



ÉCOLE POLYTECHNIQUE DE L'UNIVERSITÉ DE TOURS

64, Avenue Jean Portalis

37200 TOURS, FRANCE

Tél. (33)2-47-36-14-14

Fax (33)2-47-36-14-22

[www.polytech.univ-tours.fr](http://www.polytech.univ-tours.fr)

## Parcours des écoles d'ingénieurs Polytech

### Rapport de projet S2

# Application de gestion des présences des étudiants grâce à leur carte étudiant

Auteur(s)

**Thomas Couchoud**

[[thomas.couchoud@etu.univ-tours.fr](mailto:thomas.couchoud@etu.univ-tours.fr)]

**Victor Coleau**

[[victor.coleau@etu.univ-tours.fr](mailto:victor.coleau@etu.univ-tours.fr)]

Encadrant(s)

**Pascal Makris**

[[makris@univ-tours.fr](mailto:makris@univ-tours.fr)]

**Polytech Tours  
Département DI**

# Table des matières

---

<b>Introduction</b>	<b>1</b>
<b>1 Découverte des cartes à puce</b>	<b>2</b>
1.1 Historique . . . . .	2
1.2 Utilisations des cartes à puces . . . . .	3
1.3 Composition des cartes . . . . .	3
1.4 Fonctionnement . . . . .	4
1.5 Système d'exploitation . . . . .	5
1.6 Attaques à l'encontre des cartes . . . . .	5
1.6.1 Les attaques invasives . . . . .	6
1.6.2 Les attaques non-invasives . . . . .	6
1.6.3 Les attaques semi-invasives . . . . .	6
1.6.4 Les attaques logicielles . . . . .	6
<b>2 Application : Gestion de présence des étudiants</b>	<b>7</b>
2.1 Étude logicielle . . . . .	7
2.2 Création de l'application . . . . .	9
2.2.1 Communication avec la carte . . . . .	9
2.2.2 Notion de groupe . . . . .	11
2.2.3 Interface . . . . .	11
2.2.4 Base de donnée SQL . . . . .	15
2.2.5 Résultats . . . . .	15
2.2.6 Problèmes rencontrés et solutions . . . . .	16
<b>3 Mise en place du travail réalisé</b>	<b>18</b>
<b>4 Conclusion</b>	<b>20</b>
 <b>Annexes</b>	 <b>20</b>
<b>A Liens utiles</b>	<b>21</b>
<b>B Installation du programme</b>	<b>22</b>
<b>C Structure du module TerminalReader</b>	<b>23</b>
<b>D Structure du programme</b>	<b>24</b>
<b>E Fiche de suivi de projet PeiP</b>	<b>26</b>

# Introduction

---

L'objectif de ce projet est la création d'un logiciel permettant la gestion de présence de certains élèves durant des périodes précises de la journée. Ce dispositif est basé sur l'utilisation des cartes étudiantes, permettant ainsi de simplifier la gestion des émargements aussi bien pour la scolarité que pour les élèves.

En effet, les élèves gagnerons du temps pour leurs projets ou études car ils n'auront pas à signer la fiche de présence au bureau de la secrétaire et simplifiera le travail du personnel en numérisant automatiquement les données ainsi acquises. De plus, on peut y voir un aspect écologique dû à l'économie de papier. Cette consommation passera d'environ un demi millier de feuilles par an à une quantité quasiment nulle.

Dans un premier temps, ce projet demande une certaine connaissance de la technologie des cartes à puces. Il sera donc nécessaire de rechercher, lire et comprendre les documentations des différentes normes présentes pour un tel système.

# Chapitre 1

## Découverte des cartes à puce

---

Depuis quelques décennies, il nous est familier d'utiliser des cartes à puce pour de nombreuses applications. En effet, celles-ci permettent de conserver des informations personnelles numériquement dans un support facilement transportable. Les cartes bancaires, vitales, étudiantes, de transport en commun sont quelques exemples parmi d'autres de nos utilisations quotidiennes de ces cartes.

Nous allons dans un premier temps nous intéresser à leur historique pour nous diriger progressivement vers des aspects plus techniques de celles-ci.

### 1.1 Historique

L'idée de rassembler des informations sur une unique carte remonte aux années 1950. C'est en Amérique que la compagnie *Diners' Club International* introduisit les tous premiers exemplaires pour le paiement. Celles-ci n'étaient faites que de papier et n'étaient seulement acceptées par 27 restaurants de New York. Sur l'une des faces était inscrit les différentes adresses de ces restaurants tandis que l'autre face comportait des informations sur le propriétaire comme son nom, adresse et numéro de compte.



FIGURE 1.1 – Carte de Diners' Club International.

Cependant ce type de carte n'est pas à proprement parler celui que nous connaissons de nos jours. En effet, il a fallu attendre les années 1960 pour voir apparaître de véritables cartes à puce développées par *American Express*. Il faudra attendre 1967 pour que cette nouvelle technologie traverse l'Atlantique.

C'est ainsi que depuis 1974, on observe des améliorations successives de ce support de communication. Cela passe par l'ajout d'une mémoire programmable (Roland Moreno - 1974), d'un microprocesseur (Michel Ugon - 1977) et l'apparition de différents systèmes d'exploitations pour la carte.

## 1.2 Utilisations des cartes à puces

Aujourd'hui, nous utilisons de nombreuses cartes à puces dans notre vie de tous les jours, bien que nous ne nous en rendons pas toujours compte. Elles sont notamment employées pour l'identification d'un personnel (carte d'identité nationale Belge, carte étudiante, etc.), les transactions financières (cartes bancaires, porte-monnaies, etc...), la téléphonie mobile (carte SIM), la santé (carte vitale) ou encore la sécurité informatique.

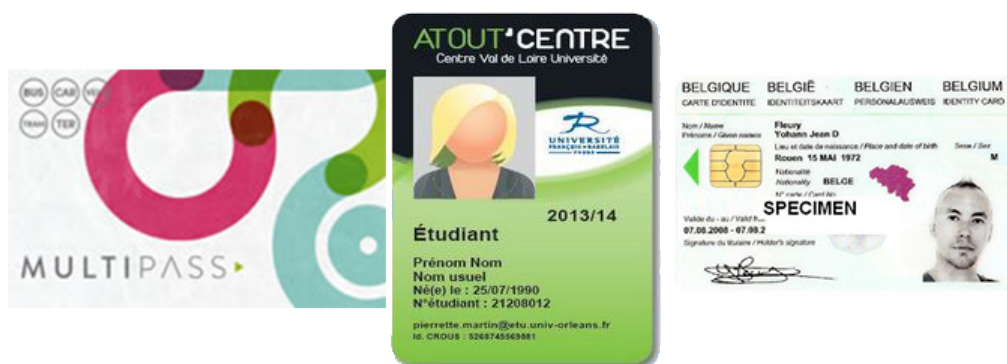


FIGURE 1.2 – Exemples de cartes.

## 1.3 Composition des cartes

Dans un premier temps, il est important de bien différencier l'enveloppe physique de sa composition interne.

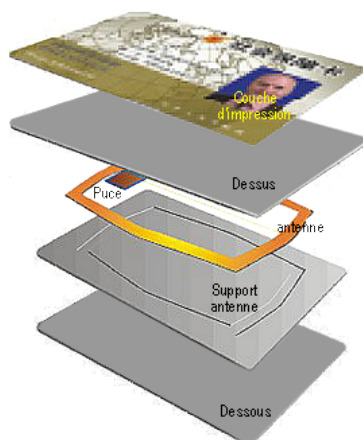


FIGURE 1.3 – Composition d'une carte.

L'enveloppe physique n'est constituée que de deux bouts de carton rectangulaire collés dos-à-dos : c'est ce que l'on tient dans nos mains.

