <ul>
32             <li ng-repeat="elt in menuElts">3
33                <a href="#{{elt.page}}" ng-click="changeCurrentMenu(elt)">{{elt.name}}</a>
34             </li>
35          </ul>
36          <h2>Page: {{page}}</h2>
37       </div>
38       <div>
39          <div ng-bind-html="currentPage"></div>
40
41          <div ng-view></div>
42       </div>
43    </div>
```

2

Ce fichier fait appel à tous les fichiers utiles pour l'application.

Au point 1 : Grâce à la directive « **ng-app** », on lie le « body » de l'index à l'application « emargemeApp » que nous avons déclarée dans le fichier js/app.js Cette liaison permet à AngularJS de créer le contexte global de l'application qui sera accessible dans tous les contrôleurs à partir du service $rootScope.

Au point 2 : Avec la directive « **ng-controller** », on associe le contrôleur « MenuDataCtrl », dont le code est contenu dans le fichier js/controlleurs.js, à la balise « div » ; ainsi on peut avoir un sous-contexte de l'application sur une partie de l'index. Dans le contrôleur, seule les objets ou les fonctions préfixés par $scope. seront accessible dans le DOM (code HTML). Ce contrôleur possède deux fonctionnalités :

- Il initialise le contexte $scope en chargeant dans l'objet « menuEtls » la liste d'élément constituant le menu en passant par le service $http utilisé pour faire des requêtes Ajax. Cette liste est représentée par un objet JSON situé dans le fichier data/menu.json.
  Voici la structure d'un objet menu :

```
{
    "name": "Etudiants",
    "page": "studentView/default",
}
```

  Name : le nom de la rubrique du menu

  Page : l'identifiant de la page. Cet identifiant est configuré dans le fichier app.js, il est utilisé pour charger les vues qui leurs sont associées.

    - Il implémente la fonction *changeCurrentMenu*() qui permet de modifier le menu en cours dans le contexte de l'application.

Au point 3 : On affiche les données définies dans le scope du contrôleur.

- La directive « **ng-repeat** » permet de parcourir tous les éléments du tableau menuElts définis dans le contrôleur.
- L'expression *{{elt.name}}* associe un lien HTML à l'identifiant de la page. En cliquant sur ce lien, depuis un navigateur, l'identifiant du menu s'affiche dans la barre d'adresse et il est ensuite prise en compte par le fichier app.js qui va rediriger l'application sur la bonne vue.

Annexe 8 : Fichier js/app.js

Ce fichier contient la configuration de l'application.

1. **Initialisation des modules utiles pour l'application.**

   Le fichier *js/app.js* est le premier fichier exécuté par AngularJS pour configurer l'application. Dans l'index, il doit toujours être en dernière position de fichiers à charger pour permettre à angularJS d'initialiser rapidement les autres modules au moment du chargement.

```
1  'use strict';
2
3  /* App Module */
4
5  var emargemeApp = angular.module('emargemeApp', [
6      'ngRoute',
7      'httpServices',
8      'indexControllers',
9      'promoControllers',
10     'studentControllers',
11     'ueControllers',
12     'matiereControllers',
13     'groupControllers',
14     'teacherControllers',
15     'ficheControllers',
16     'ficheDetailControllers'
17 ]);
```

Ce code permet de créer le module « emargemeApp » qui constitue l'objet central de l'application. Il est obligatoire de lister tous les autres modules Angular (contrôleurs, services, filtre, …) qui seront utilisés dans l'application :

- **ngRoute** : Ce module contient le service $*routeProvider* qui est utilisé pour gérer l'URL. Il veille sur l'objet $location.url et tente de diriger une adresse, un lien, sur la bonne interface.
- **httpServices** : Ce module, dont le code se trouve dans le fichier *js/services/http.js*, déclare un nouveau service http. Ce dernier permet de simplifier l'écriture d'une requête AJAX dans les contrôleurs.
- **xxControllers** : Ces modules déclarent les contrôleurs qui seront utilisés dans l'application. Les fichiers javascript associés à ces contrôleurs sont dans le dossier *js/controleurs*.

2. Configuration de la redirection

```
emargemeApp.config(['$routeProvider',function($routeProvider) {
    $routeProvider.
    when('/promoView/:idPage', {
        templateUrl: 'view/promoView.php',
        controller: 'MenuDataCtrl'
    }).
    when('/studentView/:idPage', {
        templateUrl: 'view/studentView.php',
        controller: 'MenuDataCtrl'
    }).
    when('/teacherView/:idPage', {
        templateUrl: 'view/teacherView.php',
        controller: 'MenuDataCtrl'
    }).
    when('/ueView/:idPage', {
        templateUrl: 'view/ueView.php',
        controller: 'MenuDataCtrl'
    }).
    when('/matiereView/:idPage', {
        templateUrl: 'view/matiereView.php',
        controller: 'MenuDataCtrl'
    }).
```

Ce code teste la valeur de l'URL. L'expression WHEN permet à l'application d'intégrer, dans la directive ng-view de l'index, un template dont l'identifiant est spécifié dans l'URL. Dans notre cas, les identifiants sont de la forme :

/nomDelaPage/idPage

Le paramètre ***nomDelaPage*** nous permettra d'aller dans le dossier « view » et de charger la bonne vue.

Le paramètre ***idPage*** permet de passer un deuxième paramètre à l'url pour assurer des traitements plus complexe. Par défaut, ***idPage*** vaut « default »

Par exemple : ***/ficheView/139*** demande à l'application d'afficher la fiche dont l'identifiant est 139 dans la base de donnée. La valeur 139 sera récupérée dans les contrôleurs de fiche grâce au service $routeParams en précisant juste le nom du paramètre déclaré : ***$routeParams.idPage***

3. *Déclaration de Filtre*

