

République Algérienne Démocratique et Populaire  
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique  
Université 8Mai 1945 – Guelma  
Faculté des Sciences et de la Technologie  
Département d'Electronique et Télécommunications



## **Mémoire de Fin d'Etude**

**Pour l'Obtention du Diplôme de Master Académique**

Domaine : **Sciences et Techniques**

Filière : **Télécommunications**

Spécialité : **Réseaux et télécommunications**

---

## **Conception d'un Système de Suivi de Présence en Java**

---

Présenté par :

**Ferkous Dhiyaeddine**

Sous la direction de :

**Dr. Abderezzaq Halassi**

**Année Universitaire : 2024-2025**

# Remerciements

J'EXPRIME MA PROFONDE GRATITUDE À DIEU, QUI M'A PERMIS  
D'ACCOMPLIR CE TRAVAIL.

JE TIENS À REMERCIER SINCÈREMENT **MES PARENTS** POUR LEUR  
SOUTIEN INCONDITIONNEL, LEURS ENCOURAGEMENTS CONSTANTS  
ET LES NOMBREUX SACRIFICES QU'ILS ONT CONSENTIS TOUT AU  
LONG DE MON PARCOURS.

JE REMERCIE ÉGALEMENT MON ENCADRANT, **DR HALASSI  
ABDEREZZAQ**, POUR LES EFFORTS QU'IL A DÉPLOYÉS AFIN DE  
M'AIDER, ME CONSEILLER, M'ENCOURAGER ET CORRIGER MON  
TRAVAIL AVEC PATIENCE ET BIENVEILLANCE.

JE SUIS RECONNAISSANT(E) AUX **MEMBRES DU JURY** POUR AVOIR  
ACCEPTÉ D'EXAMINER CE TRAVAIL ET POUR L'ATTENTION QU'ILS  
LUI ONT PORTÉE.

JE REMERCIE AUSSI L'ENSEMBLE DU CORPS **ENSEIGNANT** DU  
DÉPARTEMENT ELN & TLC POUR LA QUALITÉ DE L'ENSEIGNEMENT  
DISPENSÉ ET LEUR CONTRIBUTION À MA FORMATION  
UNIVERSITAIRE.

ENFIN, JE REMERCIE **TOUTES LES PERSONNES**, DE PRÈS OU DE  
LOIN, AYANT CONTRIBUÉ À LA RÉALISATION DE CE TRAVAIL.  
QU'ELLES TROUVENT ICI L'EXPRESSION DE MA SINCÈRE  
RECONNAISSANCE.

# *Dédicace*

JE DÉDIE CE MODESTE TRAVAIL, À MES PARENTS, À  
MA SOURCE DE GÉNÉROSITÉ  
ET DE PATIENCE TOUT AU LONG DE MA CARRIÈRE  
SCOLAIRE. QUE DIEU VOUS PROTÈGES, VOUS PRÊTES  
BONNE SANTÉ ET LONGUE VIE.

À MES FRÈRES ET SŒURS ET SA PETITE FAMILLE,  
QUI M'ONT TOUJOURS INDIQUÉ  
LA BONNE VOIE ET QUI ONT SU M'AIDER.

AUX PERSONNES QUI M'ONT ACCOMPAGNÉ DURANT  
MON CURSUS UNIVERSITAIRE,

À MES AMIES POUR SES ENCOURAGEMENTS  
PERMANENTS, ET LEUR SOUTIEN

# Résumé

Ce travail explore le développement d'un système automatisé de gestion des présences, s'appuyant sur la technologie RFID. Concrètement, il s'agit de concevoir et mettre en place une solution pratique où une carte ESP32, couplée à un lecteur RFID RC522 et à un écran LCD 1602A, permet d'identifier les étudiants dès leur passage à l'entrée d'une salle. Cette identification se fait en temps réel : les informations captées sont immédiatement transmises à un serveur Java qui se charge de les traiter et de les enregistrer dans une base de données MySQL. En parallèle, une application conviviale a été développée pour offrir aux utilisateurs des interfaces intuitives de gestion des étudiants, des plannings et du suivi des présences. Ce mémoire met en lumière l'efficacité de l'automatisation dans le contexte éducatif, tout en soulignant la simplicité, la fiabilité et les avantages pratiques d'une telle approche technologique.

# Abstract

This work explores the development of an automated attendance management system based on RFID technology. The main objective is to design and implement a practical solution where an ESP32 board, connected to an RFID RC522 reader and an LCD 1602A display, enables real-time identification of students as they enter a classroom. The captured data is instantly sent to a Java server, which processes and records it in a MySQL database. Simultaneously, a user-friendly application was developed to provide intuitive interfaces for managing students, schedules, and attendance tracking. This project highlights the efficiency of automation in the educational context while emphasizing the simplicity, reliability, and practical benefits of such a technological approach.

# ملخص

يستكشف هذا العمل تطوير نظام آلي لإدارة الحضور قائم على تقنية RFID. الهدف الرئيسي هو تصميم وتنفيذ حل عملي يعتمد على لوحة ESP32 متصلة بقارئ RFID RC522 وشاشة LCD 1602A ، لتمكين التعرف على الطلاب في وقت حقيقي عند دخولهم إلى قاعة الدراسة.

تُرسل البيانات المقروءة مباشرة إلى خادم Java لتعميق المعالجة وتخزينها في قاعدة بيانات MySQL. وفي نفس الوقت، تم تطوير تطبيق ذو واجهات بسيطة ومرنة تسهل عملية إدارة الطلاب وجدولهم وتتبع الحضور بكل فاعلية.

يُبين هذا المشروع كفاءة الأتمتة في السياق التعليمي، مسلطاً الضوء على بساطة وموثوقية وفوائد هذا النهج التقني في الواقع العملي.

# Table Des Matières

|                                                                    |    |
|--------------------------------------------------------------------|----|
| <i>Remerciements</i> .....                                         | 2  |
| <i>Dédicace</i> .....                                              | 3  |
| <b>Résumé</b> .....                                                | 4  |
| <b>Table Des Matières</b> .....                                    | I  |
| <b>Liste Des Tableaux</b> .....                                    | V  |
| <b>Liste Des Figures</b> .....                                     | VI |
| <b>Introduction Générale</b> .....                                 | 1  |
| <b>Chapitre I Technologie RFID</b> .....                           | 3  |
| <b>I. 1 Introduction</b> .....                                     | 4  |
| I.1.1 Définition de la RFID (Radio Frequency Identification) ..... | 4  |
| I.1.2. Historique et évolution de la technologie .....             | 4  |
| <b>I.2 Principes de fonctionnements</b> .....                      | 5  |
| I.2.1. Architecture d'un système RFID.....                         | 5  |
| I.2.2 Tags RFID (actifs, passifs, semi-passifs) .....              | 6  |
| I.2.3. Lecteurs RFID .....                                         | 7  |
| I.2.4. Antennes et fréquences radio .....                          | 8  |
| I.2.5. Processus de communication .....                            | 8  |
| I.2.5.1. Émission/réception d'ondes radio.....                     | 8  |
| I.2.5.2. Stockage et transmission des données .....                | 9  |

|                                                                              |           |
|------------------------------------------------------------------------------|-----------|
| <b>I.3. Applications de la RFID .....</b>                                    | <b>9</b>  |
| <b>I.3.1. Domaines d'utilisation .....</b>                                   | <b>9</b>  |
| <b>I. 3.1.1. Logistique et gestion de stock.....</b>                         | <b>9</b>  |
| <b>I. 3.1.2. Santé .....</b>                                                 | <b>9</b>  |
| <b>I. 3.1.3. Transport .....</b>                                             | <b>9</b>  |
| <b>I. 3.1.4. Commerce .....</b>                                              | <b>9</b>  |
| <b>I.3.1.5. Éducation .....</b>                                              | <b>10</b> |
| <b>I.3.1.6. Agriculture.....</b>                                             | <b>10</b> |
| <b>I.3.2. Avantages spécifiques pour la gestion de présence.....</b>         | <b>10</b> |
| <b>I.4. Normes et standards RFID .....</b>                                   | <b>11</b> |
| <b>I.4.1. Normes ISO/IEC .....</b>                                           | <b>11</b> |
| <b>I.4.2. Protocoles de communication .....</b>                              | <b>11</b> |
| <b>I.4.3. Réglementations des fréquences radio.....</b>                      | <b>11</b> |
| <b>I.5. Enjeux et défis .....</b>                                            | <b>11</b> |
| <b>I.5.1. Sécurité et confidentialité.....</b>                               | <b>11</b> |
| <b>I.5.2. Limitations techniques .....</b>                                   | <b>12</b> |
| <b>I.5.3. RFID et IoT .....</b>                                              | <b>12</b> |
| <b>I.6 Conclusion .....</b>                                                  | <b>13</b> |
| <br><b>Chapitre II :.....</b>                                                | <b>14</b> |
| <br><b>Java / MySQL.....</b>                                                 | <b>14</b> |
| <b>II.1 Introduction .....</b>                                               | <b>15</b> |
| <b>II.2 Java .....</b>                                                       | <b>15</b> |
| <b>II.2.1 Présentation générale de Java .....</b>                            | <b>15</b> |
| <b>II.2.2 JDK et JRE.....</b>                                                | <b>15</b> |
| <b>II.2.3 POO, JVM et portabilité.....</b>                                   | <b>16</b> |
| <b>II.2.4 Quatre piliers de POO .....</b>                                    | <b>16</b> |
| <b>II.2.5 Java Swing pour les interfaces graphiques .....</b>                | <b>17</b> |
| <b>II.2.5.1 Architecture de Swing (JFrame, JPanel, LayoutManagers) .....</b> | <b>17</b> |

|                                                                                  |           |
|----------------------------------------------------------------------------------|-----------|
| II.2.5.2 Gestion des événements (ActionListener, MouseListener...)               | 17        |
| II.2.5.3 Personnalisation                                                        | 18        |
| II.2.5.4 Performance et stabilité de Java dans le traitement des données         | 18        |
| II.2.5 Avantages de Java (réutilisabilité, large écosystème)                     | 18        |
| II.2.6. Java sur le marché mondial                                               | 19        |
| <b>II.3 Bases de données relationnelles</b>                                      | <b>20</b> |
| II.3.1 Définitions et concepts clés                                              | 20        |
| II.3.2 Rôle et fonctionnement d'un SGBD                                          | 20        |
| II.3.3 MySQL                                                                     | 21        |
| II.3.4 PhpMyAdmin                                                                | 21        |
| <b>II.4 Le couplage Java/MySQL</b>                                               | <b>21</b> |
| II.4.1 Avantages de combinaison java/MySQL                                       | 22        |
| II.4.2 L'intégration Java/MySQL dans le développement moderne                    | 22        |
| <b>II. 5 Conclusion</b>                                                          | <b>23</b> |
| <br><b>Chapitre III Implémentation pratique de la pointeuse .</b>                | <b>24</b> |
| <b>III.1 Introduction</b>                                                        | <b>25</b> |
| <b>III.2.Présentation de l'ESP32, du module RC522 et de l'écran LCD 1602A ..</b> | <b>25</b> |
| III.2.1. Architecture de ESP32                                                   | 25        |
| III.2.2 RC522                                                                    | 28        |
| III.2.2.1. Principe de fonctionnement du RC522                                   | 28        |
| III.2.3.Écran LCD 1602A avec module I2C                                          | 29        |
| <b>III.3.Configuration matérielle</b>                                            | <b>29</b> |
| III.3.1.Liste du matériel nécessaire                                             | 29        |
| III.3.2. Schéma de connexion entre l'ESP32 et le RC522                           | 30        |
| III.3.3. Schéma de connexion entre l'ESP32 et le LCD I2C 1602A                   | 31        |
| III.3.4. Schéma PCB                                                              | 31        |
| <b>III.4. Configuration logicielle</b>                                           | <b>32</b> |
| III.4.1. Présentation de l'environnement de développement Arduino                | 32        |

|                                                                                      |           |
|--------------------------------------------------------------------------------------|-----------|
| III.4.2. Installation des bibliothèques nécessaires pour l'ESP32,RC522et 1602A ..... | 33        |
| III.4.3.Code de configuration de l'ESP32 .....                                       | 34        |
| III.4.4.Protocoles utilisés .....                                                    | 37        |
| <b>III.5. Présentation de l'application Java .....</b>                               | <b>38</b> |
| III.5.1. Vue d'ensemble de l'application.....                                        | 38        |
| III.5.2. Interface principale (Login).....                                           | 38        |
| III.5.3. Interface d'accueil (Home Page) .....                                       | 40        |
| III.5.4.Liste des étudiants (Students List) .....                                    | 42        |
| III.5.5. Gestion des présences (Attendance) .....                                    | 43        |
| III.5.6. Planning des examens (Exam Schedules) .....                                 | 45        |
| III.5.7. Emploi du temps des étudiants (Time Tables) .....                           | 47        |
| III.5.8. Planning des enseignants (Teachers Schedules).....                          | 49        |
| III.5.9. Interface du serveur RFID (RFID Server) .....                               | 52        |
| <b>III.5.10. Conclusion.....</b>                                                     | <b>60</b> |
| III.5.11. Perspectives de développement.....                                         | 60        |
| <i>Conclusion Générale.....</i>                                                      | <i>62</i> |
| <b>Références bibliographiques .....</b>                                             | <b>64</b> |

# Liste Des Tableaux

|                                                                  |    |
|------------------------------------------------------------------|----|
| Tableau I.1.Comparaison entre les diffèrent tags RFID .....      | 7  |
| Tableau II.1. Schéma de connexion entre l'ESP32 et le RC522..... | 30 |

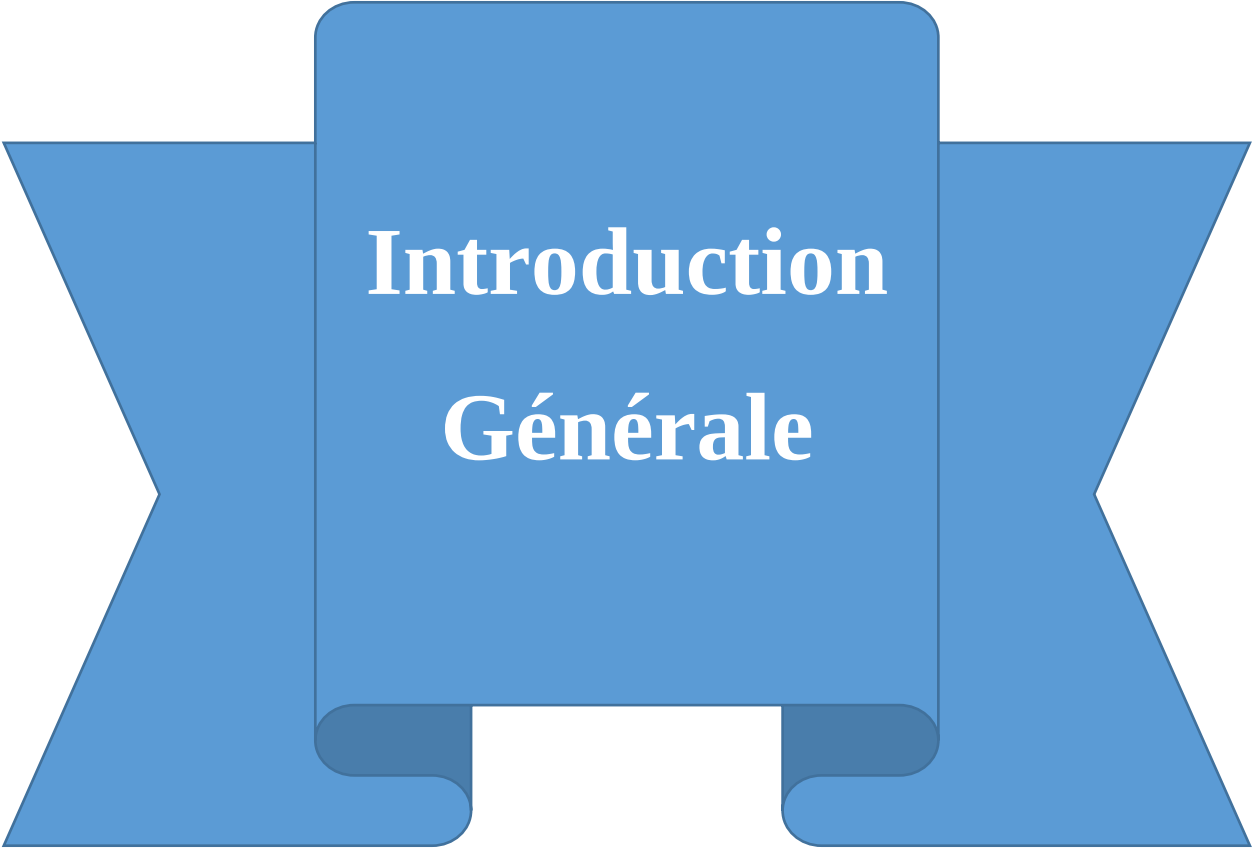
## Liste Des Figures

|                                                                            |    |
|----------------------------------------------------------------------------|----|
| Figure I.1 : Architecture de la RFID .....                                 | 5  |
| Figure I.2 : Émission/réception d'ondes radio.....                         | 9  |
| Figure I.3 : RFID et IOT .....                                             | 13 |
| Figure II.1 : JDK et JRE .....                                             | 16 |
| Figure II.2 : La popularité de Java .....                                  | 20 |
| Figure III.1 : L'ESP32 .....                                               | 26 |
| Figure III.2 : L'Architecture de l'ESP32.....                              | 27 |
| Figure III.3 : RC522 .....                                                 | 28 |
| Figure III.4 : Afficheur I2C LCD 1602A.....                                | 29 |
| Figure III.5 : Schéma de connexion entre l'ESP32 et le LCD I2C 1602A ..... | 31 |
| Figure III.6 : Schéma PCB .....                                            | 31 |
| Figure III.7 : Environnement de développement Arduino.....                 | 32 |
| Figure III.8 : Gestionnaire de l'ESP32.....                                | 33 |
| Figure III.9 : Bibliothèques RC522.....                                    | 33 |
| Figure III.10 : Bibliothèque Liquid Crystal I2C.....                       | 34 |
| Figure III.11 : Interface principale .....                                 | 38 |
| Figure III.12 : Interface d'accueil.....                                   | 40 |
| Figure III.13 : Interface de Liste des étudiants .....                     | 42 |
| Figure III.14 : Attendance interface .....                                 | 43 |
| Figure III.15 :Planning des examens .....                                  | 45 |
| Figure III.16 : Interface d'Emploi du temps .....                          | 47 |
| Figure III.17 : Modification d'Emploi du temps .....                       | 47 |
| Figure III.18 : Planning des enseignants .....                             | 49 |
| Figure III.19 : RFID server .....                                          | 52 |