La composition interne est l'ensemble des composants électroniques qui permettent le fonctionnement informatique de la carte. Sans ceux-ci, il ne s'agirait ni plus ni moins que d'un morceau de carton. Cette dernière est toujours plus ou moins la même et comprend :

- Une mémoire vive extrêmement petite (256 octets).
- Une mémoire morte de taille variable mais restant petite en comparant aux autres appareils. Elle se compte en Ko.
- Un microprocesseur 8 bit de 4 MHz.
- Une mémoire vive ne s'effaçant pas lors de la mise hors tension, appelée mémoire Flash.
- Une antenne dans le cas d'une carte sans contact.

Les cartes à puce ont suivi l'évolution générale de la technologie. Ainsi, grâce à la miniaturisation de bien des composants, on a pu voir l'apparition de microprocesseurs assez petits et légers pour pouvoir être utilisés dans une telle application.

## 1.4 Fonctionnement

De nos jours, on peut distinguer quatre catégories de cartes à puce. Celles-ci se classent en fonction de leur mode de communication et de leur moyen de contrôle :

- Communication par contact ou par radiofréquences.
- Contrôle par logique câble ou par microprocesseur.

Les modes de communication permettent de définir l'accessibilité de la puce en elle-même.

- Dans le cas d'une communication par contact, les échanges de données entre la carte et le lecteur se font à l'aide d'un circuit imprimé doré et très fin appelé micromodule (souvent nommé à tort « puce »). Celui-ci est divisé en huit parties, chacune ayant un rôle précis lors de transmission de données. Grâce à ce système, la puce peut alors être cachée dans la carte qui la contient.
- Lors de communication sans contact, l'antenne interne est composée de spires qui sont gravées dans la carte elle-même. Les échanges se font alors à faible ou moyenne distance par radiofréquences.

Le moyen de contrôle (logique câblée ou programmée) définit, entre autres, les possibilités d'actions et d'adaptations de la puce.

- La logique câblée est une méthode permettant de répondre à un besoin de manière logique à l'aide d'un circuit fini. De fait, chaque circuit ne peut répondre qu'aux demandes pour lesquelles il a été conçu. De plus, plus les problèmes sont nombreux et/ou complexes à résoudre, plus la taille du circuit augmente.
- La logique programmée repose sur un microprocesseur qui va traiter les informations qu'il reçoit. La taille des composants ne varie donc que très peu lors de la complexification des demandes. De plus, la reprogrammabilité permet de modifier et/ou ajouter les droits, options et fonctionnalités de la carte sans devoir la détruire et en refaire une autre.

A la vue de ces éléments, il est compréhensible que la logique programmée, avec microprocesseur, ait aujourd'hui largement prit le pas sur la logique câblée.

## 1.5 Système d'exploitation

Comme la plupart des matériels informatique, les cartes à puces nécessitent un système d'exploitation. Celui-ci dirigera les applications. Pour ce faire, il reçoit des demandes des logiciels exécutés qu'il accepte ou refuse. Cela passe par l'allocation de mémoire et mémoire vive, la communication avec des périphériques extérieurs ou un réseau, l'utilisation du processeur et la création de fichiers.

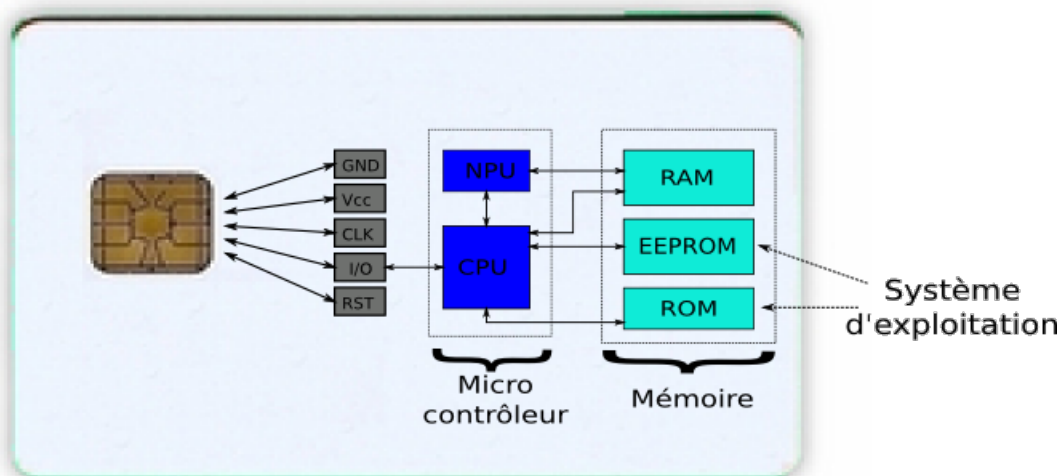


FIGURE 1.4 – Système d'exploitation.

Le système d'exploitation d'une carte à puce devra répondre aux mêmes besoins que celui d'un ordinateur mais à une échelle réduite du fait de la faible puissance de calcul du microprocesseur et du peu de mémoire disponible. En outre, ce type de cartes servant le plus souvent à une identification du propriétaire, des algorithmes de cryptage devront être présents en vue d'assurer la protection des données sensibles.

Afin d'assurer le bon fonctionnement de la carte et d'éviter les erreurs grossières telles que la suppression ou la modification involontaire du système d'exploitation, celui-ci se trouve sur une partie non ré-inscriptible de la mémoire.

De plus, pour que les cartes puissent être utilisées le plus largement possible, il faut que le système d'exploitation soit conforme à un standard de communication ainsi qu'à une structure de transmission des données (norme ISO 7816-4).

## 1.6 Attaques à l'encontre des cartes

Comme tout matériel pouvant contenir des données sensibles, les cartes à puces que nous étudions sont susceptibles de subir des attaques ayant pour objectif de dérober des informations. Pour cela, les personnes mal intentionnées peuvent avoir recours à quatre types d'attaques : invasives, non-invasives, semi-invasives et logicielle.

### 1.6.1 Les attaques invasives

Ce type d'agressions consiste à faire fondre certains composants de la carte pour avoir un accès direct au circuit électronique et ainsi installer différentes sondes lisant illégalement le contenu de la carte. Le principal inconvénient des attaques invasives est la destruction du support physique de la puce donnant à autrui le doute d'une utilisation frauduleuse de la carte.

### 1.6.2 Les attaques non-invasives

Ce deuxième type d'agressions s'appuie aussi sur l'aspect matériel de la carte mais sans l'endommager. En effet, grâce à l'exploitation de canaux auxiliaires (observation extérieure du système), il est possible de mesurer l'énergie consommée ou le temps passé pour réaliser une action, permettant ainsi d'en déduire les données traitées normalement secrètes.

### 1.6.3 Les attaques semi-invasives

Ce troisième type d'agressions est un peu différent des deux premiers puisqu'il cherche volontairement à provoquer une erreur dans le système cible. Pour cela, un grand nombre de moyens est disponible, comme injecter de la lumière ultraviolet ou perturber l'alimentation énergétique de la carte. Les bugs ainsi provoqués permettent l'exécution d'actions normalement interdites en vue de récupérer des informations habituellement inaccessibles. Ces attaques peuvent endommager la carte si elles ne sont pas contrôlées et limitées.

### 1.6.4 Les attaques logicielles

Dans ce dernier type d'attaque, il n'est plus question du hardware de la carte mais bien du software. Il est envisageable avec les cartes multi-applicatives (qui ne sont pas spécifiques à une seule et même action) de créer un logiciel espion que l'on installera sur la carte afin de garder une trace illégale de toutes les activités de cette dernière et des informations qu'elle traitait.



## Chapitre 2

# Application : Gestion de présence des étudiants

---

Afin de répondre à la problématique, nous avons été amenés à réaliser un programme de gestion de présence des étudiants grâce à leur carte Atout-centre. Celui-ci ayant pour but de remplacer les feuilles d'émargements, il devra être capable de définir pour chaque plage horaire les différentes personnes devant justifier de leur présence. Enfin, il est nécessaire de traiter les différentes informations acquises pour les rendre utilisables par la scolarité.