```
58 emargemeApp.filter('checkmark', function() {
59     return function(input) {
60         if(input == 1)
61             return '\u2713';
62         else
63             return '\u2718';
64     };
65 });
```

Ce code permet de créer un filtre. Il a pour nom « checkmark » et en fonction du paramètre input, il renvoie soit le caractère relatif à faux si input = 0 et vrai si input = 1.

Par exemple, ce filtre est utilisé dans le fichier *view/detail/ficheDetailView.js* pour afficher la présence avec l'expression:

{{emargement.student.presence | checkmark }}

Emargement.student.presence est le booléen passé en paramètre de la fonction checkmark qui traduit un booléen à un caractère ASCII.

Annexe 9 : Contrôleur Angular

```
1  'use strict';
2
3  var ficheControllers = angular.module('ficheControllers', ['httpServices']);
4
5  ficheControllers.controller('ficheCtrl', ['$scope', 'HTTP',
6          function ficheCtrl($scope, HTTP){
7
8              $scope.send = function(date, hdebut, hfin){
9                  var date = $scope.year+"-"+$scope.month+"-"+$scope.day;
10                 var hdebut = $scope.dh+":"+$scope.dm;
11                 var hfin = $scope.fh+":"+$scope.fm;
12                 var idgroup = $scope.groupValue;
13
14                 var ueData = {hdebut:hdebut, hfin:hfin, date:date, groups_idgroup:idgroup};
15
16                 HTTP.save({object:"form", data: ueData}, function(msg, headers){
17                     if(msg.result == 1){
18                         $scope.errorMessage = "OK";
19                         $scope.year = "";
20                         $scope.month = "";
21                         $scope.day = "";
22                         $scope.dh = "";
23                         $scope.dm = "";
24                         $scope.fh = "";
25                         $scope.fm = "";
26                         $scope.promoValue = "";
27                         $scope.groupValue = "";
28                     }
29                     else if(msg.result == 2){
30                         $scope.errorMessage = "Existe déjà";
31                     }
32                     else
33                         $scope.errorMessage = "rien";
34                 });
35             };
```

Ce code permet de créer un nouveau module. Un module peut être un service, un filtre, un contrôleur, …

Le module ci-haut se nomme '*ficheControllers*' et ce nom doit être signalé dans le fichier de configuration *js/app.js* parmi les modules utilisés par l'application.

Grâce à la méthode *controller* invoqué sur l'objet *ficheControllers*, on déclare le contrôleur *ficheCtrl* en précisant les services qui seront utilisés dans le contexte de ce contrôleur, en occurrence, les services $*scope* (toujours obligatoire) et le service *HTTP* que nous avions créé dans *js/services/http.js*

Le corps de la fonction *ficheCtrl*() définit le contexte du contrôleur ainsi que les variables et fonctions susceptible d'être utilisé par la vue lié à ce contrôleur.

Annexe 10 : une VUE Angular

```
1  <div ng-controller="ficheCtrl" id="body">
2  <div id="ficheView" class="float_left">
3      <div>
4          <span class="error" id="haut" >{{errorMessage}}</span>
5          <form>
6              <fieldset>
7                  <legend>Ajouter une nouvelle fiche d'emargement:</legend>
8                  <table id="formTable">
9                      <tr>
10                         <td><strong>date: </strong></td>
11                         <td>
12                             <table class="includedTable">
13                                 <tr>
14                                     <td><label for="day"><strong>Jour: </strong></label></td>
15                                     <td><label for="month"><strong>Mois: </strong></label></td>
16                                     <td><label for="year"><strong>année: </strong></label></td>
17                                 </tr>
18                                 <td>
19                                     <select  name="day" id="day" ng-model="day">
20                                         <option value=""></option>
21                                         <option ng-repeat="num in ['01','02','03','04','05','06'
22                                     </select>
23                                 </td>
24                                 <td>
25                                     <select  name="month" id="month" ng-model="month">
26                                         <option value=""></option>
27                                         <option ng-repeat="num in ['01','02','03','04','05','06'
28                                     </select>
29
30                                 </td>
31                                 <td>
32                                     <select  name="year" id="year" ng-model="year">
33                                         <option value=""></option>
```

Une vue angular c'est quasiment du code HTML. Angular rend plus intelligent le code HTML en lui ajoutant des directives qui permettent de faire la liaison entre le fichier javascript et le DOM.

A la ligne 1 de ce code, grâce à la directive ***ng-controller*** on lie le contrôleur *ficheCtrl* (ANNEXE 9) à la balise DIV dont l'identifiant est « body ». Ainsi, toutes les balises contenues dans celle-ci appartiennent au contexte du contrôleur *ficheCtrl*. En dehors de « body » aucune fonctionnalité du contrôleur *ficheCtrl* ne sera accessible.

A la ligne 4, nous remarquerons l'expression ***{{errorMessage}}.***

Cette expression permet d'afficher dans la vue la valeur de la variable « errorMessage » déclarée dans le contrôleur *ficheCtrl* comme ceci : $scope.errorMessage= 'OK', par exemple.

A la ligne 19, nous pouvons lire la directive ***ng-model***.

Cette directive permet de lier la valeur d'un formulaire (Input, Select, …) à une variable du contrôleur.

Dans le cas où ng-model=day, il existe une variable $scope.day dont la valeur sera modifiée à chaque fois que l'utilisateur changera la valeur de 'day'. Ceci est fait automatiquement par Angular.