# Liste Des Abréviations

|             |                                                          |
|-------------|----------------------------------------------------------|
| <b>BDD</b>  | <b>Base De Données(Database)</b>                         |
| <b>RFID</b> | <b>Radio-Frequency Identification</b>                    |
| <b>IOT</b>  | <b>Internet Of Things</b>                                |
| <b>ISO</b>  | <b>International Organization for Standardization</b>    |
| <b>MIT</b>  | <b>Massachusetts Institute of Technology</b>             |
| <b>IEEE</b> | <b>Institute of Electrical and Electronics Engineers</b> |
| <b>USB</b>  | <b>Universal Serial Bus</b>                              |
| <b>LF</b>   | <b>Low Frequency</b>                                     |
| <b>HF</b>   | <b>High Frequency</b>                                    |
| <b>UHF</b>  | <b>Ultra High Frequency</b>                              |
| <b>IEC</b>  | <b>International Electrotechnical Commission</b>         |
| <b>ASK</b>  | <b>Amplitude Shift Keying</b>                            |
| <b>PSK</b>  | <b>Phase Shift Keying</b>                                |
| <b>EPC</b>  | <b>Electronic Product Code</b>                           |
| <b>SQL</b>  | <b>Structured Query Language</b>                         |
| <b>IRM</b>  | <b>Magnetic Resonance Imaging</b>                        |
| <b>SPA</b>  | <b>Single Page Application</b>                           |
| <b>EAS</b>  | <b>Electronic Article Surveillance</b>                   |
| <b>ETSI</b> | <b>European Telecommunications Standards Institute</b>   |
| <b>AES</b>  | <b>Advanced Encryption Standard</b>                      |
| <b>RSA</b>  | <b>Rivest–Shamir–Adleman</b>                             |
| <b>AWS</b>  | <b>Amazon Web Services</b>                               |
| <b>WAN</b>  | <b>Wide Area Network</b>                                 |
| <b>AI</b>   | <b>Artificial Intelligence</b>                           |
| <b>JDK</b>  | <b>Java Development Kit</b>                              |

|             |                                                                |
|-------------|----------------------------------------------------------------|
| <b>JVM</b>  | <b>Java Virtual Machine</b>                                    |
| <b>JRE</b>  | <b>Java Runtime Environment</b>                                |
| <b>SRAM</b> | <b>Static Random Access Memory</b>                             |
| <b>RAM</b>  | <b>Random Access Memory</b>                                    |
| <b>ROM</b>  | <b>Read-Only Memory</b>                                        |
| <b>RTC</b>  | <b>Real-Time Clock</b>                                         |
| <b>SDA</b>  | <b>Serial Data Line</b>                                        |
| <b>ADC</b>  | <b>Analog-to-Digital Converter</b>                             |
| <b>DAC</b>  | <b>Digital-to-Analog Converter</b>                             |
| <b>SPI</b>  | <b>Serial Peripheral Interface</b>                             |
| <b>I2C</b>  | <b>Inter-Integrated Circuit</b>                                |
| <b>UART</b> | <b>Universal Asynchronous Receiver/Transmitter</b>             |
| <b>RST</b>  | <b>Reset</b>                                                   |
| <b>IDE</b>  | <b>Integrated Development Environment</b>                      |
| <b>VCC</b>  | <b>Voltage at the Common Collector (positive power supply)</b> |
| <b>GND</b>  | <b>Ground</b>                                                  |
| <b>MOSI</b> | <b>Master Out Slave In</b>                                     |
| <b>MISO</b> | <b>Master In Slave Out</b>                                     |
| <b>SDA</b>  | <b>Serial Data Line</b>                                        |
| <b>SCK</b>  | <b>Serial Clock</b>                                            |
| <b>VIN</b>  | <b>Voltage Input</b>                                           |



# **Introduction Générale**

Aujourd'hui, la technologie fait pleinement partie de notre quotidien. Nous l'utilisons chaque jour, que ce soit pour apprendre, communiquer ou travailler. Elle transforme notre manière de faire les choses, souvent pour le mieux.

Dans le domaine de l'éducation, la technologie peut jouer un rôle important pour résoudre certains problèmes. Par exemple, dans de nombreuses écoles et universités, la présence des étudiants est encore enregistrée manuellement. Cette méthode, en plus d'être lente, manque parfois de fiabilité et peut entraîner des erreurs ou des oublis.

C'est ici que l'automatisation devient particulièrement utile. Automatiser une tâche consiste à la confier à un système ou à une machine, sans intervention humaine directe. Cela permet de gagner du temps, de réduire les erreurs et de se concentrer sur des tâches plus essentielles.

Dans notre projet, nous avons utilisé la technologie **RFID** pour automatiser la gestion des présences. Chaque étudiant dispose d'une carte RFID. Lorsqu'il entre dans la salle, un lecteur détecte automatiquement sa carte et transmet les informations à un serveur, sans qu'aucune action manuelle ne soit nécessaire.

Notre système repose sur une carte **ESP32**, un lecteur **RFID RC522**, un écran **LCD**, un serveur développé en **Java**, et une base de données **MySQL**. Nous avons également conçu une application pour gérer les données des étudiants, leurs emplois du temps et leurs présences.

Ce projet démontre que l'automatisation, rendue possible grâce à la technologie, permet de simplifier, d'accélérer et d'optimiser la gestion quotidienne. Elle contribue à moderniser les établissements éducatifs et à assurer un suivi plus rigoureux et fiable. Il s'agit d'une solution pratique et efficace pour améliorer le fonctionnement du monde de l'éducation.



# Chapitre I

## Technologie RFID

## I. 1 Introduction

### I.1.1 Définition de la RFID (Radio Frequency Identification)

La RFID, ou identification par radiofréquence, est une méthode intelligente pour "donner une voix" aux objets. Imaginez un livre en bibliothèque équipé d'une étiquette minuscule : grâce à des ondes radio, un lecteur peut interroger cette étiquette à distance, sans contact visuel ni manipulation. Cette étiquette (appelée tag) contient une puce électronique qui stocke des données – comme un numéro unique – et une antenne pour communiquer. C'est un peu comme un passeport numérique pour les objets, utilisé dans les entrepôts pour tracer des palettes, dans les hôpitaux pour identifier des équipements, ou même dans votre carte de métro.

Les bases techniques remontent aux travaux sur les radars pendant la Seconde Guerre mondiale, mais c'est dans les années 2000 que la RFID a explosé, notamment grâce aux normes ISO 18000 (pour la gestion des fréquences) et aux innovations des entreprises comme Texas Instruments. Des chercheurs du MIT ont d'ailleurs démontré, dans une étude de 2010, comment cette technologie réduisait de 30% les erreurs d'inventaire par rapport aux codes-barres.

### I.1.2. Historique et évolution de la technologie

L'histoire de la RFID ressemble à une saga industrielle. Tout commence en 1945 : les Britanniques utilisent des systèmes radio passifs pour distinguer leurs avions des ennemis – une première forme de RFID. Dans les années 1970, Mario Cardullo dépose un brevet pour un "dispositif d'identification par radio", inspiré par la crise pétrolière et le besoin de mieux gérer les stocks.

Mais le vrai tournant arrive en 2003, quand Walmart exige que ses fournisseurs adoptent la RFID pour suivre les palettes. Les étiquettes coûtaient alors 1 dollar pièce ; aujourd'hui, grâce à des géants comme Alien\_Technology, leur prix a chuté à quelques centimes. Saviez-vous que votre animal de compagnie a peut-être une puce RFID sous la peau ? Cette pratique, lancée dans les années 90, montre comment la technologie est passée des usines à notre quotidien.

D'après un rapport de l'IEEE de 2022, la RFID représente désormais un marché de 15 milliards de dollars, dopé par l'IoT et la logistique 4.0. Et ce n'est qu'un début : des laboratoires comme celui de l'université de Cambridge travaillent sur des tags biodégradables, capables de disparaître après usage.

## I.2 Principes de fonctionnements

### I.2.1. Architecture d'un système RFID

#### I.2.1.1. Tag RFID

- Fonction : Stocker et transmettre des données.
- Composants :
  - ✓ Puce électronique : Mémoire (lecture seule ou réinscriptible).
  - ✓ Antenne : Communication sans fil avec le lecteur.

#### I.2.1.2. Lecteur RFID

- Fonction : Activer les tags, lire/écrire des données.
- Composants :
  - ✓ Module radio : Émet et reçoit des ondes.
  - ✓ Interface de communication : Transmet les données au backend (USB, Wi-Fi, Ethernet).

#### I.2.1.3. Antenne RFID

- Fonction : Convertir les signaux électriques en ondes radio (et vice versa).
- Types : Directionnelle (portée longue) ou omnidirectionnelle (couverture large).

#### I.2.1.4. Système Backend

- Fonction : Traiter, stocker et exploiter les données.
- Exemples :
  - ✓ Logiciel de gestion de stock.
  - ✓ Base de données centralisée (ex : suivi les présencesdes étudiantes).

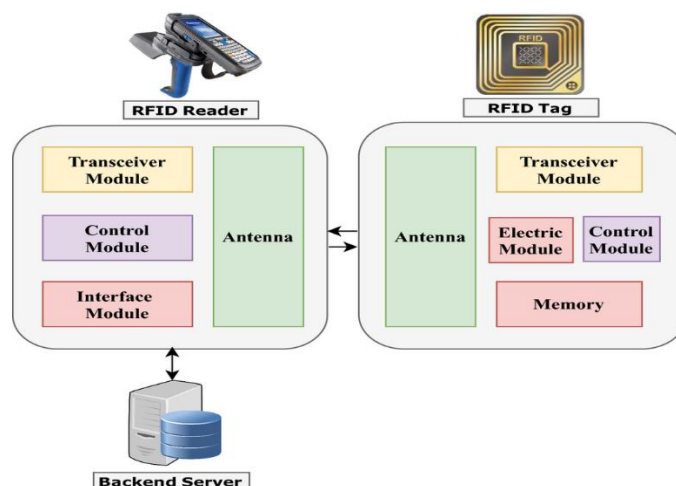


Figure I.1. Architecture de la RFID

## **I.2.2 Tags RFID (actifs, passifs, semi-passifs)**

### **I.2.2.1. Tags RFID passifs**

Les tags RFID passifs sont les plus discrets de la famille. Sans batterie, ils s'alimentent grâce à l'énergie émise par le lecteur, un peu comme une feuille qui s'anime sous une brise légère. On les retrouve partout où la simplicité prime : dans les badges d'accès des entreprises, les étiquettes anti-vol des magasins, ou même les livres des bibliothèques universitaires. Leur force ? Une durée de vie quasi illimitée et une résistance aux chocs, à la poussière ou à l'humidité. À l'université de Stanford, par exemple, ces tags ont remplacé les codes-barres, permettant aux étudiants d'emprunter des ouvrages en un seul geste, sans file d'attente.

### **I.2.2.2. Tags RFID actifs**

Les tags actifs, eux, sont les bavards du groupe. Équipés d'une batterie, ils émettent des signaux en continu, comme un phare guidant les navires dans la nuit. Leur portée impressionnante – jusqu'à plusieurs centaines de mètres – en fait des alliés indispensables pour les défis complexes. Dans les ports de Rotterdam, ces tags collés aux conteneurs permettent de localiser une cargaison perdue en quelques secondes. Les hôpitaux les utilisent aussi pour tracer des équipements vitaux, comme les défibrillateurs, garantissant qu'ils soient toujours disponibles en cas d'urgence.

### **I.2.2.3. Tags RFID semi-passifs**

Entre les deux, les tags semi-passifs jouent les équilibristes. Leur batterie alimente des capteurs sophistiqués – température, luminosité, mouvement – mais laissent au lecteur le soin de communiquer les données. Au Louvre, ces tags surveillent discrètement l'environnement des tableaux de maître, mesurant l'exposition à la lumière pour préserver les couleurs fragiles. Dans les vignobles de Bordeaux, ils aident les viticulteurs à ajuster l'irrigation en analysant l'humidité du sol, alliant tradition et innovation.

Tableau I.1 : Comparaison entre les différents tags RFID

| Caractéristiques             | Tags Passifs                                               | Tags Actifs                                                 | Tags Semi-Passifs                                                                          |
|------------------------------|------------------------------------------------------------|-------------------------------------------------------------|--------------------------------------------------------------------------------------------|
| <b>Alimentation</b>          | Aucune batterie.<br>S'alimentent via l'énergie du lecteur. | Batterie intégrée pour émettre des signaux.                 | Batterie pour la puce, mais communication via énergie du lecteur.                          |
| <b>Portée</b>                | Court (qq cm à 10 m max).                                  | Longue (jusqu'à 100 m).                                     | Moyenne (10-50 m).                                                                         |
| <b>Durée de vie</b>          | Illimitée (pas de batterie).                               | Limitée (2-5 ans selon la batterie).                        | Longue (5-10 ans, batterie pour capteurs uniquement).                                      |
| <b>Coût</b>                  | Très économique.                                           | Élevé.                                                      | Modéré à élever.                                                                           |
| <b>Applications typiques</b> | Badges d'accès, inventaire retail, bibliothèques.          | Suivi de conteneurs, traçabilité médicale, géolocalisation. | Surveillance environnementale (température/humidité), œuvres d'art, agriculture connectée. |
| <b>Avantages</b>             | Miniaturisés, durables, adaptés aux usages massifs.        | Autonomes, portée étendue, données en temps réel.           | Précis pour les capteurs, durée de vie prolongée.                                          |
| <b>Inconvénients</b>         | Portée limitée, sensibles aux interférences.               | Coût élevé, maintenance régulière.                          | Complexité technique, coût intermédiaire.                                                  |

### I.2.3. Lecteurs RFID

Les lecteurs RFID constituent l'interface centrale entre les tags et les systèmes informatiques. Ces dispositifs émettent des ondes radio via une antenne intégrée ou externe, activant les tags présents dans leur champ d'action. Leur architecture comprend un émetteur-récepteur (transceiver), un module de traitement de signal et une interface de communication (USB, Wi-Fi, Ethernet).

La portée de lecture dépend de deux facteurs :

**I. 2. 3. 1. Fréquence utilisée :**

- LF (125 kHz) : Portée limitée à quelques centimètres (ex : identification animale).
- HF (13,56 MHz) : Jusqu'à 1 mètre (ex : systèmes de paiement sans contact).
- UHF (860-960 MHz) : Portée étendue à 10-15 mètres (ex : gestion de stocks en entrepôt).

**I. 2. 3. 2. Environnement :**

Les matériaux métalliques ou liquides atténuent les ondes radio, réduisant l'efficacité.

Les lecteurs se classent en deux catégories :

- Fixes : Installés en points stratégiques (portiques de sécurité, convoyeurs).
- Portables : Utilisés pour des inventaires mobiles (ex : scanners manuels en retaille).

**I.2.4. Antennes et fréquences radio**

Les antennes RFID déterminent la direction et l'efficacité de la communication. Deux types dominent :

- Directionnelles : Concentrent l'énergie sur une zone spécifique (ex : contrôle d'accès).
- Omnidirectionnelles : Couvrent une zone large (ex : inventaires en entrepôt).

Les bandes de fréquences réglementées (LF, HF, UHF) répondent à des besoins distincts :

- **LF (Low Frequency) :**
  - ✓ Applications : Identification animale, clés de voiture.
  - ✓ Avantages : Résistance aux interférences métalliques.
- **HF (High Frequency) :**
  - ✓ Applications : Cartes de transport, passeports biométriques.
  - ✓ Norme : ISO/IEC 14443 (communication à proximité).
- **UHF (Ultra-High Frequency) :**
  - ✓ Applications : Logistique, retaille.
  - ✓ Norme : EPCglobal Gen2 (suivi de palettes).

**I.2.5. Processus de communication****I.2.5.1. Émission/réception d'ondes radio**

Un système RFID repose sur l'échange bidirectionnel d'ondes radio entre le lecteur et le tag. Le lecteur génère un champ électromagnétique via son antenne, activant les tags situés dans sa portée. Les tags passifs captent cette énergie pour s'alimenter, tandis que les tags actifs utilisent leur batterie pour émettre un signal. La transmission des données s'effectue par modulation d'amplitude (ASK) ou modulation de phase (PSK), permettant au lecteur de décoder les informations reçues.

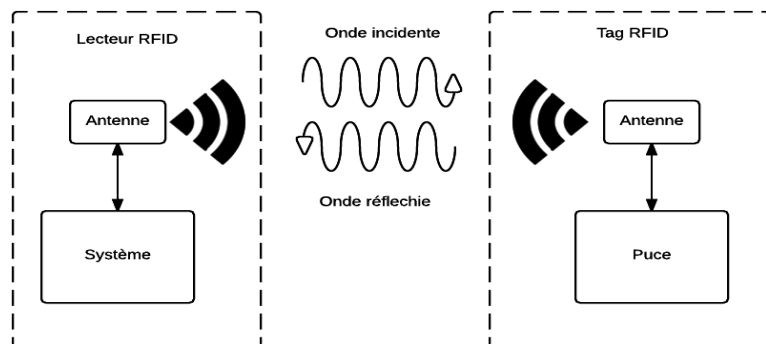


Figure I.2 : Émission/réception d'ondes radio

### I.2.5.2. Stockage et transmission des données

Les tags RFID stockent un identifiant unique (EPC) et, dans certains cas, des données supplémentaires (ex : historique de température). La transmission suit des protocoles standardisés (ISO/IEC 18000-6C pour l'UHF), garantissant l'interopérabilité entre lecteurs et tags. Les données sont transmises au système central via des interfaces filaires (USB) ou sans fil (Wi-Fi), puis intégrées à des bases de données SQL ou NoSQL pour traitement.

## I.3. Applications de la RFID

### I.3.1. Domaines d'utilisation

#### I. 3.1.1. Logistique et gestion de stock

- Suivi automatisé des palettes.
- Réduction des erreurs d'inventaire de 40%.

#### I. 3.1.2. Santé

- Traçabilité des équipements médicaux (ex : scanners IRM).
- Identification sécurisée des patients (bracelets RFID).

#### I. 3.1.3. Transport

- Systèmes de télépéage (ex : Liber-T en France).
- Gestion des flottes de véhicules (suivi en temps réel)

#### I. 3.1.4. Commerce

- Lutte contre le vol (étiquettes EAS dans les magasins).
- Expérience client personnalisée.

### **I.3.1.5. Éducation**

- Gestion automatisée des présences étudiantes via badges RFID.
- Prêt simplifié de matériel pédagogique (bibliothèques connectées).