Le programme a été réalisé dans le langage JAVA et nécessite une base de donnée SQL pour la liste des personnes associées à leur numéro de carte.

Pour ce faire, nous allons séparer le problème en plusieurs parties. La première portera sur la communication entre le programme et la carte. La seconde définira la notion de groupe. La suivante décrira l'interface visible par les utilisateurs. Puis nous parlerons de la base de donnée SQL. Nous terminerons par l'exploitation des données.

Cependant, commençons par une analyse du travail qui devra être réalisé.

## 2.1 Étude logicielle

Le programme se décomposant principalement en deux grosses parties, nous allons tenter de schématiser simplement ce que nous voulons réaliser. Les parties de traitement au niveau de l'interface seront mises de côté puisqu'il s'agit seulement de l'aspect visuel du programme, celui-ci pourra très bien fonctionner sans celle-ci.

La première partie qui sera analysée est celle permettant de récupérer la présence d'une carte ainsi que ses informations. Plusieurs points sont à noter :

- L'initialisation du lecteur se fait à chaque boucle. Cela permet de détecter l'ajout ou la suppression d'un lecteur. Ainsi on peut démarrer le programme sans avoir connecté le lecteur et le brancher plus tard.
- L'accès à la "Fin" se fait uniquement lorsqu'une carte est retirée. En soi cela pose un problème car si l'on ferme l'application celle-ci peut attendre indéfiniment que l'on retire une carte avant de se finir.

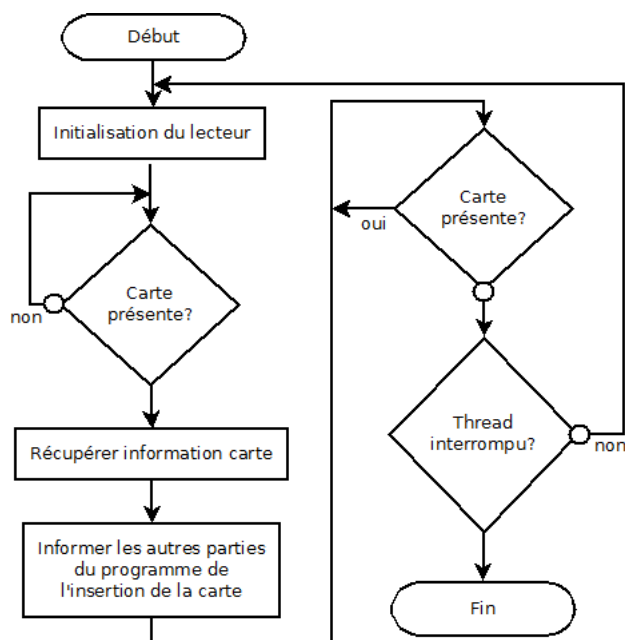


FIGURE 2.1 – Algorithme simplifié de la gestion du lecteur.

La deuxième partie essentielle du programme est la gestion des différentes périodes. En schématisant cela de manière très simple, on obtient l'algorithme ci-dessous :

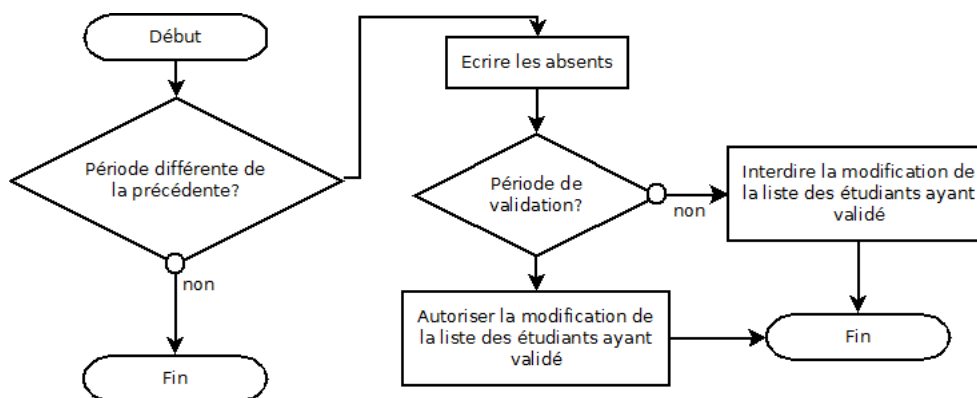


FIGURE 2.2 – Algorithme simplifié de la gestion d'émargement.

Ce traitement est réalisé pour chaque groupe de façon constante. A noter que dans le programme final, la partie "autorisation" ou "interdiction" de modifier la liste des étudiants, ne se trouve pas dans cette boucle. Cette partie a été déplacé dans la fonction permettant de valider un étudiant. Cependant afin que l'algorithme garde un sens, nous avons mis cette vérification après l'écriture des absents (les deux procédés sont totalement équivalents).

## 2.2 Création de l'application

### 2.2.1 Communication avec la carte

La fonction principale de ce "module" du programme est d'établir une communication avec un lecteur de carte et ainsi pouvoir détecter la présence d'une carte devant celui-ci. Cette partie a été réalisée de façon totalement indépendante vis-à-vis de l'application de gestion. On peut la voir comme une sorte de librairie à importer dans un projet. Il ne reste par la suite plus qu'à implémenter une interface à l'endroit souhaité pour la gestion de l'ajout/suppression d'une carte.

#### Communication avec le lecteur

La communication avec le lecteur est réalisée grâce à la librairie `javax smartcardio`. Celle-ci nous permet principalement de récupérer une liste des lecteurs connectés à l'ordinateur et réaliser des connections avec les cartes qui seront présentées devant ceux-ci.

Afin de rendre la gestion du lecteur de carte indépendant de l'application l'utilisant, la partie détaillée qui va suivre est réalisée dans un Thread qui est exécuté lors de la création de notre objet `TerminalReader`.

Celui-ci exécute une boucle de façon permanente. Elle consiste en la répétition de trois portions de code :

1. On obtient la liste des lecteurs de carte connectés à l'ordinateur. Puis selon des critères définis, le nom dans notre cas, on récupère celui souhaité. Si le lecteur n'a pas été trouvé, on revient au début de la boucle. De plus, une comparaison de la présence du lecteur est effectuée entre cette boucle et la précédente. Ainsi, on peut déterminer si ce dernier a été branché ou débranché, et avertir d'autres parties du programme de ce changement.
2. Une fois que nous avons identifié notre lecteur, nous attendons l'insertion d'une carte. Dès que cet événement se produit, nous récupérons le numéro de carte. Une fois cette tâche réalisée, nous informons les autres parties du programme de la présence de cette carte.
3. Pour finir, on attend que la carte soit retirée pour recommencer notre boucle.

#### Récupération des informations de la carte

L'information qui nous intéresse, dans la carte, est son numéro (UID). En effet, elle représente pour nous un moyen d'identifier chaque étudiant puisque ce numéro est unique à chaque carte.

Afin de récupérer l'UID, nous devons envoyer une requête à la carte. Ces requêtes suivent une forme bien précise :

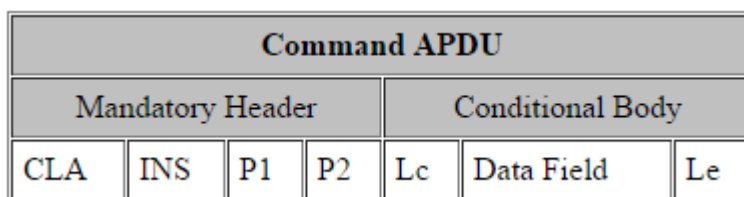


FIGURE 2.3 – Trame d'une requête APDU.