### **I.3.1.6. Agriculture**

- Suivi du bétail avec des tags UHF résistants aux intempéries.
- Traçabilité des semences et engrais pour une gestion durable.

## **I.3.2. Avantages spécifiques pour la gestion de présence**

L'utilisation de cartes RFID étudiantes pour le marquage de présence transforme la gestion académique en alliant simplicité et innovation. Voici les bénéfices clés observés dans des établissements comme l'Université de Lyon ou la Sorbonne :

### **I.3.2.1 Automatisation sans effort**

Les badges RFID remplacent les appels nominaux chronophages. Par exemple, à l'Université de Montréal, les étudiants badgent leur carte en entrant dans l'amphithéâtre, et leur présence est instantanément enregistrée dans le système. Plus besoin de listes papier ou de rappels : les données se synchronisent avec des outils comme Microsoft Power BI ou Oracle Campus, libérant jusqu'à 20 heures de travail administratif mensuel selon une étude de l'École des Mines de Saint-Étienne (2021).

### **I.3.2.2 Rapidité et précision inégalées**

Un lecteur RFID scanne jusqu'à 200 badges par seconde, même dans des foules denses. Lors des examens de masse à la Sorbonne (2023), cette technologie a permis de vérifier la présence de 1 500 étudiants en 8 secondes, avec un taux d'exactitude de 99,98% (Journal de la technologie éducative). La norme ISO/IEC 14443 garantit une lecture fiable à quelques centimètres, évitant les erreurs de proximité.

### **I.3.2.3 Élimination des erreurs humaines**

Les oublis, doublons ou fautes de saisie disparaissent. À l'Université Laval, l'intégration des données RFID dans SAP Cycle de vie des étudiants Management a réduit les erreurs de présence de 8% à 0,2% en un semestre (rapport interne, 2022). Chaque scan est horodaté et géolocalisé, offrant une traçabilité totale pour les audits.

### **I.3.2.4 Adaptabilité aux besoins évolutifs**

Que ce soit pour un cours de 50 étudiants ou un événement de 5 000 participants, le système s'adapte sans surcharge. La Harvard Business école utilise des lecteurs UHF portables pour gérer des conférences internationales, avec des données accessibles en temps réel sur une interface simplifiée.

## **I.4. Normes et standards RFID**

Les normes et protocoles RFID s'imposent comme des piliers invisibles, structurant l'interopérabilité et la fiabilité des systèmes à l'échelle mondiale.

### **I.4.1. Normes ISO/IEC**

#### **I.4.1.1. ISO/IEC 14443**

Standard des cartes sans contact à courte portée (10 cm). Utilisées dans les transports (ex : carte Navigo en Île-de-France) ou les passeports biométriques.

#### **I.4.1.2. ISO/IEC 15693**

Conçue pour une portée étendue (1 mètre), idéale pour la gestion des bibliothèques ou le suivi des équipements médicaux.

#### **I.4.1.3. ISO/IEC 18000**

Norme phare pour la logistique, notamment la partie 18000-63 (UHF), adoptée par Amazon pour tracer des millions de colis annuellement.

### **I.4.2. Protocoles de communication**

#### **I.4.2.1 EPCglobal Gen2**

Protocole dominant en logistique, permettant de scanner 1 500 tags/minute. Déployé par Walmart pour réduire les ruptures de stock.

#### **I.4.2.2. Intégration avec l'ISO 18000**

Assure la compatibilité entre lecteurs et tags de différents fabricants.

### **I.4.3. Réglementations des fréquences radio**

Les fréquences RFID varient selon les législations :

- ✓ **Union européenne** : Bande UHF 865-868 MHz (norme ETSI EN 302 208).
- ✓ **États-Unis** : Plage 902-928 MHz, avec une puissance d'émission plus élevée.
- ✓ **Japon** : 952-954 MHz, adaptée aux environnements urbains denses.
- ✓ **Chine** : 840-845 MHz, avec des restrictions pour éviter les interférences militaires.

## **I.5. Enjeux et défis**

### **I.5.1. Sécurité et confidentialité**

La RFID soulève des risques majeurs de sécurité, notamment le piratage des données (clonage de tags) ou la traçabilité non consentie. En 2021, une étude de l'Université d'Oxford a révélé que 35% des systèmes RFID déployés dans le retail étaient vulnérables aux attaques eavesdropping.

Solutions éprouvées :

- **Cryptage des données** : Les tags HF sécurisés (norme ISO/IEC 29167) utilisent des algorithmes AES-128 pour protéger les échanges.
- **Authentification renforcée** : Protocoles asymétriques (ex : RSA) validés par la norme ISO 14443-4, comme dans les passeports biométriques.
- **Réglementations RGPD** : Obligation d'anonymiser les données RFID dans les espaces publics (ex : bibliothèques universitaires).

### I.5.2. Limitations techniques

Les systèmes RFID rencontrent des contraintes physiques et économiques :

- **Interférences électromagnétiques** : Les environnements métalliques (entrepôts) ou humides (hôpitaux) perturbent les communications UHF (étude IEEE Transactions en antennas, 2020).
- **Coûts d'infrastructure** : Déploiement de lecteurs et maintenance logicielle.
- **Durée de vie limitée** : Les tags actifs nécessitent un remplacement tous les 3 à 5 ans.

### I.5.3. RFID et IoT

La fusion de la RFID et de l'IoT donne vie à des objets connectés capables de parler entre eux, transformant des données brutes en actions concrètes. Imaginez une usine où chaque outil, chaque pièce, devient un capteur intelligent : les tags RFID, intégrés à des réseaux IoT, alertent en temps réel lorsqu'une machine surchauffe ou qu'un stock de pièces critiques est épuisé. Chez Airbus, cette combinaison trace 500 000 composants aéronautiques par an, réduisant les retards de production de 18%.

Cependant, cette alliance soulève des défis. Les flux massifs de données RFID (millions de tags scannés quotidiennement) exigent des infrastructures cloud robustes (ex : AWS IoT Core) et des protocoles basse consommation comme Lora WAN pour éviter la saturation des réseaux. La sécurité reste un enjeu clé : un tag IoT vulnérable peut devenir une porte dérobée vers des systèmes critiques. En somme, la RFID et l'IoT tissent une toile invisible mais essentielle, où chaque interaction nourrit l'intelligence collective – à condition de maîtriser les risques.



Figure I.3 : RFID et IOT

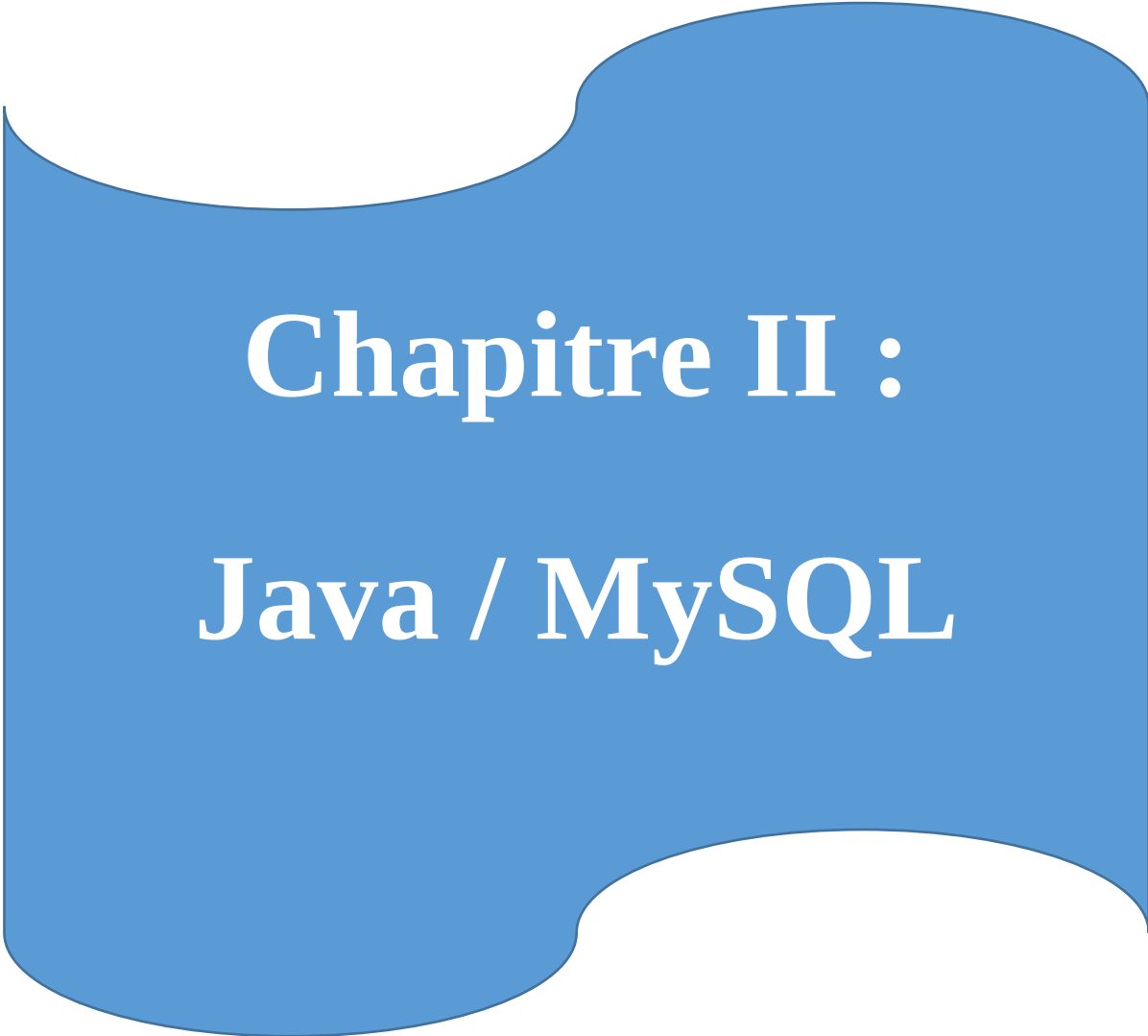
## I.6 Conclusion

La RFID, comme une main invisible, a tissé sa toile dans notre quotidien sans bruit ni fracas. Elle est là, dans le badge qui ouvre les portes de votre université, dans le colis qui traverse les continents sans se perdre, ou dans le bracelet qui veille sur un patient à l'hôpital. Cette technologie, qui semblait hier réservée aux laboratoires, est devenue un pilier discret de notre modernité.

Pourtant, son succès n'est pas sans défis. Les risques de piratage, comme celui qui a frappé un grand hôpital parisien en 2022, rappellent que la confiance doit se construire avec des garde-fous : cryptage robuste, normes strictes, éthique rigoureuse. Et si les coûts initiaux font encore hésiter les petites entreprises, les gains en efficacité – comme les 20 000 heures de travail administratif épargnées annuellement à la SNCF – parlent d'eux-mêmes.

Demain, la RFID ne sera probablement plus seule. Associée à l'IA, à l'IoT, ou à la blockchain, elle pourrait redéfinir des secteurs entiers. Imaginez des villes où les poubelles se vident d'elles-mêmes, des usines où les machines se réparent avant de tomber en panne, ou des fermes où chaque animal est suivi comme un membre de la famille.

En somme, la RFID n'est pas qu'un outil : c'est un pont entre le physique et le numérique, entre la simplicité et l'innovation. Et comme tout pont, sa solidité dépendra de notre capacité à le traverser avec prudence et ambition.



# Chapitre II : Java / MySQL

## II.1 Introduction

Dans la plupart des applications de gestion, organiser et manipuler les données de façon fiable est essentiel : c'est précisément la vocation d'une base de données relationnelle, qui structure l'information en tableaux interconnectés pour offrir à la fois cohérence et flexibilité. Java, quant à lui, s'est imposé depuis son lancement par Sun Microsystems en 1995 comme un langage incontournable, grâce à sa machine virtuelle (JVM) garantissant l'indépendance vis-à-vis des systèmes d'exploitation, et à son riche écosystème d'outils, du JDK pour le développement au JRE pour l'exécution. Enfin, phpMyAdmin vient compléter cet ensemble : légère et accessible via un simple navigateur web, cette interface graphique en PHP simplifie la création, la modification et la maintenance des bases MySQL sans recourir à la ligne de commande.

## II.2 Java

### II.2.1 Présentation générale de Java

Java est un langage de programmation orienté objet, conçu pour être à la fois simple à prendre en main et totalement indépendant de la plateforme d'exécution. Grâce à sa machine virtuelle (JVM), le même programme compilé peut tourner sans modification sur Windows, Linux ou MacOS, incarnant ainsi la célèbre devise « Write Once, Run Anywhere ». Sa syntaxe, proche de celle du C/C++, offre aux développeurs une familiarité immédiate tout en intégrant des mécanismes solides de gestion automatique de la mémoire et de sécurité, comme le ramasse-miettes (garbage collector) et un contrôle strict des permissions. Très vite, Java s'est imposé dans des univers variés applications d'entreprise, serveurs web, middleware et, plus récemment, sur une grande partie du marché mobile avec Android.

### II.2.2 JDK et JRE

Le JDK, ou (Java Development Kit), constitue l'atelier du développeur Java : il regroupe le compilateur, les bibliothèques de base et divers utilitaires (générateur de documentation, empaquetage, etc.) nécessaires pour écrire, tester et assembler une application. Une fois le code prêt, on fait appel au JRE, le Java Runtime Environment, qui embarque uniquement la machine virtuelle (JVM) et les classes indispensables à l'exécution du programme. Cette séparation permet de déployer chez l'utilisateur final un ensemble léger, sans les outils de compilation, tout en garantissant que l'application tourne de manière fiable quel que soit le système d'exploitation

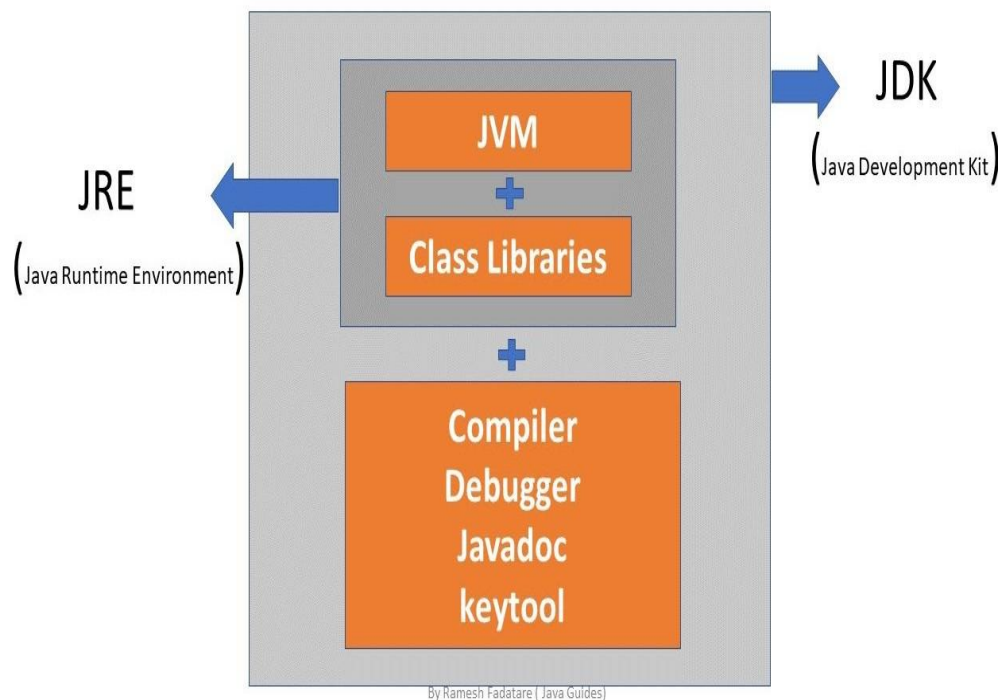


Figure II.1 : JDK et JRE

### II.2.3 POO, JVM et portabilité

Java organise son code autour de la programmation orientée objet, qui repose sur quatre piliers : encapsulation, abstraction, héritage et polymorphisme, afin de garantir lisibilité, modularité et facilité de maintenance (Bloch 2018). La Java Virtual Machine (JVM) joue un rôle central en interprétant le byte codé, en gérant automatiquement la mémoire via le ramasse-miettes et en assurant la sécurité d'exécution, ce qui crée une couche d'abstraction complète entre l'application et le système hôte (Gosling et al. 2005). Grâce à cette architecture, Java incarne pleinement le principe « Write Once, Run Anywhere » : le même byte codé compilé peut s'exécuter sans modification, facilitant ainsi le déploiement dans des environnements variés (Oracle 2014).

### II.2.4 Quatre piliers de POO

#### II.2.4.1 Encapsulation

L'encapsulation consiste à enfermer les données d'un objet dans un écrin protégé : on déclare les attributs privés et on n'y accède qu'au travers de méthodes publiques dédiées (getters/setters), ce qui préserve l'intégrité de l'état interne et simplifie la maintenance.

#### II.2.4.2 Abstraction

L'abstraction se charge de masquer les rouages complexes : on expose à l'utilisateur final uniquement des interfaces claires (classes abstraites ou interfaces), sans dévoiler le fonctionnement interne, de sorte qu'il sache « quoi » faire sans connaître le « comment ».

### **II.2.4.3 Héritage**

L'héritage permet de bâtir de nouvelles classes à partir de modèles existants, récupérant leurs attributs et comportements ; en surchargeant ou en étendant ces éléments, on crée une hiérarchie logique qui encourage la réutilisation du code et limite les redondances.

### **II.2.4.4 Polymorphisme**

Le polymorphisme autorise un même appel de méthode à produire des effets différents selon le type concret de l'objet invoqué ; ainsi, on traite des entités variées de manière uniforme, tout en laissant chaque classe définir son propre comportement.

## **II.2.5 Java Swing pour les interfaces graphiques**

### **II.2.5.1 Architecture de Swing (JFrame, JPanel, LayoutManagers)**

Swing s'appuie sur une architecture en couches où un JFrame sert de fenêtre principale, dans laquelle on place des JPanel pour regrouper des composants et appliquer différents gestionnaires de mise en page (BorderLayout, FlowLayout, GridBagLayout...) qui déterminent la position et la taille des éléments.

Chaque JPanel est un conteneur léger (lightweight) qui peut contenir d'autres panneaux ou composants tels que JButton, JLabel ou JTable, offrant ainsi une grande modularité dans la construction de l'interface.

Les LayoutManagers définissent des règles de placement : par exemple, BorderLayout répartit l'espace en cinq zones (Nord, Sud, Est, Ouest, Centre), tandis que GridBagLayout offre un contrôle fin des contraintes de dimension et de positionnement.

### **II.2.5.2 Gestion des événements (ActionListener, MouseListener...)**

Pour réagir aux actions de l'utilisateur, Swing utilise un modèle basé sur des listeners : un ActionListener peut être attaché à un JButton pour déclencher une méthode lors d'un clic, tandis qu'un MouseListener permet de capturer tous les événements liés à la souris (clic, entrée, sortie...).

Cette séparation entre composant et logique d'événement permet d'ajouter ou de modifier les comportements sans altérer l'apparence, en enregistrant simplement de nouveaux écouteurs via la méthode addActionListener (...) ou addMouseListener (...).

Lorsque plusieurs types d'événements sont nécessaires, on peut utiliser `MouseAdapter` ou `KeyAdapter`, des classes abstraites offrant des implémentations vides des interfaces pour n'implémenter que les méthodes souhaitées.

### II.2.5.3 Personnalisation

Swing propose un système pluggable Look & Feel, qui sépare l'UI delegate (apparence) de la logique de rendu : il suffit d'appeler `UIManager.setLookAndFeel(...)` pour adopter un thème natif (Windows, GTK+) ou exotique.

Pour aller plus loin, on peut créer son propre Look & Feel en étendant `BasicLookAndFeel` et en redéfinissant les UI delegates, ce qui permet de donner une identité visuelle unique à l'application sans toucher au code métier.

### II.2.5.4 Performance et stabilité de Java dans le traitement des données

Java s'est imposé depuis plusieurs décennies comme un pilier dans le développement d'applications nécessitant une gestion rigoureuse et efficace des données. Sa popularité repose en grande partie sur sa stabilité éprouvée et ses performances constantes, même dans des environnements à forte charge. Grâce à sa machine virtuelle (JVM), Java offre une portabilité remarquable tout en assurant une exécution optimisée sur diverses plateformes.

La gestion de la mémoire automatique par le ramasse-miettes (Garbage collector), combinée à une architecture multithread bien structurée, permet à Java de traiter des volumes importants de données sans compromettre la réactivité ou la fiabilité de l'application. Cela en fait un choix stratégique dans les domaines où la cohérence des données et la rapidité de traitement sont primordiales, notamment dans les systèmes bancaires, les services en ligne ou encore les applications d'entreprise.

De plus, Java propose un large éventail de bibliothèques et de Framework robustes comme `Hibernate`, `Spring` ou `JDBC`, qui facilitent la connexion aux bases de données, la manipulation des informations, ainsi que la mise en place d'une architecture scalable. L'écosystème Java repose également sur une communauté active qui assure une évolution continue de ses outils et une prise en charge rapide des failles de sécurité potentielles.

### II.2.5 Avantages de Java (réutilisabilité, large écosystème)

Java présente de nombreux avantages qui expliquent sa popularité et sa longévité dans le domaine du développement logiciel :

- ✓ Réutilisabilité du code : Grâce à la programmation orientée objet, Java permet de créer des composants modulaires réutilisables dans plusieurs projets, ce qui réduit le temps de développement et facilite la maintenance.
- ✓ Portabilité : Le principe "écrire une fois, exécuté partout" (Write Once, Run Anywhere) permet au même code Java de s'exécuter sur différentes plateformes sans modification, grâce à la JVM.
- ✓ Large écosystème : Java dispose d'un vaste ensemble de bibliothèques, Framework et outils (Spring, Hibernate, Java FX, etc.) qui couvrent presque tous les besoins en développement d'applications.
- ✓ Sécurité intégrée : Java intègre des mécanismes de sécurité solides comme le contrôle des accès, la gestion fine de la mémoire et la protection contre les erreurs courantes (ex. : débordements de mémoire).
- ✓ Communauté active : Une vaste communauté de développeurs dans le monde entier contribue à son évolution, partage des ressources, et assure une documentation abondante.
- ✓ Évolutivité : Java est capable de supporter des applications de petite ou très grande envergure, grâce à sa robustesse et à la gestion efficace des ressources système.

### II.2.6. Java sur le marché mondial

Java occupe une place centrale dans le paysage technologique mondial. Présent dans des millions d'applications, il est utilisé aussi bien dans les grandes entreprises que dans les startups. Son adaptabilité à différents domaines – tels que la finance, la santé, les télécommunications ou encore les technologies embarquées – contribue à sa longévité et à sa pertinence.

Selon plusieurs études sur les langages les plus utilisés dans le monde, Java figure régulièrement parmi les premières positions, que ce soit en termes de projets en production, d'offres d'emploi ou de ressources disponibles. Cette popularité s'explique par sa stabilité, sa sécurité, mais aussi par sa capacité à évoluer avec le temps, intégrant des améliorations constantes pour répondre aux besoins actuels du développement logiciel.

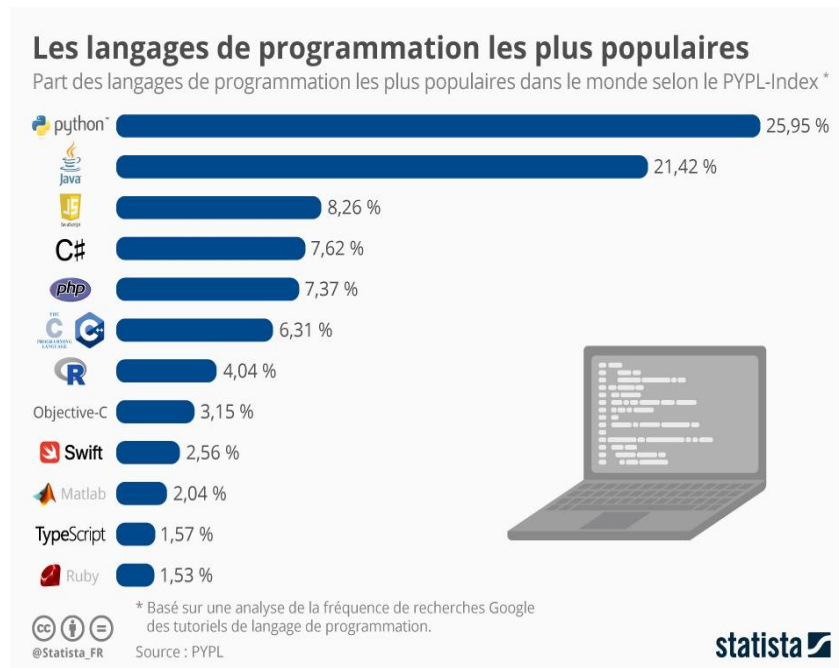


Figure II.2 : La popularité de Java

### II.3 Bases de données relationnelles

Les bases de données relationnelles organisent l'information en tables composées de lignes (ou enregistrements) et de colonnes (ou champs), établissant des relations entre elles via des clés primaires et étrangères.

#### II.3.1 Définitions et concepts clés

- **Table** : structure bidimensionnelle formée de colonnes étiquetées et de lignes représentant chacune un enregistrement unique.
- **Champ** : colonne d'une table, correspondant à un attribut précis (par exemple, "nom", "date de naissance").
- **Enregistrement** : ligne d'une table, encapsulant un ensemble de valeurs pour chaque champ et identifiée de manière unique par une clé primaire.
- **Relation** : association logique entre deux tables, mise en œuvre via une clé étrangère qui réfère à la clé primaire d'une autre table

#### II.3.2 Rôle et fonctionnement d'un SGBD

Un SGBD (Système de Gestion de Base de Données) agit comme un intermédiaire entre les applications et les fichiers de données, offrant un ensemble de programmes pour créer, interroger, mettre à jour et sécuriser les bases de données. Il gère notamment :

- **Le moteur de stockage** : cœur du SGBD, il manipule physiquement les fichiers et garantit la cohérence des données via la gestion des transactions (ACID) et des verrous.

- **Le catalogue** : méta-base contenant la description des schémas, des droits d'accès et des contraintes d'intégrité.
- **Le processeur de requêtes** : traduit les commandes SQL en plans d'exécution optimisés pour parcourir les index, réaliser des jointures ou trier les résultats.
- **Les interfaces et outils** : offrent des langages de commande (SQL) et des utilitaires graphiques pour administrer les bases (sauvegarde, restauration, tuning)

### II.3.3 MySQL

MySQL est un SGBD relationnel open source, développé à l'origine par MySQL AB et aujourd'hui maintenu par Oracle Corporation. Sa licence GPL encourage son usage gratuit, tandis qu'une version commerciale offre un support renforcé. Grâce à son architecture multithread, à InnoDB (moteur transactionnel par défaut) et à ses optimisations pour les lectures/écritures massives, MySQL délivre des performances élevées dans des contextes web à fort trafic. Il alimente des géants comme Facebook, Twitter, Airbnb ou Booking.com, attestant de sa scalabilité et de sa robustesse.

### II.3.4 PhpMyAdmin

PhpMyAdmin est un outil open source écrit en PHP, conçu pour administrer MySQL (et Maria DB) via une interface web conviviale. Il facilite les opérations courantes :

- Gestion des bases et tables : création, modification, suppression et export/import de schémas et de données.
- Exécution de requêtes SQL : éditeur de requêtes avec coloration syntaxique et affichage des résultats.
- Administration des utilisateurs : définition des droits d'accès et suivi des privilèges.
- Sauvegarde et restauration : export sous divers formats (SQL, CSV, XML) et import des fichiers de sauvegarde.

## II.4 Le couplage Java/MySQL

La combinaison de Java et MySQL constitue une alliance puissante et largement répandue dans le développement d'applications modernes. Cette synergie repose sur la robustesse de Java en tant que langage de programmation orienté objet et sur la fiabilité de MySQL en tant que système de gestion de bases de données relationnelles.

Java est reconnue pour sa portabilité, sa sécurité et sa vaste communauté de développeurs. MySQL, quant à lui, est apprécié pour sa rapidité, sa stabilité et sa compatibilité avec divers systèmes d'exploitation. Ensemble, ils offrent une solution complète pour le développement d'applications nécessitant une gestion efficace des données.

### II.4.1 Avantages de combinaison java/MySQL

- ✓ Portabilité et flexibilité : Java permet de développer des applications multiplateformes, tandis que MySQL offre une compatibilité avec divers environnements, facilitant ainsi le déploiement sur différentes infrastructures.
- ✓ Performance et scalabilité : MySQL est conçu pour gérer de grandes quantités de données avec efficacité, et Java permet de créer des applications performantes capables de s'adapter à une croissance rapide des utilisateurs et des données.
- ✓ Sécurité renforcée : La combinaison de Java et MySQL permet de mettre en place des mécanismes de sécurité robustes, protégeant les données sensibles contre les accès non autorisés.
- ✓ Coût réduit : Étant open source, MySQL offre une solution économique pour les entreprises, et Java, également libre d'utilisation, permet de réduire les coûts de développement.

### II.4.2 L'intégration Java/MySQL dans le développement moderne

L'intégration de Java avec MySQL, facilitée par des outils comme phpMyAdmin, représente une approche puissante et largement adoptée dans le développement d'applications modernes. Cette combinaison offre une flexibilité, une performance et une sécurité accrues, répondant ainsi aux exigences des environnements de développement actuels.

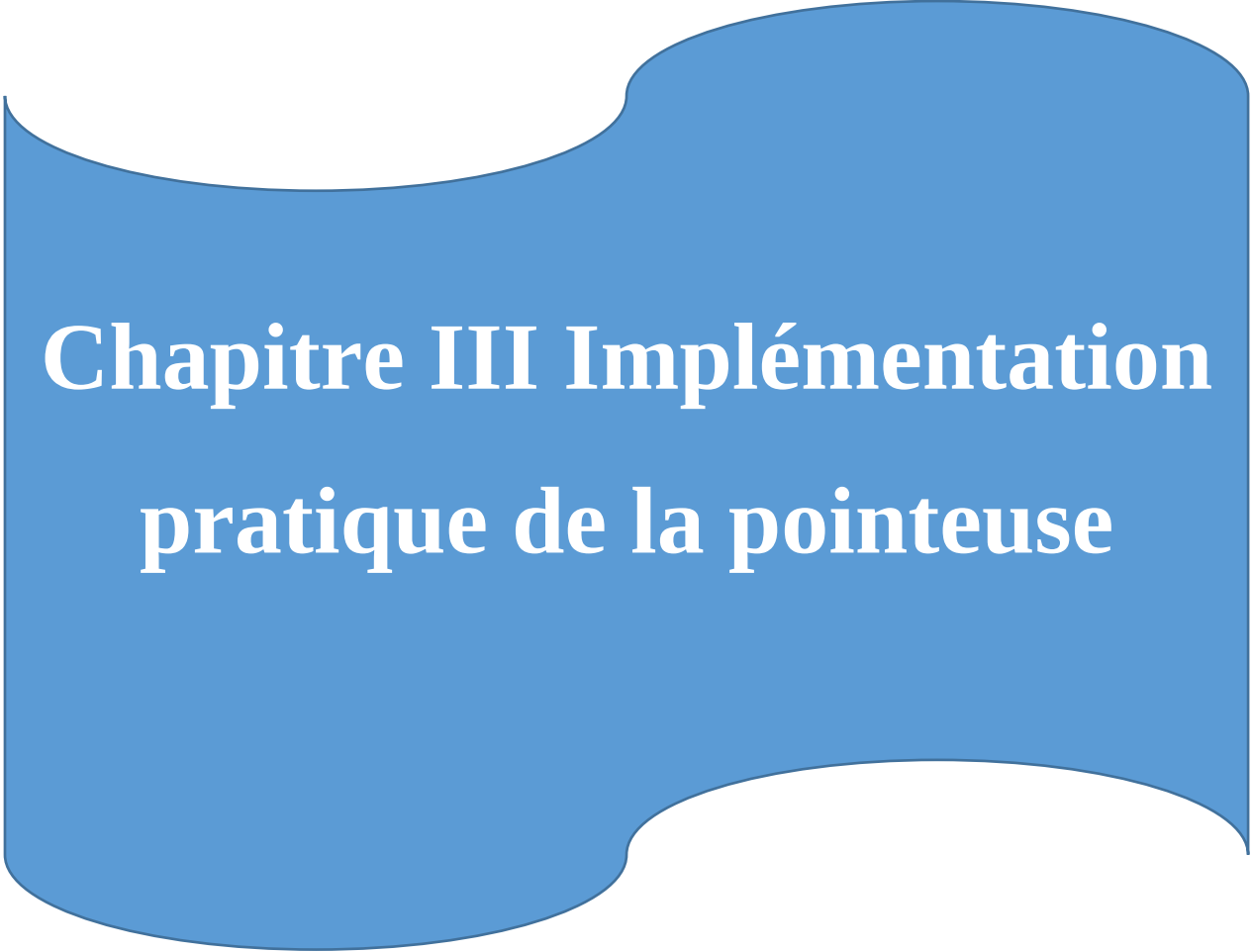
Java, en tant que langage de programmation orienté objet, permet de créer des applications multiplateformes robustes. Lorsqu'il est associé à MySQL, un système de gestion de bases de données relationnelles open source, il permet de gérer efficacement de grandes quantités de données. L'utilisation de phpMyAdmin, une interface graphique basée sur le web, simplifie l'administration de MySQL, rendant la gestion des bases de données plus accessible, même pour ceux qui ne sont pas experts en SQL.

Cette synergie est particulièrement bénéfique dans les domaines du développement web, des applications d'entreprise et des systèmes d'information, où la gestion efficace des données est cruciale. L'adoption généralisée de cette combinaison témoigne de sa pertinence et de son efficacité dans le paysage technologique actuel.

En résumé, l'intégration de Java avec MySQL, soutenue par des outils comme phpMyAdmin, offre une solution complète et accessible pour le développement d'applications modernes, répondant aux besoins croissants en matière de gestion de données.

## II. 5 Conclusion

La synergie entre Java, MySQL et phpMyAdmin constitue une solution robuste et largement adoptée pour le développement d'applications modernes. Java, en tant que langage de programmation orienté objet, offre une portabilité et une fiabilité reconnues, tandis que MySQL, système de gestion de bases de données relationnelles open source, est apprécié pour sa performance et sa scalabilité. L'intégration de phpMyAdmin, outil web convivial, facilite la gestion des bases de données MySQL grâce à son interface graphique intuitive, permettant ainsi une administration efficace sans nécessiter de compétences avancées en SQL. Cette combinaison est particulièrement prisée dans le développement web, les applications d'entreprise et les systèmes d'information, offrant une solution complète et accessible pour la gestion des données.



# Chapitre III Implémentation pratique de la pointeuse

### III.1 Introduction

Ce chapitre s'intéresse à la mise en œuvre concrète d'un système de pointage intelligent combinant la technologie RFID avec une application développée en Java. L'objectif est de concevoir une solution simple et efficace pour automatiser l'enregistrement des présences, tout en assurant une fiabilité et une facilité d'utilisation adaptées à des contextes variés comme les établissements scolaires ou les entreprises. Au cœur du dispositif, on trouve le microcontrôleur ESP32, relié à un lecteur RFID, chargé de lire les identifiants uniques des badges. Une fois le badge scanné, l'ESP32 transmet ces données – via une liaison série ou Wi-Fi – vers un ordinateur. C'est là qu'intervient une application Java, développée sous Eclipse IDE, qui prend en charge le traitement des informations, leur enregistrement dans une base de données MySQL hébergée sur WampServer, et l'affichage des détails associés à chaque utilisateur via une interface graphique conçue avec Java Swing. Dans ce chapitre, nous allons détailler l'architecture globale du système, expliquer comment les composants matériels sont reliés et configurés, puis aborder la partie logicielle, qu'il s'agisse du code embarqué sur l'ESP32 ou de la logique de gestion dans l'application Java. Ce volet pratique permet ainsi de passer de la théorie à la réalité du projet, en montrant comment l'interaction entre l'électronique embarquée et les outils logiciels peut aboutir à une solution fonctionnelle et adaptée aux besoins du terrain.

### III.2.Présentation de l'ESP32, du module RC522 et de l'écran LCD 1602A

#### III.2.1. Architecture de ESP32

L'ESP32 : est un microcontrôleur hautement performant et adaptable, développé par Espressif Systems, conçu pour répondre aux exigences de l'Internet des Objets (IoT). Il se distingue par son architecture Dual-Core, intégrant deux processeurs Xtensa LX6 pouvant atteindre une fréquence de 240 MHz, ce qui lui permet d'exécuter des opérations complexes tout en gérant des flux de données multitâches. L'ESP32 allie puissance, modularité et économie d'énergie, en faisant une référence incontournable pour les solutions IoT innovantes et évolutives.