Chaque partie de la trame représente un byte :

- CLA : Le byte de classe d'instruction.
- INS : Le byte d'instruction.
- P1 : Le premier byte de paramètre.
- P2 : Le second byte de paramètre.
- LC : (Optionnel) Indique la longueur du champ de donnée.
- DATA : (Optionnel) Le champ de donnée.
- LE : (Optionnel) Indique le nombre maximal de bytes attendus en réponse.

Si on résume un peu tout ces paramètres, CLA et INS représentent la commande à effectuer, P1 et P2 les paramètres pour la commande et LC, DATA et LE servent pour d'éventuels données à transmettre.

Dans notre cas, nous lui envoyons la commande `FF CA 00 00 00`. A noter que notre commande à un champ DATA de taille 0, nous ne sommes par conséquent pas obligé de renseigner le champ DATA. Cette commande est envoyée par le canal de communication généré grâce à la librairie javax.

Après avoir envoyé la requête et le traitement fait par la carte, celle-ci nous enverra une réponse APDU. Elle se présente sous cette forme :

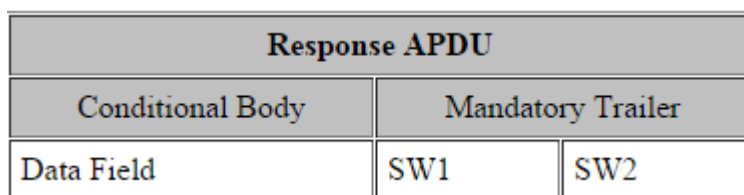


FIGURE 2.4 – Trame d'une réponse APDU.

Chaque partie de la trame représente un octet :

- DATA : (Optionnel) D'éventuels données envoyées par la carte.
- SW1 : La première partie du résultat de la requête.
- SW2 : La seconde partie du résultat de la requête.

SW1 et SW2 nous servent à savoir si le processus s'est bien exécuté, et si ce n'est pas le cas, savoir quelle erreur s'est produite. Dans notre cas, la carte nous renverra une réponse, avec dans le champ DATA, l'UID de la carte.

### 2.2.2 Notion de groupe

Maintenant que nous savons récupérer les informations concernant une carte, il nous faut savoir quels étudiants doivent valider leur présence et quand. Pour ce faire, nous avons mis en place des "Groupes". Pour nous, cet objet est constitué de deux éléments principaux :

- Une liste des différents étudiants composant ce groupe.
- Une liste des différentes périodes durant lesquelles les étudiants doivent valider leur présence.

```
public class Group implements Serializable
{
    private final ArrayList<Student> students;
    private final ArrayList<Period> periods;
```

FIGURE 2.5 – Une partie de cet objet Groupe dans notre code.

En ayant introduit ces groupes, il sera par la suite très simple pour un utilisateur quelconque de paramétrer le logiciel pour qu'il fonctionne comme voulu. Il est en effet bien plus agréable et moins long de configurer un seul groupe que de devoir spécifier les plages horaires pour chaque étudiant.

De plus, il est possible de gérer plusieurs groupes indépendamment les uns des autres. Par exemple, on peut avoir un groupe d'élèves de DII4 et un autre, d'étudiants de cette même classe, suivant un cours optionnel. Même si leur nom est présent dans les deux groupes, le programme interprétera qu'ils doivent être présents aux deux cours.

Finalement lorsqu'un changement doit être effectué (modification de la liste des étudiants, changement des horaires de présence), il sera très aisé de modifier un groupe en retirant juste un élève ou une période sans devoir tout reconfigurer.

Cette notion de groupe est donc un moyen simple et rapide de gérer les différentes classes. Seule la première configuration sera plus longue mais n'est à faire qu'une seule fois.

### 2.2.3 Interface

L'interface est la partie centrale de notre programme. En effet, celle-ci permet l'affichage et la gestion des différentes données. Elle exécute aussi de façon transparente différentes actions nécessaires au bon fonctionnement.

Parlons tout d'abord de ces actions que nous ne voyons pas. En réalité l'interface exécute un Thread à son lancement. Celui-ci, va de manière permanente, actualiser les différents groupes. En premier lieu, il effectue la récupération des étudiants devant valider leur présence. Ce qui nous permettra de constituer notre liste à afficher dans la fenêtre. En seconde lieu il procède à l'écriture des fichiers d'absences lorsqu'une période se termine.

#### Interface principale

Maintenant attardons nous sur l'apparence du programme qui à été réalisé de façons à être instinctif. La fenêtre principale que nous verrons la plus part du temps est celle-ci :

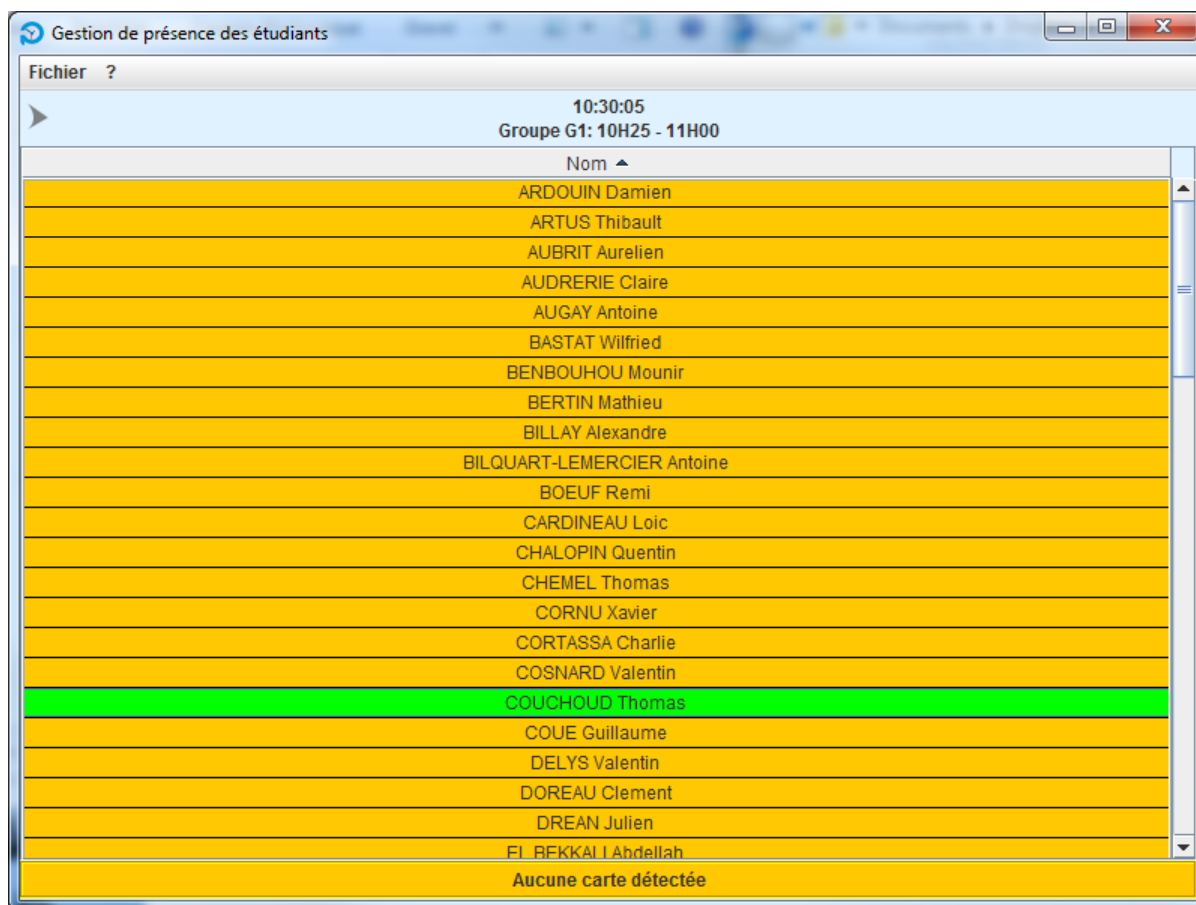


FIGURE 2.6 – Fenêtre principale.

On y observe les informations essentiels :

- L'heure actuelle, qui sera utilisée pour tracer la validation.
- La liste des groupes devant valider ainsi que la période dans laquelle nous nous trouvons actuellement pour ce groupe.
- La liste des élèves devant valider (en orange) et ceux ayant validé (en vert).
- Le statut du lecteur (ici "Aucune carte détectée").

De plus il est possible de faire apparaître différentes options de gestion du programme en appuyant sur **Ctrl + Alt + P** ou en appuyant sur la flèche dans le coin supérieur gauche de la fenêtre.



FIGURE 2.7 – Fenêtre principale avec carte du personnel.

## Configuration des groupes

Comme vu précédemment dans la partie 2.2.2, la gestion des groupes est une partie importante du logiciel et doit être la plus pratique possible afin de ne pas perdre de temps lors de la configuration.

Pour ce faire l'interface de gestion des groupes se présente simplement comme une liste de ceux déjà créés et pouvant être modifiés. On peut de plus ajouter de nouveaux groupes grâce à un bouton situé dans la partie inférieure.

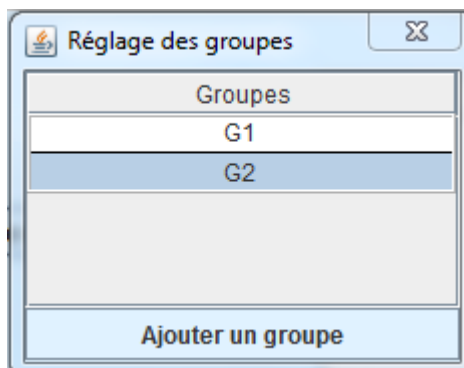


FIGURE 2.8 – Fenêtre avec les options de gestion.

Éditer un groupe se présente de la même manière sauf que l'on modifiera la liste des différents étudiants ou des périodes.

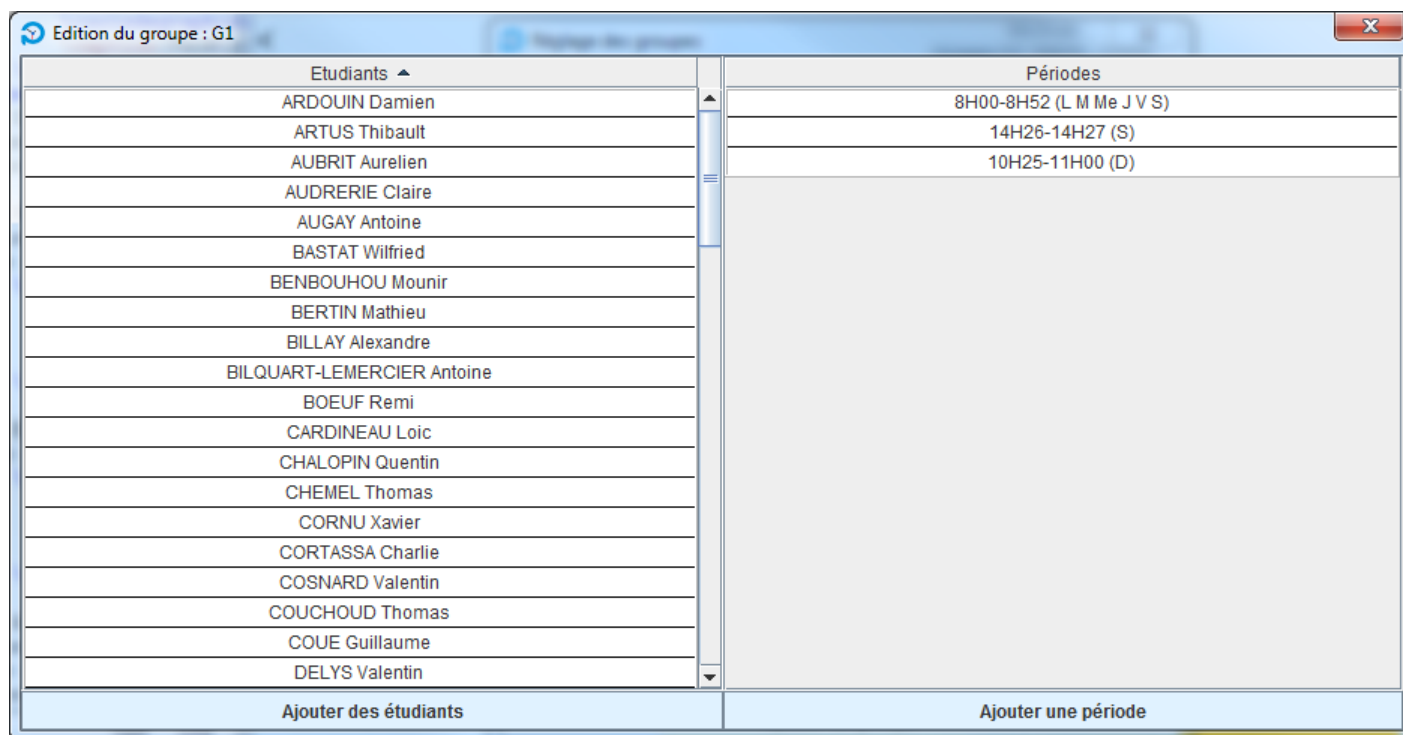


FIGURE 2.9 – Fenêtre d'édition d'un groupe.

A noter que, toujours dans un soucis de simplicité, l'ajout d'une nouvelle personne nous met à disposition la liste des différents étudiants connus pouvant être ajoutés au groupe. Le choix est ainsi plus simple et permet d'éviter d'éventuelles fautes de frappe.

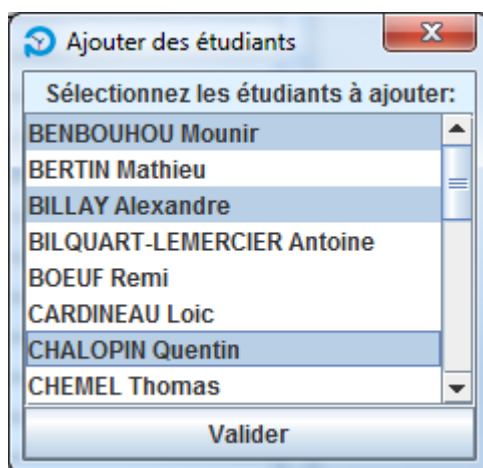


FIGURE 2.10 – Fenêtre permettant l'ajout d'étudiants à un groupe.



## 2.2.4 Base de donnée SQL

La base de donnée représente un support pour l'application afin de disposer de la liste des différents étudiants associés leur carte Atout-Centre. Celle-ci se présente sous une forme très simple.


Nom	Type	Longueur	Décimales	Non null	
CSN	varchar	18	0	<input checked="" type="checkbox"/>	 1
Firstname	varchar	100	0	<input checked="" type="checkbox"/>	
Lastname	varchar	100	0	<input checked="" type="checkbox"/>	

FIGURE 2.11 – Structure de la base de donnée.

- Un champ CSN représentant l'UID de la carte de l'étudiant.
- Un champ Lastname représentant le nom de l'étudiant.
- Un champ Firstname représentant le prénom de l'étudiant.

CSN	Firstname	Lastname
0450747AA63A80	Thomas	COUCHOUD
0451117AA63A80	Mathieu	Bertin
0451447AA63A80	Jason	Loyau
04548D7AA63A80	Wilfried	Bastat
0455538A813A80	Guillaume	Morand

FIGURE 2.12 – Exemple de base de donnée avec valeurs.

## 2.2.5 Résultats

Les résultats donnés par le programme consistent en la création de deux fichiers.