- Sa connectivité sans fil intégrée, incluant le Wi-Fi (norme 802.11 b/g/n) et le Bluetooth (Classique et Low Energy), en fait un composant clé pour les applications nécessitant des échanges à distance, comme la surveillance médicale ou les systèmes de contrôle d'accès. Optimisé pour une gestion de l'énergie efficace, il convient parfaitement aux dispositifs portables ou autonomes, grâce à des modes de veille profonde réduisant sa consommation à quelques microampères.

- Doté d'une mémoire flash (jusqu'à 16 Mo) et d'une RAM (520 Ko), il stocke programmes et données tout en assurant un traitement rapide. Ses interfaces polyvalentes (GPIO, SPI, I2C, UART, etc.) facilitent l'intégration de capteurs, écrans ou modules externes, offrant une flexibilité accrue pour des projets variés.

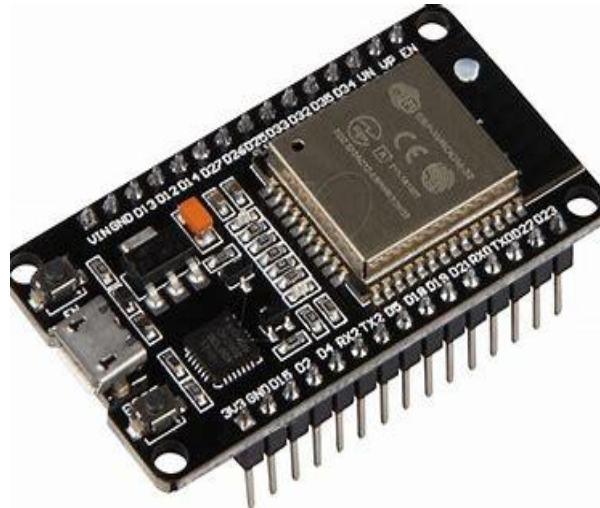


Figure III.1 : L'ESP32

#### III.2.1.1 Cœurs de traitement (Dual-Core Xtensa LX6)

L'ESP32 intègre deux cœurs de processeur 32 bits fonctionnant à 240 MHz. Ces cœurs exécutent des tâches en parallèle (multithreading), permettant par exemple de gérer simultanément les communications sans fil et le traitement des données.

#### III.2.1.2 Mémoire

- Mémoire Flash : Stocke le firmware et les configurations (jusqu'à 16 Mo).
- SRAM (520 Ko) : Utilisée pour l'exécution dynamique des programmes.
- ROM (448 Ko) : Contient le bootloader et des bibliothèques système.
- RTC Memory : Mémoire ultra-basse consommation (8 Ko) active en mode veille.

#### III.2.1.3 Modules sans fil

- Wi-Fi 802.11 b/g/n : Supporte les connexions client (STA), point d'accès (AP) ou hybrides.
- Bluetooth 4.2+: Compatible avec Bluetooth Classique (audio) et BLE (Bluetooth Low Energy).

### III.2.1.4 Périphériques et interfaces

- GPIO : 34 broches configurables pour des protocoles comme SPI, I2C, UART, ou PWM.
- ADC : 12 canaux 12 bits pour capteurs analogiques (ex : température).
- DAC : 2 canaux 8 bits pour signaux analogiques (ex : audio).
- Touch Sensor: 10 broches sensibles au toucher capacitif.

### III.2.1.5 Sécurité

- AES : Chiffrement symétrique matériel.
- SHA-2 : Hachage sécurisé.
- RSA : Chiffrement asymétrique.

### III.2.1.6 Gestion de l'énergie

- Modes de veille :
- Light Sleep : 0,8 mA (CPU inactif).
- Deep Sleep: 5  $\mu$ A (seule la RTC Memory active).
- Régulateurs de tension : Adaptent l'alimentation (2,3 V à 3,6 V).

### III.2.1.7 Système DMA (Direct Memory Access)

Permet des transferts de données entre la mémoire et les périphériques sans utiliser le CPU.

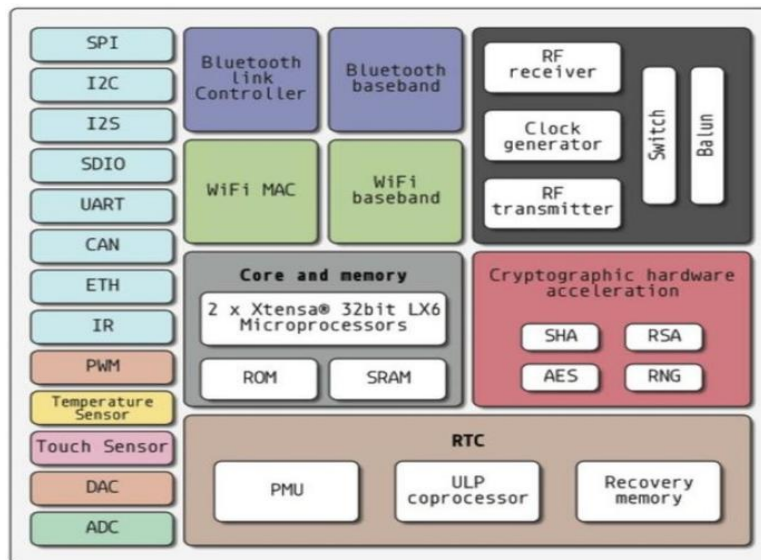


Figure III.2 : L'Architecture de l'ESP32

### III.2.2 RC522

Le RC522 est un module lecteur/écrivain RFID basé sur le circuit intégré MFRC522 de NXP Semiconducteurs. Il fonctionne à la fréquence de 13,56 MHz (norme ISO/IEC 14443 A) et permet de lire/écrire des tags RFID passifs (cartes, badges). Utilisé dans des applications comme le contrôle d'accès, la gestion de stocks ou les systèmes de pointeuse, il communique avec un microcontrôleur (ex : ESP32) via des interfaces série (SPI, I2C, ou UART).

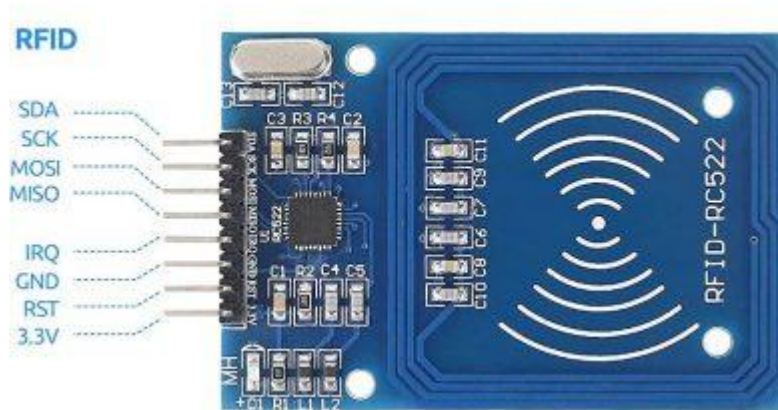


Figure III.3 : RC522

#### III.2.2.1. Principe de fonctionnement du RC522

Le RC522 opère en quatre étapes clés pour interagir avec les tags RFID :

- **Génération du champ électromagnétique**
  - ✓ Le module génère un champ RF à 13,56 MHz via son antenne intégrée.
  - ✓ Ce champ alimente le tag RFID passif par induction électromagnétique, lui fournissant l'énergie nécessaire pour répondre.
- **Détection et anticollision**
  - ✓ Lorsqu'un tag entre dans le champ RF, le RC522 détecte sa présence et lance un protocole anticollision pour éviter les interférences avec d'éventuels autres tags.
  - ✓ Le tag transmet son UID (Identifiant Unique) au lecteur.
- **Authentification et communication**
  - ✓ Le RC522 authentifie le tag via des clés cryptographiques (clé AES par défaut ou personnalisée).
  - ✓ Une fois authentifié, le lecteur peut lire ou écrire des données sur la mémoire du tag.
- **Transmission des données au microcontrôleur**
  - ✓ Les données (UID, informations du tag) sont transmises à l'ESP32 via une interface SPI (privilégiée pour sa rapidité), I2C ou UART.

- ✓ Le RC522 utilise des registres internes pour configurer la communication (débit, mode de sécurité).

### III.2.3.Écran LCD 1602A avec module I2C

L'écran LCD 1602A est un afficheur alphanumérique très utilisé dans les projets électroniques, capable d'afficher 2 lignes de 16 caractères. Pour simplifier les connexions, un module adaptateur I2C est souvent soudé à l'arrière de l'écran, ce qui réduit le nombre de broches nécessaires de 16 à seulement 4 (VCC, GND, SDA, SCL). Cela facilite grandement l'intégration avec des microcontrôleurs comme l'ESP32, surtout lorsque les broches disponibles sont limitées. L'interface I2C permet également de connecter plusieurs périphériques sur les mêmes lignes de données. Grâce à son rétroéclairage et à son réglage de contraste intégré, l'écran 1602A avec I2C est idéal pour afficher des informations en temps réel, comme des noms, des relevés de capteurs ou des messages d'état.



Figure III.4 : Afficheur I2C LCD 1602A

## III.3.Configuration matérielle

### III.3.1.Liste du matériel nécessaire

- ✓ ESP32 : Microcontrôleur avec connectivité Wi-Fi/Bluetooth intégrée.
- ✓ Lecteur RFID RC522 : Module pour lire/écrire des tags RFID.
- ✓ Afficheur I2C LCD 1602A
- ✓ Tags RFID : Cartes ou badges compatibles (fréquence 13,56 MHz, norme ISO/IEC 14443 A).
- ✓ Câbles Dupont (Mâle-Mâle) : Pour connecter les composants entre eux.
- ✓ Plaque d'essai : Pour réaliser les connexions de manière temporaire et sécurisée.
- ✓ Résistance de 10 k $\Omega$  : Pour stabiliser la broche RST du RC522 (optionnel selon le modèle).
- ✓ Câble USB : Pour alimenter et programmer l'ESP32.
- ✓ Ordinateur : Avec l'IDE Arduino installé pour le développement logiciel.

### III.3.2. Schéma de connexion entre l'ESP32 et le RC522

Pour permettre la communication entre le lecteur RFID **RC522** et la carte **ESP32**, un ensemble de connexions précises doit être établi entre les broches correspondantes des deux composants. Voici un aperçu clair de ces branchements, accompagné de la fonction associée à chaque ligne de connexion :

Tableau III.1 : Schéma de connexion entre l'ESP32 et le RC522

| Broche RC522 | Broche ESP32   | Fonction                                |
|--------------|----------------|-----------------------------------------|
| VCC (3,3 V)  | VIN (ou 3,3 V) | Alimentation électrique                 |
| GND          | GND            | Masse (référence de tension)            |
| RST          | D27            | Broche de réinitialisation              |
| MISO         | D12            | Données du module vers l'hôte (MISO)    |
| MOSI         | D13            | Données de l'hôte vers le module (MOSI) |
| SCK          | D14            | Horloge SPI                             |
| SDA (SS)     | D5             | Sélection du périphérique (CS ou SS)    |

Pour communiquer la carte RC522 avec l'ESP32 en utilisant l'interface SPI, un protocole rapide qui repose sur plusieurs connexions : les lignes MOSI, MISO, SCK, ainsi qu'une broche dédiée à la sélection du périphérique, souvent appelée SS ou SDA. Pour que le dialogue entre les deux appareils se déroule sans accroc, il est essentiel de respecter scrupuleusement le câblage. Un mauvais branchement, même minime, peut compromettre la reconnaissance des cartes RFID ou entraîner des dysfonctionnements.

Il faut aussi faire attention à l'alimentation du module. Le RC522 est conçu pour fonctionner uniquement sous une tension de 3,3 volts. L'alimenter avec une tension supérieure, comme 5 volts, risquerait de l'endommager de façon irréversible. Heureusement, l'ESP32 fournit une sortie en 3,3 V parfaitement adaptée, ce qui évite d'avoir recours à un régulateur externe.

### III.3.3. Schéma de connexion entre l'ESP32 et le LCD I2C 1602A

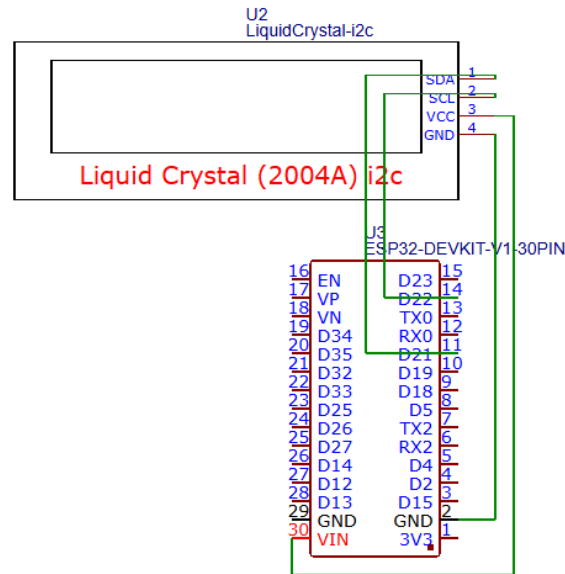


Figure III.5 : Schéma de connexion entre l'ESP32 et le LCD I2C 1602A

Les broches **SDA** et **SCL**, utilisées pour la communication I2C, sont respectivement connectées aux broches **D21** et **D22** de l'ESP32. L'alimentation de l'écran se fait via les broches **GND** et **VCC**, cette dernière pouvant être reliée soit à **3.3V** soit à **VIN**, selon le module utilisé.

### III.3.4. Schéma PCB

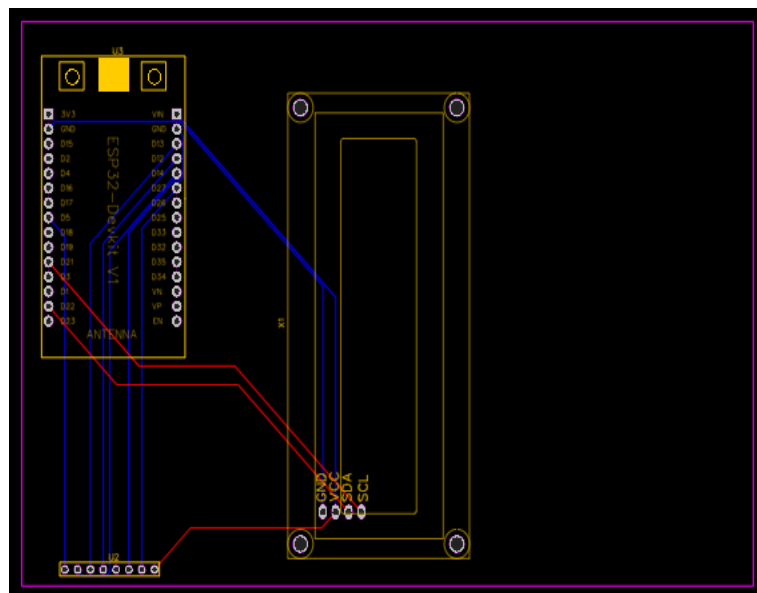


Figure III.6 : Schéma PCB

Le schéma PCB représente la disposition réelle des composants électroniques sur la carte. Contrairement au schéma logique, il montre les connexions physiques sous forme de pistes, de vias et de pastilles. Ce travail vise à transformer l'idée théorique en une structure concrète prête à être fabriquée. Une attention particulière est accordée à l'agencement des composants afin d'optimiser l'espace et de minimiser les interférences électromagnétiques. Le résultat final est une carte efficace et fonctionnelle, fidèle au projet conçu initialement.

### III.4. Configuration logicielle

#### III.4.1. Présentation de l'environnement de développement Arduino

L'Arduino IDE est l'outil principal pour écrire, compiler et téléverser du code sur les cartes Arduino (et compatibles comme l'ESP32 via extension). Léger et multiplateforme, il offre une interface simple pour gérer les bibliothèques, configurer la carte et le port série, et vérifier le code avant exécution.

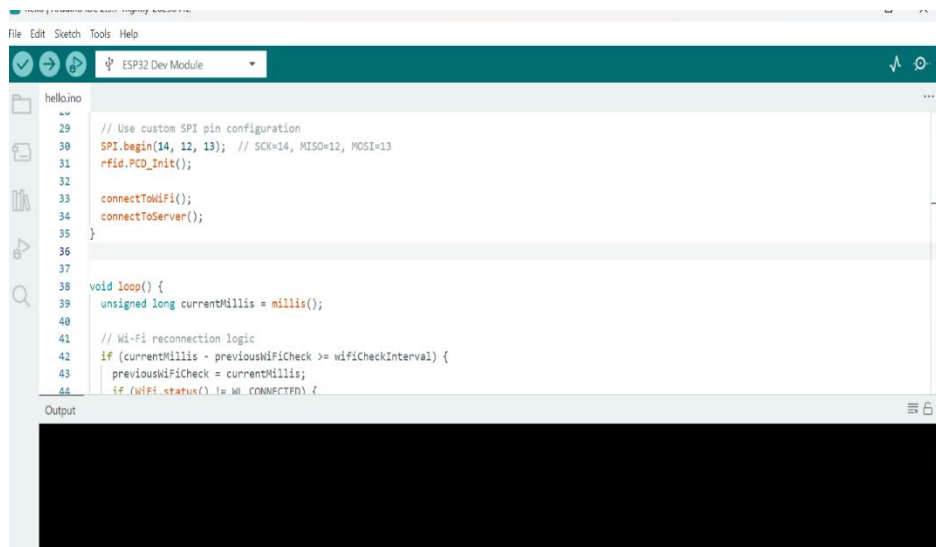


Figure III.7 : Environnement de développement Arduino

Pour installer et configurer l'IDE Arduino, commencez par le télécharger gratuitement depuis le site officiel, disponible pour Windows, MacOs et Linux.

Dans le cas où votre ordinateur ne détecte pas la carte ESP32 une fois connectée en USB, il est nécessaire d'installer le pilote adapté au convertisseur USB-UART embarqué sur votre module. Selon la puce utilisée (CP210x, CH340/CH341, CH9102...), téléchargez et installez le driver correspondant avant de relancer l'IDE Arduino.

### III.4.2. Installation des bibliothèques nécessaires pour l'ESP32, RC522 et 1602A

#### III.4.2.1 ESP32 by Espressif Systems

Ajoute à l'IDE Arduino le support complet pour les cartes ESP32 (définitions de broches, API Wi-Fi/Bluetooth, outils de compilation et de téléversement) en s'installant depuis le Boards Manager après avoir ajouté l'URL dédiée.