L'un *checked\_<année>.csv* représente une trace de toutes les validations (si l'option a été activée).

	A	B	C
1	[CET] 21/02/2015 19:17:25	Thomas C	0450747AA63A80
2	[CET] 21/02/2015 19:20:39	Thomas C	0450747AA63A80

FIGURE 2.13 – Trace de validation.

Le second est un fichier regroupant les absences d'un certain élève. Nous avons ainsi la date de l'absence, la période durant laquelle l'absence a été relevée ainsi que le nom de l'étudiant. Il se nomme *absent\_<nom>\_<année>\_<mois>.csv* et se présente sous cette forme :

	A	B	C
1	21/02/2015	14H24 - 14H25	T2
2	21/02/2015	14H26 - 14H28	T2
3	21/02/2015	18H00 - 19H00	T2

FIGURE 2.14 – Absences.

### 2.2.6 Problèmes rencontrés et solutions

Au cours de la réalisation de l'application, nous avons été confronté à un problème majeur : la gestion des différents groupes de façon indépendante. En effet, nous avons à la base réalisé le système de sorte que l'on ai qu'une seule liste d'étudiants devant valider toutes les deux heures durant une période. Cela était très peu flexible car on définissait l'heure de début et de fin d'une journée ce qui implique que le temps pour manger et autres temps libre sont comptés comme un cours. De plus, cela ne répondait pas du tout à ce qui était demandé puisque chaque classe n'a pas forcément les mêmes horaires.

Afin de contrer ce problème, nous avons mis en place la notion de groupes vu dans la partie 2.2.2. Nous avons donc séparé le "groupe principal" en plusieurs groupes ayant leur propres paramètres. Nous en avons aussi profité pour régler le problème des temps libres. Au lieu d'avoir une heure de début et de fin, où chaque élève doit valider toutes les deux heures, on demande plusieurs périodes où l'étudiant devra valider une fois durant cette période.

D'autres "problèmes" sont survenus mais ceux-ci n'était pas bloquant pour l'application. Ce sont principalement des améliorations du programme, le rendant plus simple ou permettant de répondre plus précisément à la demande du secrétariat.

- La liste des élèves devant valider se trouvait, dans une version précédente, dans un fichier CSV devant être crée à la main. Cela pouvait entrainer différentes complications notamment si le nom d'une personne était mal écrit. Maintenant on passe par le groupe qui est sérialisé.
- De la même manière, les membres du personnel était renseignés dans un fichier CSV. Dorénavant c'est dans la base de donnée SQL que ce trouve cette information. Cela évite la création d'un fichier supplémentaire ou d'éventuelles modifications non voulues de ces paramètres.
- Cependant, nous avons totalement abandonné l'idée de différencier les cartes du personnel. En effet en gardant contacte avec le groupe de DII4 qui travaille sur un projet similaire, nous avons pu obtenir un "prototype" de la base de donnée qui pourra être fournie en début d'année par la DTIC. Celle-ci n'inclut pas le personnel et par conséquent il nous faudrait modifier cette base de donnée pour l'adapter à notre programme. Ayant jugé cette fonctionnalité non essentielle pour l'application, nous avons remplacé l'ouverture du menu du personnel par un raccourcis clavier. Certes nous perdons en sécurité puisque toute personne connaissant le raccourcis peut dorénavant accéder au menu mais nous gagnons en simplicité pour une éventuelle importation d'une base de

donnée fournie par la DTIC.

- La création des fichiers de sortie du programme ne correspondait pas aux attentes du secrétariat. En effet pour chaque personne nous indiquions quand elle avait validé sa présence. Or c'est l'inverse qui est voulu, avoir seulement les moments où la personne n'était pas présente. Cela a été corrigé tout en laissant la possibilité d'avoir une trace de toutes les validations au cas où un problème pendant la création de ces fichiers surviendrait. En effet les absences sont notées à la fin d'une période. Et si, par on ne sait quel problème, le programme venait à se fermer durant une période (coupure électrique ou crash de l'ordinateur), nous ne pourrions pas récupérer les étudiants ayant validés.

## Chapitre 3

# Mise en place du travail réalisé

---

Durant la réalisation de notre projet, nous avons été amenés à le mettre en place dans des conditions "réelles" d'utilisation.

En effet, notre projet ayant pour objectif de gérer plusieurs classes d'étudiants, il nous était difficile de simuler une telle masse d'étudiant avec seulement nos deux cartes personnelles. Afin d'obtenir des résultats sur le bon fonctionnement du programme et éventuellement déceler les problèmes, nous avons tout d'abord entrepris d'installer un prototype dans le bureau de Mme Belair.

Après deux semaines d'utilisation, nous avons eut un retour favorable de sa part. Aucun problème majeur n'est survenu. Seul deux suggestions ont été proposées : l'une concernant un possible triage par ordre alphabétique des étudiants, et l'autre consistant à ajouter le temps d'absence dans les fichiers de sortie.

Nous avons par la suite eut l'opportunité d'étendre notre application aux besoins de la Faculté de médecine. Suite à une rencontre avec l'administration, nous avons pu déterminer précisément leurs attentes et ainsi adapter notre projet. La principale différence entre les deux utilisations réside dans le type d'événement à couvrir. En effet, Polytech' a besoin de définir un planning hebdomadaire pour chaque classe tandis que la Faculté de médecine utiliserait le programme dans le cas d'événements particuliers (conférence) adressés à l'ensemble des étudiants.

Cela entraine une modification importante du fonctionnement du logiciel notamment en détruisant la notion de groupe qui avait été introduite. De ce fait, nous avons refait l'application d'une manière vraiment différente en s'appuyant d'avantage sur la base de donnée qui est présente. Ainsi aucune donnée n'est stockée dans un fichier au fur et à mesure que le programme tourne, tout est dans la base de donnée et sera traité lorsque le personnel effectue l'exportation des résultats.

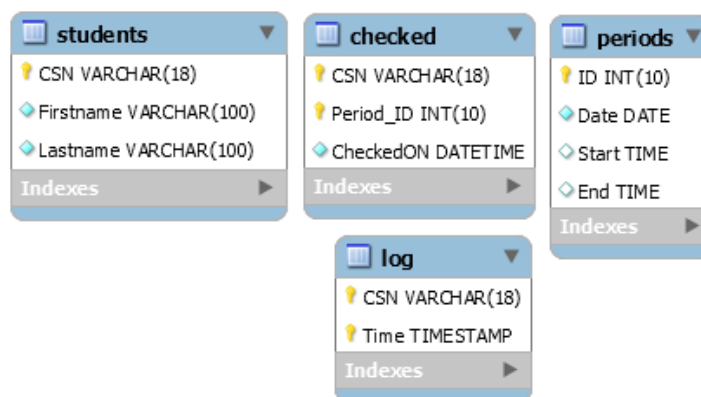


FIGURE 3.1 – Structure de la base de donnée.

- La table "Students" permet d'associer un nom à un numéro de carte. C'est d'ailleurs la seule qui est présente dans la version Polytech' du programme.
- La table "Checked" permet de connaître les périodes de présence d'un étudiant.
- La table "Periods" définit les périodes de validation avec une date ainsi que les heures de début et fin.
- La table Log où s'enregistre les validations de toutes les cartes.

A l'heure actuelle nous n'avons eu aucun retour sur la mise en place de ce projet avec la faculté de médecine.

## Chapitre 4

# Conclusion

---

Ce projet nous à permis de découvrir le mode des cartes à puce. En effet même si l'on utilise cette technologie très souvent, on ne s'y intéresse pas forcément et laissons la partie technique de cette innovation de coté. Nous avons commencé par puiser nos informations dans les normes ISO mais celles-ci se sont avérées bien plus compliquées que prévu. Nous nous sommes donc rabattu sur des information plus générales mais qui sont tout de même très instructives sur le domaine et nous ont permis de comprendre cette technologie que nous utilisons au quotidien.