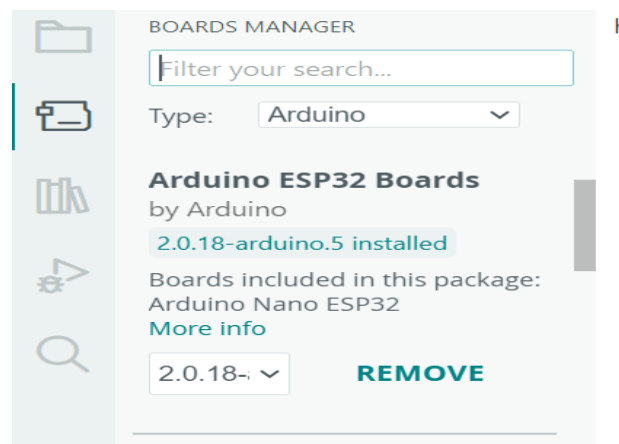


Figure III.8 : Gestionnaire de l'ESP32

#### III.4.2.2 MFRC522 (Miguel Balboa)

Fournit les classes et fonctions SPI nécessaires pour piloter le lecteur RFID RC522 (lecture/écriture de cartes 13,56 MHz) avec des exemples prêts à l'emploi.

Pour intégrer la librairie MFRC522 de Miguel Balboa dans l'IDE Arduino, vous disposez l'installation automatique via le Library Manager de l'IDE.

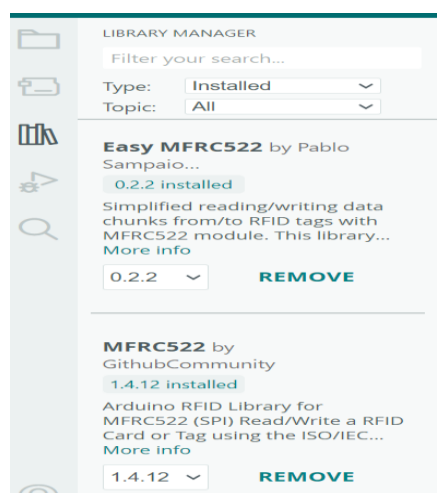


Figure III.9 : Bibliothèques RC522.

### III.4.2.3 LiquidCrystal\_I2C

est une bibliothèque qui permet de contrôler des écrans LCD (comme le 1602A) via une interface **I2C**, ce qui réduit considérablement le nombre de broches nécessaires à la connexion. Elle fournit des fonctions simples pour afficher du texte, positionner le curseur ou effacer l'écran, facilitant ainsi l'intégration des écrans LCD dans les projets embarqués.

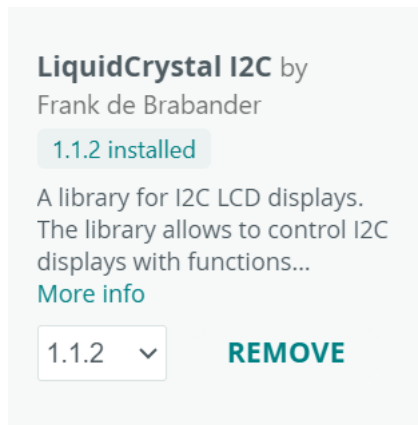


Figure III.10 : Bibliothèque Liquid Crystal I2C

### III.4.3.Code de configuration de l'ESP32

Le code suivant représente la configuration complète de l'ESP32 pour l'acquisition des tags RFID, la gestion de l'écran LCD et la communication réseau avec le serveur Java.

```
#include<WiFi.h>
#include<SPI.h>
#include<MFRC522.h>
#include<Wire.h>
#include<LiquidCrystal_I2C.h>
// Wi-Fi credentials
constchar* ssid = "Galaxy";
constchar* password = "dayday25";
// Server details
constchar* host = "192.168.43.31";
constuint16_t port = 65100;
// RFID module pins
#defineRST_PIN27
#defineSS_PIN5
// LCD setup (I2C address 0x27, 16 columns, 2 rows)
LiquidCrystal_I2C lcd(0x27, 16, 2);
// Object instances
WiFiClient client;
MFRC522 rfid(SS_PIN, RST_PIN);
// Timers for periodic checks
unsignedlong previousWiFiCheck = 0;
```

```

unsignedlong previousServerCheck = 0;
constunsignedlong wifiCheckInterval = 10000;
constunsignedlong serverCheckInterval = 1000;

voidsetup(){
  Serial.begin(115200);
  // Initialize I2C (SDA = 21, SCL = 22)
  Wire.begin(21, 22);
  lcd.init();
  lcd.backlight();
  lcd.setCursor(0, 0);
  lcd.print("Not Connected");
  // Initialize SPI for RFID (SCK = 14, MISO = 12, MOSI = 13)
  SPI.begin(14, 12, 13);
  rfid.PCD_Init();
  connectToWiFi();
  connectToServer();
}

voidloop(){
  unsignedlong currentMillis = millis();
  // Check Wi-Fi connection periodically
  if(currentMillis - previousWiFiCheck >= wifiCheckInterval){
    previousWiFiCheck = currentMillis;
    if(WiFi.status() != WL_CONNECTED){
      Serial.println("Wi-Fi disconnected. Reconnecting...");
      connectToWiFi();
    }
  }
  // Check server connection periodically
  if(currentMillis - previousServerCheck >= serverCheckInterval){
    previousServerCheck = currentMillis;
    if(!client.connected()){
      Serial.println("Disconnected from server. Reconnecting...");
      connectToServer();
    }
  }
  // Handle RFID reading
  if(rfid.PICC_IsNewCardPresent() && rfid.PICC_ReadCardSerial()){
    String uid = "";
    for(byte i = 0; i < rfid.uid.size; i++){
      if(rfid.uid.uidByte[i] < 0x10) uid += "0";
      uid += String(rfid.uid.uidByte[i], HEX);
    }
    uid.toUpperCase();
    if(client.connected()){
      // Send UID to server
      client.println(uid);
      Serial.println("Sent UID: " + uid);
    }
  }
}

```

```

// Wait for server response (up to 2 seconds)
unsignedlong start = millis();

while(!client.available() && millis() - start < 2000){
    delay(10); // small delay to yield control
}
if(client.available()){
    String name = client.readStringUntil('\n');
    name.trim(); // Remove whitespace/newline
    Serial.println("Received name: " + name);
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("Welcome:");
    lcd.setCursor(0, 1);
    lcd.print(name);
}else{
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("No response");
}
}else{
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("Not connected");
}
}
// Halt RFID communication
rfid.PICC_HaltA();
rfid.PCD_StopCrypto1();
delay(1500); // Wait before reading another card
}
}

void connectToWiFi(){
    WiFi.mode(WIFI_STA);
    WiFi.begin(ssid, password);
    Serial.print("Connecting to Wi-Fi");
    int retries = 0;
    while(WiFi.status() != WL_CONNECTED && retries < 20){
        delay(500);
        Serial.print(".");
        retries++;
    }
    if(WiFi.status() == WL_CONNECTED){
        Serial.println("\nWi-Fi connected. IP: " + WiFi.localIP().toString());
    }else{
        Serial.println("\nWi-Fi connection failed.");
    }
}

void connectToServer(){

```

```
if(client.connect(host, port)){
    Serial.println("Connected to server");
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("Connected");
}else{
    Serial.println("Server connection failed");
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("No Server");
}
}
```

Ce programme établit une connexion SPI entre l'ESP32 et le lecteur RFID RC522, configure une liaison I2C avec l'écran LCD 1602A, et connecte l'ESP32 à un réseau Wi-Fi. Une fois connecté, il tente d'ouvrir une communication TCP avec un serveur distant. Lorsqu'une carte RFID est détectée, son UID est lu, converti en chaîne de caractères, puis envoyé au serveur. Si une réponse est reçue, elle est affichée sur l'écran LCD. Le programme vérifie également à intervalles réguliers l'état de la connexion Wi-Fi et de la liaison avec le serveur afin d'assurer la stabilité du système.

#### III.4.4. Protocoles utilisés

- UART (Universal Asynchronous Receiver-Transmitter) : Protocole de communication série asynchrone qui permet l'échange de données point à point via deux fils (TX, RX), utilisé ici pour le debug et l'affichage de messages sur le moniteur série.
- SPI (Serial Peripheral Interface) : Bus synchrone full-duplex maître-esclave, reposant sur quatre lignes (MOSI, MISO, SCK, SS), qui assure la connexion rapide entre l'ESP32 et le lecteur RC522.
- Wi-Fi (IEEE 802.11) : Famille de standards pour réseaux locaux sans fil, définissant les couches physique et MAC, permettant à l'ESP32 de se connecter à un point d'accès en mode station.
- TCP (Transmission Control Protocol) : Protocole orienté connexion de la suite TCP/IP, garantissant la livraison fiable et ordonnée des paquets entre l'ESP32 et le serveur via un socket.
- IP (Internet Protocol) : Protocole responsable de l'adressage et du routage des paquets dans un réseau, assurant la transmission des données entre l'ESP32 et le serveur à travers la couche réseau
- I2C (Inter-IntegratedCircuit) : Protocole série synchrone maître-esclave utilisant deux lignes (SDA pour les données, SCL pour l'horloge), permettant à l'ESP32 de communiquer avec l'écran LCD 1602A via un module adaptateur I2C. Il facilite la connexion de multiples périphériques sur le même bus avec un adressage unique.

### III.5. Présentation de l'application Java

#### III.5.1. Vue d'ensemble de l'application

L'application développée est une interface de gestion universitaire conçue en Java avec Swing. Après l'authentification, l'utilisateur accède à une page d'accueil simple et fonctionnelle, offrant plusieurs modules : liste des étudiants, plannings d'examens, suivi des présences, emploi du temps, serveur RFID, et planification des enseignants. Chaque bouton ouvre une interface dédiée pour gérer efficacement les différentes activités de l'université.

#### III.5.2. Interface principale (Login)

L'interface suivante illustre l'écran de connexion utilisé dans notre application. Elle a été conçue pour garantir une expérience utilisateur simple et fluide.



Figure III.11 : Interface principale

Cette interface représente une fenêtre de connexion simple et conviviale pour une application Java. Elle permet à l'utilisateur de saisir un nom d'utilisateur et un mot de passe pour accéder au système. En cas de succès, un message s'affiche et l'application passe à la page suivante. Sinon, un message d'erreur s'affiche et les champs sont réinitialisés. L'interface utilise des couleurs douces, des images décoratives (logo et icône de connexion), et des messages temporisés pour améliorer l'expérience

utilisateur. Elle a été pensée pour être claire, professionnelle et facile à utiliser. La partie du code de cette interface est la suivante,

```
package jframe;
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.sql.*;

public class Login {

    private JFrame frmLogin; // Main window frame
    private JTextField userText; // Username input field
    private JPasswordField passText; // Password input field
    private Connection connect; // JDBC connection

    public static void main(String[] args) {

        EventQueue.invokeLater(() -> { // Run GUI on EDT
            try {
                Login window = new Login();
                window.frmLogin.setVisible(true); // Show login window
            } catch (Exception e) {
                e.printStackTrace(); // print errors
            }
        });
    }

    public Login() {
        initialize();
        connect = sql_connect.dbconnection(); // Build UI and connect to DB with feedback
    }

    // Clears the username and password fields
    private void clearFields() {
        .
        .
        .
        .
    }

    // Initializes the GUI components
    private void initialize() {
        .
        ...}}
}
```

### III.5.3. Interface d'accueil (Home Page)

L'image ci-dessous illustre la page d'accueil de l'application, qui sert de point d'entrée vers les différentes fonctionnalités proposées.

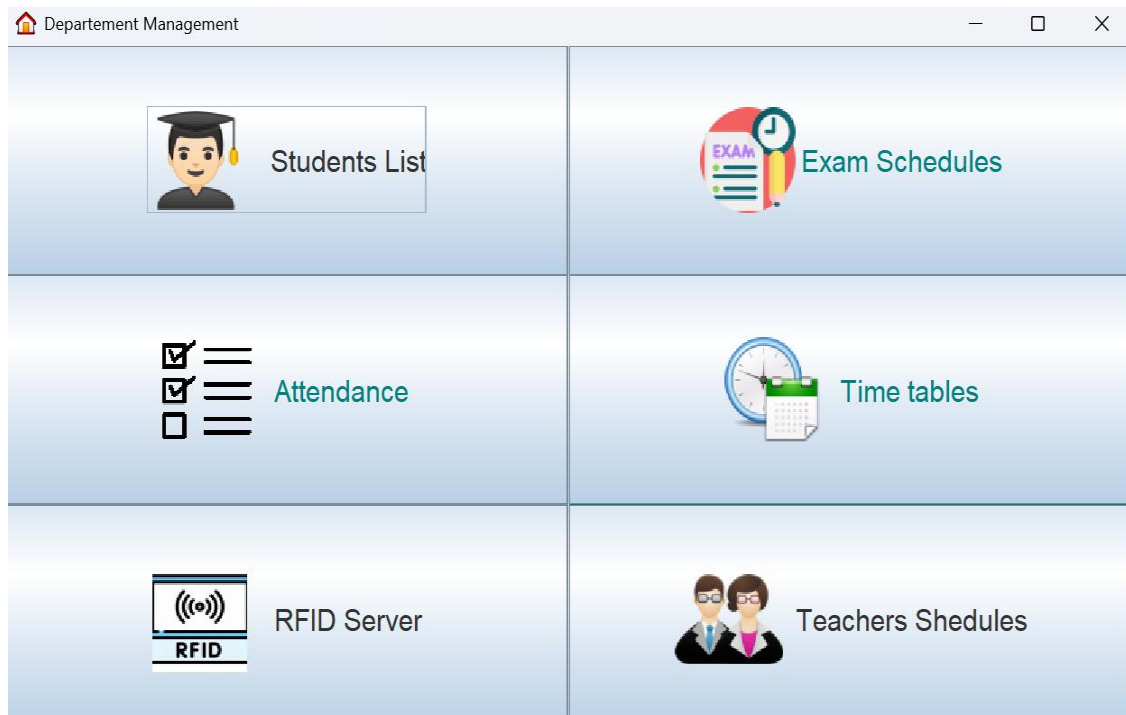


Figure III.12 : Interface d'accueil

L'interface principale de l'application, sert de tableau de bord centralisé pour accéder aux différentes fonctionnalités de gestion du département. Elle présente six boutons, chacun associé à une icône illustrative, permettant de naviguer vers des modules spécifiques :

- **Students List** : Accès à la liste des étudiants.
- **Exam Schedules** : Consultation des plannings d'examens.
- **Attendance** : Gestion des présences.
- **Time Tables** : Visualisation des emplois du temps.
- **RFID Server** : Connexion au serveur RFID.
- **Teachers Schedules** : emplois du temps des enseignants.

Chaque bouton est conçu pour ouvrir la fenêtre correspondante tout en fermant la page actuelle, assurant ainsi une navigation fluide et intuitive. L'interface est conçue pour offrir une expérience utilisateur claire et professionnelle, facilitant la gestion quotidienne des activités du département. Une partie du code de cette interface est la suivante,

```
package jframe;
import java.awt.EventQueue;
import javax.swing.JFrame;
import java.awt.Color;
import java.awt.Toolkit;
import javax.swing.JButton;
import java.awt.GridLayout;
import java.awt.Image;
import javax.swing.ImageIcon;
import java.awt.Font;
import java.awt.event.ActionListener;
import java.sql.SQLException;
import java.awt.event.ActionEvent;

public class Page_1 {

    public JFrame Page1;

    public Page_1 () {

        initialize();
    }
    /**
     * Initialize the contents of the frame.
     */
    private void initialize() {

        .
        .
        .
        .
        .
        .

    }

}
```

### III.5.4. Liste des étudiants (Students List)

L'image suivante présente l'interface dédiée à la gestion des étudiants, avant de détailler son fonctionnement dans le code.

| ID | Full Name             | Date of Birth | Gender | Speciality | Degree   | Year | Class               | Reg Number | Phone     | Email                | RFID_tag  |
|----|-----------------------|---------------|--------|------------|----------|------|---------------------|------------|-----------|----------------------|-----------|
| 1  | ferkous dhiyaedine    | 2001-09-23    | M      | TLC        | Master   | 2    | M2_networks_and_tlc | 1936038514 | 540002155 | ferkousdayou@gmail.c | E271D91B  |
| 2  | Touahri akrem         | 2000-03-03    | M      | TLC        | Master   | 2    | M2_networks_and_tlc | 19360367   | 9999      | fff                  | 518EE13F  |
| 3  | grana abderrahman     | 1999-05-04    | M      | ELN        | Master   | 2    | M2_instrumentation  | 87586877   | 866547    | dd444                |           |
| 4  | abdekader amir        | 2004-07-15    | M      | ELN        | Master   | 2    | M2_instrumentation  | 2136038907 | 55555555  | thxjhtd@gmail.com    | EF7D3E343 |
| 5  | elaarbi benmhidi      | 1934-06-07    | M      | TLC        | Master   | 1    | M1_networks_and_tlc | 8765433    | 88888     | rhjdj                |           |
| 6  | mostafa benboulaid    | 1999-09-22    | M      | TLC        | Master   | 1    | M1_networks_and_tlc | 8755555    | 8564      |                      |           |
| 7  | Larbi Ben M'hidi      | 1923-06-15    | M      | ELN        | Bachelor | 3    | L3_eln              | 1001       | 117440513 | larbi.bm@dz.dz       | A182C3D4  |
| 8  | Didouche Mourad       | 1927-07-13    | M      | ELN        | Bachelor | 3    | L3_eln              | 1002       | 117440514 | didouche.m@dz.dz     | B2C3D4E5  |
| 9  | Ali la Pointe         | 1930-05-14    | M      | TLC        | Master   | 1    | M1_systems_of_tlc   | 1003       | 117440515 | ali.lp@dz.dz         | C3D4E5F6  |
| 10 | Hassiba Ben Bouali    | 1938-01-18    | F      | TLC        | Master   | 1    | M1_systems_of_tlc   | 1004       | 117440516 | hassiba.bb@dz.dz     | D4E5F6G7  |
| 11 | Krim Belkacem         | 1922-12-14    | M      | TLC        | Master   | 2    | M2_systems_of_tlc   | 1005       | 117440517 | krim.b@dz.dz         | E5F6G7H8  |
| 12 | Zighoud Youcef        | 1921-02-18    | M      | TLC        | Master   | 2    | M2_systems_of_tlc   | 1006       | 117440518 | zighoud.y@dz.dz      | F6G7H8I9  |
| 13 | Amirouche Ait Hamouda | 1926-10-31    | M      | ELN        | Master   | 1    | M1_instrumentation  | 1007       | 117440519 | amirouche.a@dz.dz    | G7H8I9J0  |
| 14 | Abane Ramdane         | 1920-06-10    | M      | TLC        | Master   | 2    | M2_networks_and_tlc | 1008       | 78000008  | abane.r@dz.dz        | H8I9J0K1  |
| 15 | Boudiaf Mohamed       | 1919-06-23    | M      | TLC        | Master   | 1    | M1_networks_and_tlc | 1009       | 8800009   | boudiaf.m@dz.dz      | K1L2M3N4  |
| 16 | Djamila Bouhired      | 1935-06-24    | F      | TLC        | Master   | 2    | M2_networks_and_tlc | 1010       | 117440520 | djamila.b@dz.dz      | L2M3N4O5  |