De plus nous avons eu l'occasion de mener un projet de manière quasi autonome. Cela nous a permis de comprendre le travail de groupe et d'exploiter sa puissance en pouvant répartir la charge de travail afin d'avoir un résultat en peu de temps. Cela est passé par une recherche des informations nécessaires, par la création d'une application puis par l'utilisation de celle-ci.

Le développement de l'application nous à permis d'une part d'en connaître un peu plus sur JAVA. De plus, étant assez avancé dans ce langage, cela nous à permis d'utiliser de façon concrète nos connaissances.

Enfin nous avons eu l'opportunité de pouvoir mettre en place notre projet au près de Mme Belair au DI et à la fac de médecine. Cela eut un effet positif car il nous permis d'échanger avec des personnes extérieures au projet et susceptibles d'utiliser le programme. On peut ainsi voir où sont les difficultés d'utilisation et nous entraîne à expliquer le fonctionnement d'une application (ce qui n'est pas très évident).

D'un point de vue critique, nous pensons que notre application reste assez simple d'utilisation et intuitive pour le coté positif. Concernant le négatif, il reste des bugs qui peuvent être contraignants et surtout le code est un peu brouillon par endroits. Chacun à son point de vue sur un code brouillon mais dû aux nombreux changements effectués au cours du développement, il aurait pu être judicieux de refaire le code si le temps nous l'avait permis.

# Annexe A

## Liens utiles

---

Voici une petite liste d'url intéressantes au sujet de ce projet :

- Site de Polytech'Tours : [www.polytech.univ-tours.fr](http://www.polytech.univ-tours.fr)
- Cours sur les cartes à puce : [http://www.perso.telecom-paristech.fr/~urien/cours\\_cartes\\_urien-2008.pdf](http://www.perso.telecom-paristech.fr/~urien/cours_cartes_urien-2008.pdf)
- Norme ISO-7816 [http://www.cardwerk.com/smartcards/smartcard\\_standard\\_ISO7816-1.aspx](http://www.cardwerk.com/smartcards/smartcard_standard_ISO7816-1.aspx)
- Spécifications PC/SC <http://www.pcscworkgroup.com/specifications/specdownload.php>
- Page Wikipedia des cartes à puce : [http://fr.wikipedia.org/wiki/Carte\\_%C3%A0\\_puce](http://fr.wikipedia.org/wiki/Carte_%C3%A0_puce)
- Repository GitHub du projet : <https://github.com/MrCraftCod/RFID/>
- Javadoc javax.smartcardio <http://docs.oracle.com/javase/7/docs/jre/api/security/smartcardio/spec/javax/smartcardio/package-summary.html>
- Java SE 8 : <http://www.oracle.com/technetwork/java/javase/downloads/jre8-downloads-2133.html>
- Wamp : <http://www.wampserver.com/>

# Annexe B

## Installation du programme

---

Voici brièvement comment se déroule l'installation du programme. Il est nécessaire d'avoir trois éléments pour le faire fonctionner :

- Le programme étant codé en Java, il est nécessaire d'avoir celui-ci installé sur la machine. La version à prendre doit être Java (SE/JDK) 8 ou supérieure.
- Une base de donnée est nécessaire afin de stocker des informations supplémentaires. Nous avons personnellement utilisé wamp permettant d'avoir ce pré-requis de façon locale sur la machine en service. (La configuration d'un utilisateur pour la base de donnée SQL avec wamp est assez spéciale, pour une utilisation locale, il est nécessaire de paramétrer le client à "localhost". Les configurations équivalentes "%" ou "127.0.0.1" ne fonctionneront pas.)
- Enfin il est nécessaire d'avoir un lecteur de carte. Cela implique qu'il est peut-être nécessaire d'installer les drivers de celui-ci pour qu'il soit reconnu par l'ordinateur.

Un fois tout installé, il ne reste plus qu'à lancer le fichier JAR fournis avec Java pour que le programme démarre.



# Annexe C

## Structure du module TerminalReader

---

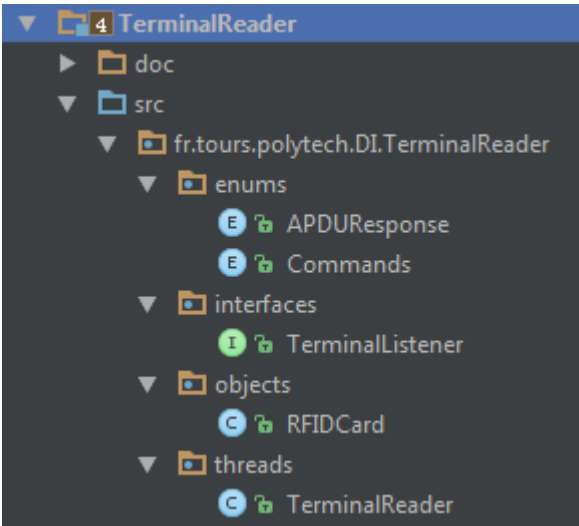


FIGURE C.1 – Arborescence des fichiers.

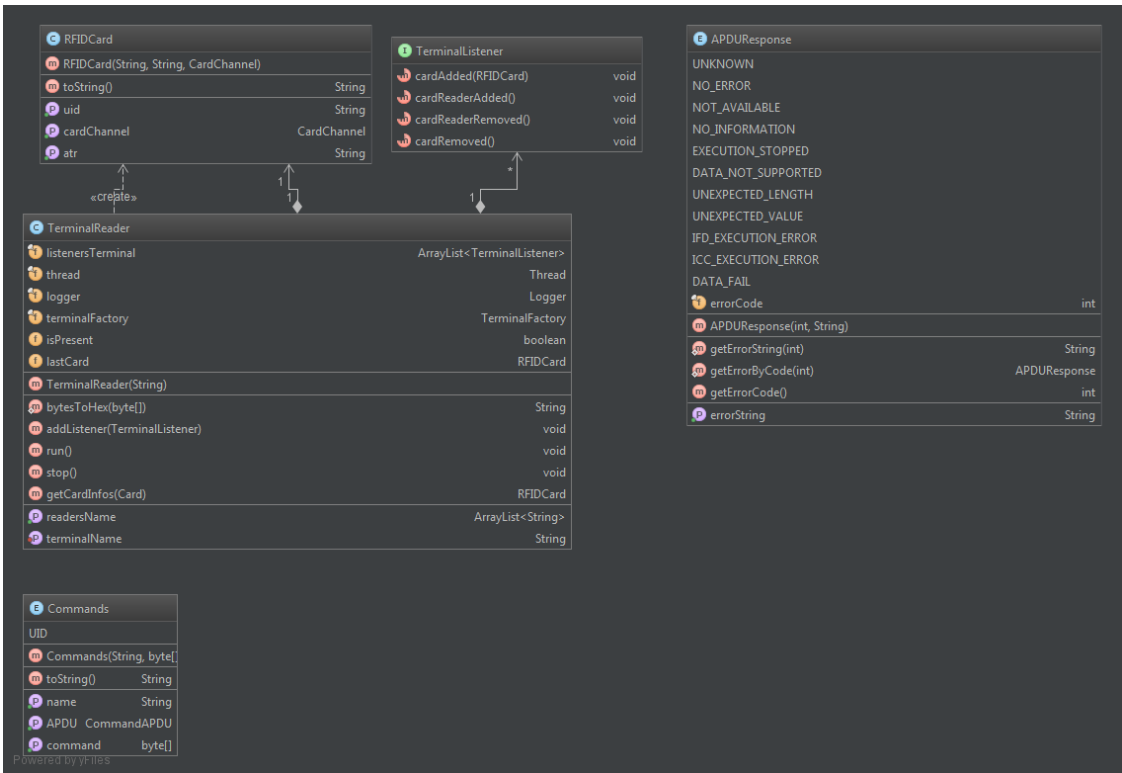


FIGURE C.2 – Diagramme UML.

# Annexe D

## Structure du programme

---

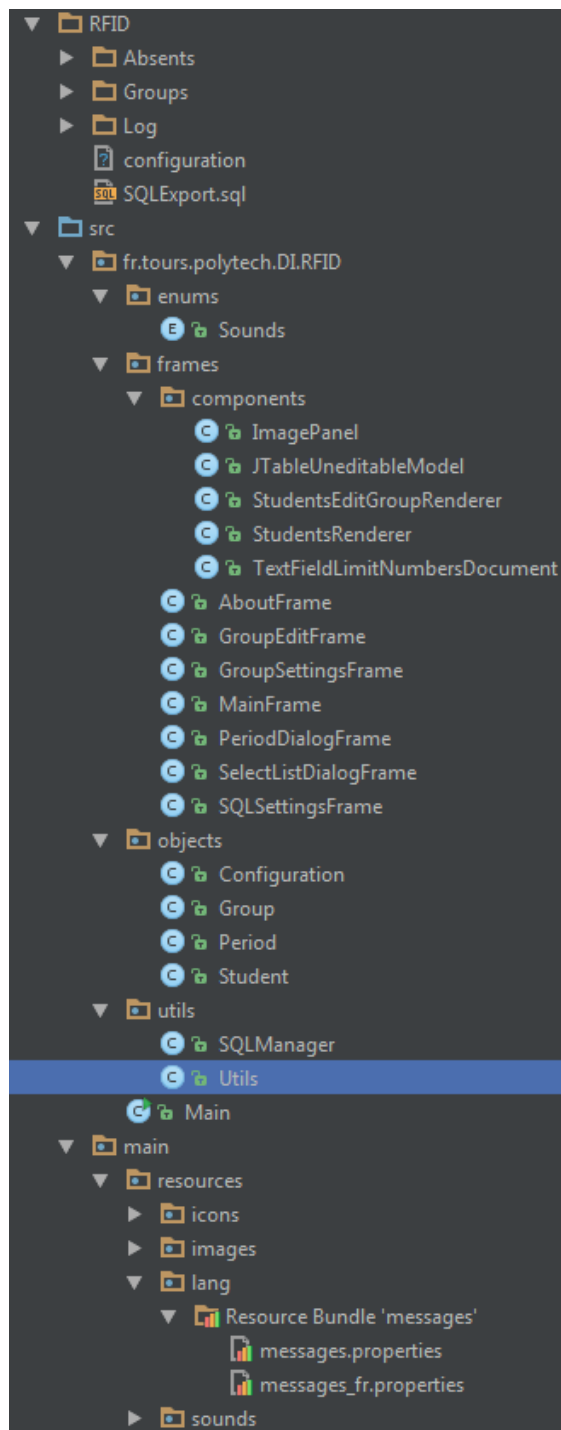


FIGURE D.1 – Arborescence des fichiers.



# Annexe E

## Fiche de suivi de projet PeiP

---

Séance n° 1 du 22/01/2015	Prise de contact avec l'encadrant. Découverte du sujet. Réception du lecteur de cartes. Recherche globale sur le sujet, test de la compatibilité du lecteur avec un ordinateur et de la carte étudiante avec le lecteur. Prise en main du fonctionnement de la carte grâce à différentes documentations trouvées sur internet.
Travail durant le temps libre du 23/01/2015 au 28/01/2015	Réalisation d'un prototype de l'application aux fonctionnalités minimales.
Séance n° 2 du 29/01/2015	Enrichissant de l'application avec par exemple l'implémentation d'une communication SQL, gestion des plages horaires et système d'administration pour le personnel. Début de rédaction du rapport de projet. Récupération des différents liens des documentations dont nous avons besoin.
Séance n° 3 du 05/02/2015	Recherches plus générales sur les cartes à puce (historique, usages des cartes, ainsi que les normes et protocoles utilisés).
Séance n° 4 du 12/02/2015	Rencontre avec l'encadrant du projet. Tentative d'un début d'analyse logicielle. Rencontre avec le secrétariat, prise de connaissance des formats de fichiers voulus. Amélioration du programme pour répondre à ces dernières conditions.
Séance n° 5 du 19/02/2015	Recherche supplémentaires sur les cartes (sécurité des cartes, système d'exploitation). Avancement du compte rendu final. Demande de renseignements sur les classes apprentis auprès de la secrétaire. Amélioration du programme (Correction de bugs mineurs dans le programme, ajout de la fonctionnalité d'enregistrement d'étudiant dans la base de données, début d'un nouveau système de gestion des élèves par groupe et par horaire).
Séance n° 6 du 26/02/2015	Due à l'absence de M. Makris, il nous a été impossible de mettre en place une session de récupération des numéros de cartes afin de constituer une base de donnée. Nous avons par conséquent avancé dans la rédaction de notre rapport.

Séance n° 7 du 12/03/2015	Avancée dans le compte rendu. Modification du programme pour pouvoir s'adapter à des configurations différentes de base de données. Récupération d'un ordinateur pour le laisser à la secrétaire et constituer une base de donnée. Installation des différents programmes nécessaires à l'utilisation de notre application sur ce dernier.
Séance n° 8 du 19/03/2015	Mise en place du système avec Mme Belair. Début d'une modification du programme pour la prise en compte d'un emploi du temps hebdomadaire. Avancée du rapport.
Séance n° 9 du 26/03/2015	Prise de contact avec Mme Belair pour obtenir les premiers résultats de la mise en place du système. Modification du programme afin de satisfaire différentes demandes de Mme Belair (liste triée alphabétiquement, ajout du temps d'absence). Création d'une dérivée du programme plus adapté aux attentes de la médecine.
Séance n° 10 du 02/04/2015	RDV avec la fac de médecine. Début des modifications du programme pour s'adapter aux besoins. Avancement dans le rapport.
Séance n° 11 du 09/04/2015	Mise en place du système dans le grand amphi de la fac de médecine. Finalisation du rapport. Début de la création de notre présentation pour la soutenance.
Séance n° 12 du 16/04/2015	Derniers ajustements du rapport. Entraînement pour la soutenance.
Séance n° 13 du 23/04/2015	Soutenance du projet.

# Application de gestion des présences des étudiants grâce à leur carte étudiant

---

## Rapport de projet S2

**Résumé :** Ce projet consiste à découvrir le fonctionnement de la technologie des cartes à puces (smart-card) que nous rencontrons quotidiennement (cartes de bus, carte étudiante, carte vitale, etc..). Nous concrétiserons ces recherches par la mise en place d'une application permettant la gestion de présence d'élèves à leurs cours. Pour cela nous aurons à disposition un travail précédemment réalisé par un ancien groupe. Cependant il est à noter que ce projet avait été réalisé avec une ancienne génération des cartes étudiantes.

**Mots clé :** RFID, PC/SC, Carte à puce, Gestion de présence

**Abstract :** This project consists in discovering how operate smart cards that we meet in our daily life (bus card, student card, etc..). We'll achieve these searches by the introduction of an application allowing the management of the students' presence in their classes. To help us in that task, we will have access to the work that have been done by a previous group. However it's important to note that this project had been realized with an older generation of our cards.

**Keywords :** RFID, PC/SC, Smart card, Presence management

Auteur(s)

**Thomas Couchoud**

[thomas.couchoud@etu.univ-tours.fr]

**Victor Coleau**

[victor.coleau@etu.univ-tours.fr]

Encadrant(s)

**Pascal Makris**

[makris@univ-tours.fr]

**Polytech Tours  
Département DI**

Ce document a été formaté selon le format EPUProjetPeiP.cls (N. Monmarché)

École Polytechnique de l'Université de Tours  
64 Avenue Jean Portalis, 37200 Tours, France  
<http://www.polytech.univ-tours.fr>