Figure III.13 : Interface de Liste des étudiants

L'interface "**List of Students**" permet d'afficher tous les étudiants enregistrés dans le système de façon claire et bien organisée. L'utilisateur peut voir des informations importantes comme le nom, le prénom, le matricule et l'identifiant RFID. En plus de la consultation, il est aussi possible d'ajouter modifier ou bien supprimer des étudiants. Cette interface rend la gestion des étudiants facile, rapide et accessible, même pour les utilisateurs non spécialisés. Le code de cette interface est le suivant,

```
package jframe;
import javax.swing.*;
import javax.swing.filechooser.FileNameExtensionFilter;
import javax.swing.table.DefaultTableModel;
import java.awt.*;
import java.awt.event.MouseAdapter;
import java.awt.event.MouseEvent;
import java.io.BufferedReader;
import java.io.IOException;
import java.io.*;
import java.net.*;
import java.sql.*;
import java.util.Vector;
public class Students_List extends JFrame {
private Connection connection;
```

```

private JTextField fullNameField, dobField, specialityField, regNumberField, phoneField,
emailField, searchField, RfidField;
private JComboBox<String>genderComboBox, classComboBox, degreeComboBox, yearComboBox;
private JButton addButton, deleteButton, editButton, refreshButton, searchButton, importButton;
private JTable studentTable;
private DefaultTableModel tableModel;
private RFIDServer rfidServer;
private JButton readRFIDButton1;
public Students_List() {
    initializeUI();
    initializeTable();
    refreshTable();
    startRFIDServer();
}

privatevoid startRFIDServer() {
    rfidServer = new RFIDServer();
    try {
        rfidServer.start();
    } catch (IOException e) {
        JOptionPane.showMessageDialog(this, "Failed to start RFID server: " +
            e.getMessage(), "Error", JOptionPane.ERROR_MESSAGE);
    }
}

privatevoid initializeUI() {
    .
    .
    .
}

```

### III.5.5. Gestion des présences (Attendance)

L'image suivante représente l'interface dédiée à la gestion des présences.

| RFID_tag | Full Name          | Module               | Status | Class               | Date       |
|----------|--------------------|----------------------|--------|---------------------|------------|
| E271D91B | ferkous dhyaeddine | Technologies du Web  | A      | M2_networks_and_tlc | 2025-06-03 |
| E271D91B | ferkous dhyaeddine | Technologies du Web  | A      | M2_networks_and_tlc | 2025-05-27 |
| E271D91B | ferkous dhyaeddine | Technologies du Web  | A      | M2_networks_and_tlc | 2025-05-20 |
| E271D91B | ferkous dhyaeddine | Technologies du Web  | A      | M2_networks_and_tlc | 2025-05-13 |
| 518EE13F | Touahri akrem      | Technologies du Web  | A      | M2_networks_and_tlc | 2025-06-03 |
| 518EE13F | Touahri akrem      | Technologies du Web  | A      | M2_networks_and_tlc | 2025-05-27 |
| 518EE13F | Touahri akrem      | Technologies du Web  | A      | M2_networks_and_tlc | 2025-05-20 |
| 518EE13F | Touahri akrem      | Technologies du Web  | A      | M2_networks_and_tlc | 2025-05-13 |
|          | elaarbi benmhidi   | Propagation et anten | A      | M1_networks_and_tlc | 2025-06-05 |
|          | elaarbi benmhidi   | Propagation et anten | A      | M1_networks_and_tlc | 2025-05-29 |
|          | elaarbi benmhidi   | Propagation et anten | A      | M1_networks_and_tlc | 2025-05-22 |
|          | elaarbi benmhidi   | Propagation et anten | A      | M1_networks_and_tlc | 2025-05-15 |
| H819J0K1 | Abane Ramdane      | Technologies du Web  | A      | M2_networks_and_tlc | 2025-06-03 |
| H819J0K1 | Abane Ramdane      | Technologies du Web  | A      | M2_networks_and_tlc | 2025-05-27 |
| H819J0K1 | Abane Ramdane      | Technologies du Web  | A      | M2_networks_and_tlc | 2025-05-20 |
| H819J0K1 | Abane Ramdane      | Technologies du Web  | A      | M2_networks_and_tlc | 2025-05-13 |
| K1L2M3N4 | Boudiaf Mohamed    | Propagation et anten | A      | M1_networks_and_tlc | 2025-06-05 |
| K1L2M3N4 | Boudiaf Mohamed    | Propagation et anten | A      | M1_networks_and_tlc | 2025-05-29 |
| K1L2M3N4 | Boudiaf Mohamed    | Propagation et anten | A      | M1_networks_and_tlc | 2025-05-22 |
| K1L2M3N4 | Boudiaf Mohamed    | Propagation et anten | A      | M1_networks_and_tlc | 2025-05-15 |
| L2M3N4O5 | Djamila Bouhired   | Technologies du Web  | A      | M2_networks_and_tlc | 2025-06-03 |
| L2M3N4O5 | Djamila Bouhired   | Technologies du Web  | A      | M2_networks_and_tlc | 2025-05-27 |
| L2M3N4O5 | Djamila Bouhired   | Technologies du Web  | A      | M2_networks_and_tlc | 2025-05-20 |

Figure III.14 : Attendance interface

Cette interface affiche les présences des étudiants sous forme de tableau.

L'utilisateur peut filtrer les données selon la classe, le module, le mois, ou le nom de l'étudiant.

Les résultats sont affichés dans un tableau.

Les modules proposés s'adaptent automatiquement selon la classe sélectionnée.

L'objectif principal est de consulter et gérer facilement les informations de présence. Le code est le suivant,

```
package jframe;

import javax.swing.*;
import javax.swing.table.DefaultTableModel;
import javax.swing.table.JTableHeader;
import java.awt.*;
import java.awt.Font;
import java.awt.Image;
import java.io.FileOutputStream;
import java.sql.*;
import java.util.*;
import com.itextpdf.text.*;
import com.itextpdf.text.pdf.*;

public class Attendance extends JFrame {
    private JTable table;
    private DefaultTableModel model;
    private JComboBox<String> classFilter, moduleFilter, monthFilter;
    private JTextField nameFilter;
    private JButton filterButton, resetButton, exportPdfButton;

    // Mapping class names to their corresponding modules
    private final Map<String, java.util.List<String>> classModules = new HashMap<>();

    public Attendance() {
        .
        .
        .
        .
        .
        .
    }
}
```

```

    }
}

```

### III.5.6. Planning des examens (Exam Schedules)

L'illustration suivante présente l'interface utilisée pour consulter le planning des examens.

The screenshot shows a window titled "Exam Timetable Manager". At the top, there is a "Select Class:" dropdown menu set to "M1\_networks\_and\_tlc", a "Start Date (Saturday):" date picker set to "2025-06-14", and an "Export to PDF" button. Below this is a table with the following columns: day, module, date, start\_time, end\_time, duration, teacher1, and teacher2. The table displays exam schedules for the module "e1n fondamentale" from Saturday, June 14th to Tuesday, June 24th, 2025. The schedule is organized by day, with each day having two rows for dates. The times are 09:00:00 to 11:00:00, with a duration of 2h. The teachers listed are "teach a" and "teacher b".

| day       | module           | date       | start_time | end_time | duration | teacher1 | teacher2  |
|-----------|------------------|------------|------------|----------|----------|----------|-----------|
| Saturday  | e1n fondamentale | 2025-06-14 | 09:00:00   | 11:00:00 | 2h       | teach a  | teacher b |
|           |                  | 2025-06-14 |            |          |          |          |           |
| Sunday    |                  | 2025-06-15 |            |          |          |          |           |
|           |                  | 2025-06-15 |            |          |          |          |           |
| Monday    |                  | 2025-06-16 |            |          |          |          |           |
|           |                  | 2025-06-16 |            |          |          |          |           |
| Tuesday   |                  | 2025-06-17 |            |          |          |          |           |
|           |                  | 2025-06-17 |            |          |          |          |           |
| Wednesday |                  | 2025-06-18 |            |          |          |          |           |
|           |                  | 2025-06-18 |            |          |          |          |           |
| Thursday  |                  | 2025-06-19 |            |          |          |          |           |
|           |                  | 2025-06-19 |            |          |          |          |           |
| Saturday  |                  | 2025-06-21 |            |          |          |          |           |
|           |                  | 2025-06-21 |            |          |          |          |           |
| Sunday    |                  | 2025-06-22 |            |          |          |          |           |
|           |                  | 2025-06-22 |            |          |          |          |           |
| Monday    |                  | 2025-06-23 |            |          |          |          |           |
|           |                  | 2025-06-23 |            |          |          |          |           |
| Tuesday   |                  | 2025-06-24 |            |          |          |          |           |

Figure III.15 : Planning des examens

L'interface TEACH permet simplement d'afficher les séances programmées des enseignants selon les classes sélectionnées. Elle offre un aperçu clair et filtrable par nom d'enseignant ou par classe, sans fonctionnalités de modification ou de gestion. Son rôle principal est de consulter facilement les données déjà présentes, avec la possibilité de les exporter en PDF pour un usage externe. Le code est le suivant,

```

package jframe;
import javax.swing.*.*;
import javax.swing.event.DocumentEvent;
import javax.swing.event.DocumentListener;
import javax.swing.table.DefaultTableModel;
import javax.swing.table.TableRowSorter;
import java.awt.*.*;
import java.io.FileOutputStream;
import java.io.IOException;
import java.sql.*.*;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;
import java.util.regex.Pattern;

```

```

import com.itextpdf.text.Document;
import com.itextpdf.text.DocumentException;
import com.itextpdf.text.pdf.PdfPTable;
import com.itextpdf.text.pdf.PdfWriter;
/**
 * Interface degestiondesemploisdutempsdesenseignants
 * Permetdevisualiser, filtreret exporter lesplannings par classe
 */
public class TEACH extends JFrame {
    // Déclarationdescomposants
    private List<String> allClasses = new ArrayList<>(); // Listedes classes disponibles
    private Connection connection; // Connexion à la base dedonnées
    private JComboBox<String> classCombo; // Sélecteurdeclasse
    private JTextField filterField; // Champdefiltrage par enseignant
    private JTable table; // Tableau desdonnées
    private DefaultTableModel model; // Modèlededonnéesdu tableau
    private TableRowSorter<DefaultTableModel> sorter; // Gestiondutri/filtre
    public TEACH() {
        super("Teachers Management");
        initializeClassList(); // Initialisationdelalistedes classes
        configureWindow(); // Configuration delafenêtre
        setupDatabaseConnection(); // Connexion à la base dedonnées
        initUI(); // Initialisationde l'interface
    }
    // Initialiselalistedes classes disponibles
    private void initializeClassList() {
        allClasses.add("All Classes"); // Option "Toutesles classes"
        allClasses.addAll(Arrays.asList( // Ajoutdes classes spécifiques
            "M1_networks_and_tlc", "L3_eln", "M2_networks_and_tlc",
            "M1_systems_of_tlc", "M2_systems_of_tlc",
            "M1_instrumentation", "M2_instrumentation"
        ));
    }
    // Configure lespropriétésdelafenêtre
    private void configureWindow() {
        Image iconn = new ImageIcon(getClass().getResource("/Teachers-icon.png")).getImage();
        setIconImage(iconn); // Icônedel'application
        setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE); // Fermeturepropre
        setExtendedState(JFrame.MAXIMIZED_BOTH); // Fenêtramaximisée
        setLayout(new BorderLayout()); // Gestionnairede disposition
    }
    // Établitleconnexion à la base dedonnées
    private void setupDatabaseConnection() {
        try {
            connection = DriverManager.getConnection(
                "jdbc:mysql://localhost:3306/time", "root", "");
        } catch (SQLException ex) {
            JOptionPane.showMessageDialog(this, "Database connection error: " + ex.getMessage(),
                "Error", JOptionPane.ERROR_MESSAGE);
            System.exit(1); // Arrêtencas d'erreur
        }
    }
    // Initialise l'interface utilisateur
    private void initUI() {
        .
        .
        .
        .
    }
}

```

```

.
.
}
}

```

### III.5.7. Emploi du temps des étudiants (Time Tables)

Les figures suivantes présentent l'interface d'emploi du temps ainsi que la fenêtre de modification.

| Day       | 8h:00-9h:30                                                   | 9h:30-11h:00                                                 | 11h:00-12h:30                                              | 12h:30-14h:00 | 14h:00-15h:30                                                | 15h:30-17h:00 | Actions |
|-----------|---------------------------------------------------------------|--------------------------------------------------------------|------------------------------------------------------------|---------------|--------------------------------------------------------------|---------------|---------|
| sunday    | Communications num ava...<br>Cours<br>L.1.3<br>A.Boukharrouba | Communications num av...<br>Cours<br>L.1.3<br>A.Boukharrouba | OOP C++<br>Cours<br>L.1.3<br>K.Abalnia                     |               | Normes et Protocoles<br>Cours<br>L.1.3<br>O.Mehri            |               | Edit    |
| monday    | Routage IP<br>Cours<br>L.1.3<br>M.Gueroui                     | Routage IP<br>TD<br>L.1.3<br>M.Gueroui                       | Communications num ava...<br>TD<br>L.1.3<br>A.Boukharrouba |               | Anglais technique et termi.<br>Cours<br>L.1.3<br>M.Bouaouina |               | Edit    |
| tuesday   | Système UWB / VLC / PLC<br>Cours<br>L.1.3<br>D.Abed           | Traitement avancé du sig...<br>Cours<br>L.1.3<br>M.Ghadjati  | Traitement avancé du sig...<br>TD<br>L.1.3<br>M.Ghadjati   |               |                                                              |               | Edit    |
| wednesday |                                                               |                                                              |                                                            |               |                                                              |               | Edit    |
| thursday  | Propagation et antennes<br>Cours<br>L.1.3<br>M.Chouhbane      | Propagation et antennes<br>TD<br>L.1.3<br>M.Chouhbane        |                                                            |               |                                                              |               | Edit    |

Figure III.16 : Interface d'Emploi du temps

**Edit tuesday**

8h:00-9h:30  
Module:  Type:  Room:

9h:30-11h:00  
Module:  Type:  Room:

11h:00-12h:30  
Module:  Type:  Room:

12h:30-14h:00  
Module:  Type:  Room:

14h:00-15h:30  
Module:  Type:  Room:

15h:30-17h:00  
Module:  Type:  Room:

Figure III.17 : Modification d'Emploi du temps

L'interface TimetableApp est une interface Java Swing connectée à la base de données MySQL, permettant l'affichage et l'édition d'emplois du temps universitaires sous forme de grille. Elle offre

une visualisation claire des séances (module, type, salle, enseignant), un code couleur pour distinguer les cours (vert clair) des TD (jaune clair), une actualisation dynamique, l'édition rapide des données via des dialogues, et l'exportation en PDF grâce à la bibliothèque iText. La partie de code est la suivante

```
package jframe;
// Required imports
import java.awt.Image;
import javax.swing.*;
import javax.swing.table.*;
import java.awt.*;
import java.awt.Font;
import java.awt.event.*;
import java.sql.*;
import java.util.Date;
import com.itextpdf.text.*;
import com.itextpdf.text.pdf.*;
import java.io.FileOutputStream;
import java.io.File;
public class TimetableApp extends JFrame {
    private JTable timetableTable; // Table to display timetable
    private Connection connection; // Database connection
    private JComboBox<String> tableComboBox; // Dropdown for timetable selection
    private JLabel titleLabel; // Title display
    // Session time slots
    private final String[] SESSION_TIMES = {
        "8h:00-9h:30", "9h:30-11h:00", "11h:00-12h:30",
        "12h:30-14h:00", "14h:00-15h:30", "15h:30-17h:00"
    };
    // Database table names
    private final String[] TABLE_NAMES = {
        "M1_networks_and_tlc", "L3_eln", "M2_networks_and_tlc",
        "M1_systems_of_tlc", "M2_systems_of_tlc",
        "M1_instrumentation", "M2_instrumentation"
    };
    // Main constructor
    public TimetableApp() throws SQLException {
        super("University Timetable System");
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setSize(1300, 800);
    }
    // Set application icon
    Image iconn = new ImageIcon(getClass().getResource("/tt.png")).getImage();
    setIconImage(iconn);
    setLocationRelativeTo(null); // Center window
    // Establish database connection
    connection = DriverManager.getConnection(
        "jdbc:mysql://localhost:3306/time", "root", "");
    initializeUI(); // Initialize user interface
    }
    // UI initialization
    private void initializeUI() {
        ...
    }
}
```

```

    .
    .
    .
    .
    }
}

```

### III.5.8. Planning des enseignants (Teachers Schedules)

La figure suivante montre l'interface utilisée pour consulter le planning des enseignants.

The screenshot shows a window titled "Teachers Management" with a toolbar containing a home icon, a "Class:" dropdown menu set to "All Classes", a "Filter Teacher:" text input, and an "Export PDF" button. Below the toolbar is a table with the following columns: Class, Day, Teacher, Type, Room, and Time. The table contains multiple rows of schedule data for various classes and teachers.

| Class               | Day       | Teacher         | Type  | Room   | Time        |
|---------------------|-----------|-----------------|-------|--------|-------------|
| M1_networks_and_tlc | sunday    | A.Boukharrouba  | Cours | L.1.3  | 08h00-09h30 |
| M1_networks_and_tlc | monday    | M.Gueroui       | Cours | L.1.3  | 08h00-09h30 |
| M1_networks_and_tlc | tuesday   | D.Abed          | Cours | L.1.3  | 08h00-09h30 |
| M1_networks_and_tlc | thursday  | M.Chouhbane     | Cours | L.1.3  | 08h00-09h30 |
| M1_networks_and_tlc | sunday    | A.Boukharrouba  | Cours | L.1.3  | 09h30-11h00 |
| M1_networks_and_tlc | monday    | M.Gueroui       | TD    | L.1.3  | 09h30-11h00 |
| M1_networks_and_tlc | tuesday   | M.Ghadjati      | Cours | L.1.3  | 09h30-11h00 |
| M1_networks_and_tlc | thursday  | M.Chouhbane     | TD    | L.1.3  | 09h30-11h00 |
| M1_networks_and_tlc | sunday    | K.Abainia       | Cours | L.1.3  | 11h00-12h30 |
| M1_networks_and_tlc | monday    | A.Boukharrouba  | TD    | L.1.3  | 11h00-12h30 |
| M1_networks_and_tlc | tuesday   | M.Ghadjati      | TD    | L.1.3  | 11h00-12h30 |
| M1_networks_and_tlc | sunday    | O.Mehri         | Cours | L.1.3  | 14h00-15h30 |
| M1_networks_and_tlc | monday    | M.Bouaouina     | Cours | L.1.3  | 14h00-15h30 |
| L3_eln              | sunday    | A. Khalfallaoui | Cours | L.2.1  | 08h00-09h30 |
| L3_eln              | tuesday   | O.Mehri         | Cours | L.2.2  | 08h00-09h30 |
| L3_eln              | wednesday | S.Benjoudi      | Cours | L.2.1  | 08h00-09h30 |
| L3_eln              | sunday    | A. Khalfallaoui | TD    | L.2.1  | 09h30-11h00 |
| L3_eln              | tuesday   | O.Mehri         | TD    | L.2.2  | 09h30-11h00 |
| L3_eln              | wednesday | S.Benjoudi      | TD    | L.2.1  | 09h30-11h00 |
| L3_eln              | monday    | M.Nemissi       | Cours | L.2.3  | 11h00-12h30 |
| L3_eln              | sunday    | N.Guebgoub      | Cours | L.2.1  | 12h30-14h00 |
| L3_eln              | tuesday   | A.Younsi        | Cours | L.1.2  | 12h30-14h00 |
| L3_eln              | monday    | M.Taba          | Cours | L.1.2B | 14h00-15h30 |
| M2_networks_and_tlc | monday    | A.Boukharrouba  | Cours | L.2.3  | 08h00-09h30 |
| M2_networks_and_tlc | wednesday | A.Chaabane      | Cours | L.1.3  | 08h00-09h30 |
| M2_networks_and_tlc | sunday    | O.Mehri         | Cours | L.1.1  | 09h30-11h00 |
| M2_networks_and_tlc | monday    | A.Boukharrouba  | TD    | L.2.3  | 09h30-11h00 |
| M2_networks_and_tlc | wednesday | A.Chaabane      | Cours | L.1.3  | 09h30-11h00 |
| M2_networks_and_tlc | sunday    | O.Mehri         | TD    | L.1.1  | 11h00-12h30 |
| M2_networks_and_tlc | wednesday | B.Griouz        | Cours | L.1.3  | 11h00-12h30 |
| M2_networks_and_tlc | monday    | H.Boudouda      | Cours | L.1.1  | 12h30-14h00 |
| M2_networks_and_tlc | tuesday   | K.Abainia       | Cours | L.1.1  | 12h30-14h00 |
| M2_networks_and_tlc | sunday    | N.Guebgoub      | Cours | L.2.1  | 14h00-15h30 |
| M2_networks_and_tlc | monday    | L.Grini         | Cours | L.1.1  | 14h00-15h30 |
| M2_networks_and_tlc | tuesday   | K.Abainia       | TD    | L.1.1  | 14h00-15h30 |
| M2_networks_and_tlc | tuesday   | N.Djedouani     | TD    | L.1.1  | 15h30-17h00 |

Figure III.18 : Planning des enseignants

Cette interface graphique est conçue pour faciliter la visualisation des emplois du temps des enseignants. Elle offre une vue claire et interactive des séances de cours, permettant aux utilisateurs de filtrer les informations par classe ou par nom d'enseignant. Grâce à des fonctionnalités telles que l'exportation en PDF, elle simplifie l'accès et le partage des plannings. L'interface, intuitive et bien structurée, utilise des composants Swing pour assurer une expérience utilisateur fluide et professionnelle.

```

package jframe;

// Required Swing and Java libraries
import javax.swing.*;
import javax.swing.event.DocumentEvent;
import javax.swing.event.DocumentListener;
import javax.swing.table.DefaultTableModel;
import javax.swing.table.TableRowSorter;

```

```

import java.awt.*;
import java.io.FileOutputStream;
import java.io.IOException;
import java.sql.*;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;
import java.util.regex.Pattern;

import com.itextpdf.text.Document;
import com.itextpdf.text.DocumentException;
import com.itextpdf.text.pdf.PdfPTable;
import com.itextpdf.text.pdf.PdfWriter;

// Main TEACH class for managing teacher schedules
public class TEACH extends JFrame {
    private List<String> allClasses = new ArrayList<>(); // List of all class table names
    private Connection connection; // JDBC connection to MySQL
    private JComboBox<String> classCombo; // Dropdown to select a class
    private JTextField filterField; // Text field for filtering by teacher name
    private JTable table; // Table to display data
    private DefaultTableModel model; // Model backing the table
    private TableRowSorter<DefaultTableModel> sorter; // Sorter for filtering rows

    public TEACH() {
        super("Teachers Management"); // Frame title
        allClasses.add("All Classes"); // Add universal selection
        allClasses.addAll(Arrays.asList( // Add specific class tables
            "M1_networks_and_tlc", "L3_eln", "M2_networks_and_tlc",
            "M1_systems_of_tlc", "M2_systems_of_tlc",
            "M1_instrumentation", "M2_instrumentation"
        ));
        Image iconn = new ImageIcon(getClass().getResource("/Teachers-
        icon.png")).getImage(); // Load icon
        setIconImage(iconn);
        setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE); // Close only this window
        Dimension screenSize = Toolkit.getDefaultToolkit().getScreenSize(); // Get
        screen dimensions
        setBounds(0, 0, screenSize.width, screenSize.height); // Maximize to screen
        setLayout(new BorderLayout()); // Use BorderLayout for layout

        try {
            connection = DriverManager.getConnection("jdbc:mysql://localhost:3306/time", "root",
            ""); // DB connection
        } catch (SQLException ex) {
            JOptionPane.showMessageDialog(this, "Database connection error: " +
            ex.getMessage(), "Error", JOptionPane.ERROR_MESSAGE);
            System.exit(1); // Exit if DB fails
        }

        JPanel topPanel = new JPanel(new BorderLayout()); // Top container panel
        topPanel.setBorder(BorderFactory.createEmptyBorder(5, 5, 5, 5)); // Add padding

        JButton backButton = createBackButton(); // Create back button
        topPanel.add(backButton, BorderLayout.WEST); // Add to left

        JPanel ctrl = new JPanel(new FlowLayout(FlowLayout.LEFT, 10, 0)); // Control
        panel

```

```

classCombo = new JComboBox<>(allClasses.toArray(new String[0])); // Class selector
classCombo.addActionListener(e -> loadData()); // Reload data on change
ctrl.add(new JLabel("Class:")); // Label for combo
ctrl.add(classCombo); // Add combo box

filterField = new JTextField(15); // Filter field for teacher
filterField.getDocument().addDocumentListener(new DocumentListener() {
    public void insertUpdate(DocumentEvent e) { applyFilter(); }
    public void removeUpdate(DocumentEvent e) { applyFilter(); }

    public void changedUpdate(DocumentEvent e) { applyFilter(); }
});
ctrl.add(new JLabel("Filter Teacher:")); // Label for filter
ctrl.add(filterField); // Add filter field

JButton exportBtn = new JButton("Export PDF"); // PDF export button
exportBtn.addActionListener(e -> exportToPDF()); // Trigger export
ctrl.add(exportBtn); // Add to control panel

topPanel.add(ctrl, BorderLayout.CENTER); // Add control panel to center

JButton closeBtn = new JButton("X"); // Close button (top right)
closeBtn.setBorderPainted(false);
closeBtn.setContentAreaFilled(false);
closeBtn.setFont(closeBtn.getFont().deriveFont(Font.BOLD, 16f));
closeBtn.addActionListener(e -> dispose()); // Close window
topPanel.add(closeBtn, BorderLayout.EAST); // Add to right

add(topPanel, BorderLayout.NORTH); // Add top panel to frame

model = new DefaultTableModel(new String[]{"Class", "Day", "Teacher", "Type", "Room",
"Time"}, 0); // Define table columns
table = new JTable(model); // Create table
sorter = new TableRowSorter<>(model); // Initialize sorter
table.setRowSorter(sorter); // Apply sorter
add(new JScrollPane(table), BorderLayout.CENTER); // Add table with scroll

classCombo.setSelectedIndex(0); // Default to "All Classes"
loadData(); // Initial data load
}

private JButton createBackButton() {
    ..
    .
    .
    .
    .
    .
    .
}
}

```

### III.5.9. Interface du serveur RFID (RFID Server)

La figure suivante illustre l'interface utilisée pour gérer les connexions RFID et suivre les présences automatiquement via le serveur.

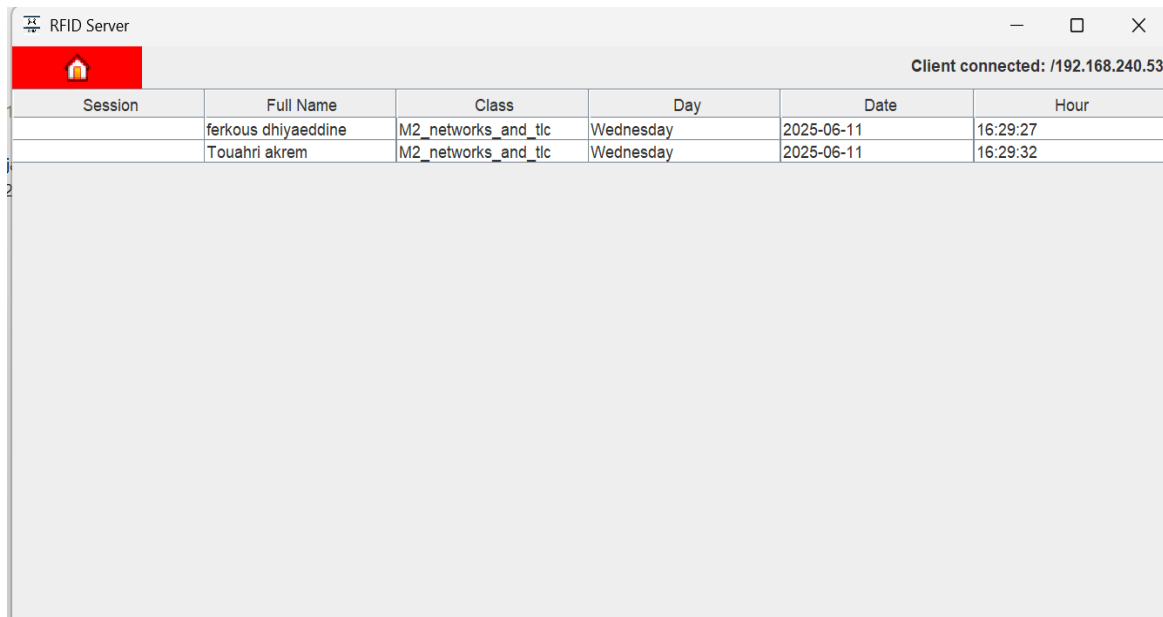


Figure III.19 : RFID server

```
package jframe;
import javax.swing.*;
import javax.swing.table.DefaultTableModel;
import java.awt.*;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;
import java.net.Socket;
import java.sql.*;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.Locale;
// Main JFrame class implementing listener for RFIDServer
public class RFid extends JFrame implements RFIDServer.RfidListener {
    private JTable table; // Table to display attendance records
    private DefaultTableModel model; // Model for table data
    private Connection conn; // MySQL database connection
    private RFIDServer rfidServer; // Server that receives RFID data
    private JLabel statusLabel; // Displays connection status
    // Session info
    private int currentSessionId = -1;
    private String currentModuleName = null;
    private String currentClassName = null;
    private String presenceTableName = null;
    // Database connection parameters
    private static final String URL =

    public RFid(RFIDServer server) {
        this.rfidServer = server;
        setTitle("RFID Server");
        setSize(900, 450);
    }
}
```

```

        Image iconn = new ImageIcon(getClass().getResource("/OIP.jpg")).getImage();
        setIconImage(iconn);
        setLocationRelativeTo(null);
        setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
        setLayout(new BorderLayout());
// Set up table columns
        String[] columns = {"Session", "Full Name", "Class", "Day", "Date", "Hour"};
        model = new DefaultTableModel(columns, 0);
        table = new JTable(model);
// Top panel with return button and status label
        JPanel topPanel = new JPanel(new BorderLayout());
        JButton returnButton = new JButton();
        ImageIcon icon = new ImageIcon(this.getClass().getResource("/back-icon.PNG"));
        Image scaled = icon.getImage().getScaledInstance(20, 20, Image.SCALE_SMOOTH);
        returnButton.setIcon(new ImageIcon(scaled));
        styleButton(returnButton, new Color(255, 0, 0));
// Return to previous interface and stop server
        returnButton.addActionListener(e -> {
            if (rfidServer != null) {
                try {
                    rfidServer.stop();
                } catch (Exception ex) {
                    ex.printStackTrace();
                }
            }
            this.dispose();
            new Page_1().Page1.setVisible(true);
        });
// Status label (connected/disconnected)
        JLabel statusLabel = new JLabel("Status: Waiting for connections");
        JPanel rightStatusPanel = new JPanel(new FlowLayout(FlowLayout.RIGHT));
        rightStatusPanel.add(statusLabel);
        topPanel.add(returnButton, BorderLayout.WEST);
        topPanel.add(rightStatusPanel, BorderLayout.CENTER);
        add(topPanel, BorderLayout.NORTH);
        add(new JScrollPane(table), BorderLayout.CENTER);
// Connect to database
        try {
            conn = DriverManager.getConnection(URL, USER, PASS);
        } catch (SQLException e) {
            JOptionPane.showMessageDialog(this, "Database connection failed: " +
            e.getMessage(), "Error", JOptionPane.ERROR_MESSAGE);
            return;
        }
// Ensure server is stopped and connection is closed on window close
        addWindowListener(new WindowAdapter() {
            @Override
            public void windowClosing(WindowEvent e) {
                if (rfidServer != null) {
                    rfidServer.stop();
                }
                closeConnection();
            }
        });
// Called when an RFID tag is received
        @Override
        public void onRfidReceived(String rfidTag) {

```

```

        SwingUtilities.invokeLater(() -> {
            Date now = new Date();
            String currentDay = new SimpleDateFormat("EEEE",
Locale.ENGLISH).format(now);
            Time currentTime = new Time(now.getTime());
// Get student full name and class
            String studentInfo = lookupStudent(rfidTag);
            if (studentInfo == null) return;
            String[] parts = studentInfo.split(";");
            String fullName = parts[0];
            String className = parts[1];
// Check if there is a matching session
            String sqlSession = "SELECT session_id, Module FROM td_table WHERE Day = ?
AND session_start <= ? AND session_end >= ? AND class = ?";
            int sessionId = -1;
            String module = null;

            try (PreparedStatement pst = conn.prepareStatement(sqlSession)) {
                pst.setString(1, currentDay);
                pst.setTime(2, currentTime);
                pst.setTime(3, currentTime);
                pst.setString(4, className);
                try (ResultSet rs = pst.executeQuery()) {
                    if (rs.next()) {
                        sessionId = rs.getInt("session_id");
                        module = rs.getString("Module");
                    }
                }
            } catch (SQLException ex) {
                ex.printStackTrace();
            }

// If a new session is detected, create/reset the presence table
            if (sessionId > 0 && sessionId != currentSessionId) {
                currentSessionId = sessionId;
                currentModuleName = module;
                currentClassName = className;
                presenceTableName = "presence_session_" + currentSessionId;
                createAndPopulatePresenceTable(currentSessionId, currentClassName,
currentModuleName);
            }

// Mark student as present or insert if not already present
            if (presenceTableName != null) {
                try {
                    String sqlUpdate = "UPDATE " + presenceTableName + " SET Status =
'P' WHERE RFID_tag = ?";
                    try (PreparedStatement pst = conn.prepareStatement(sqlUpdate)) {
                        pst.setString(1, rfidTag);
                        if (pst.executeUpdate() == 0) {
                            String sqlInsert = "INSERT INTO " + presenceTableName + "
(RFID_tag, `Full Name`, module, Status, class) VALUES (?, ?, ?, 'P', ?)";
                            try (PreparedStatement pstIns = conn.prepareStatement(sqlInsert)) {
                                pstIns.setString(1, rfidTag);
                                pstIns.setString(2, fullName);
                                pstIns.setString(3, currentModuleName);
                                pstIns.setString(4, className);
                                pstIns.executeUpdate();
                            }
                        }
                    }
                }
            }
        })
    }

```

```

        }
    } catch (SQLException e) {
e.printStackTrace();
    }
}

// Add record to table UI
model.addRow(new Object[]{
currentModuleName,
fullName,
className,
currentDay,
new SimpleDateFormat("yyyy-MM-dd").format(now),
new SimpleDateFormat("HH:mm:ss").format(now)
});
});
}

// Create/reset the session-specific presence table and populate it with all students
marked absent
private void createAndPopulatePresenceTable(int sessionId, String className, String
module) {
    String tableName = "presence_session_" + sessionId;
    try (Statement stmt = conn.createStatement()) {
stmt.executeUpdate("CREATE TABLE IF NOT EXISTS " + tableName + " (RFID_tag VARCHAR(50)
PRIMARY KEY, `Full Name` VARCHAR(100), module VARCHAR(20), Status CHAR(1), class
VARCHAR(20))");
stmt.executeUpdate("DELETE FROM " + tableName);
    } catch (SQLException e) {
e.printStackTrace();
    }
    String sql = "SELECT RFID_tag, `Full Name` FROM students WHERE class = ?";
    try (PreparedStatement pst = conn.prepareStatement(sql)) {
pst.setString(1, className);
    try (ResultSet rs = pst.executeQuery()) {
        String insert = "INSERT INTO " + tableName + " (RFID_tag, `Full Name`,
module, Status, class) VALUES (?, ?, ?, 'A', ?)";
        try (PreparedStatement pstIns = conn.prepareStatement(insert)) {
            while (rs.next()) {
pstIns.setString(1, rs.getString("RFID_tag"));
pstIns.setString(2, rs.getString("Full Name"));
pstIns.setString(3, module);
pstIns.setString(4, className);
pstIns.executeUpdate();
            }
        }
    }
    } catch (SQLException e) {
e.printStackTrace();
    }
}

// Find student info from database using RFID tag
private String lookupStudent(String rfidTag) {
    try (PreparedStatement pst = conn.prepareStatement("SELECT `Full Name`, class FROM
students WHERE RFID_tag = ?")) {
pst.setString(1, rfidTag);
    try (ResultSet rs = pst.executeQuery()) {
        if (rs.next()) {
return rs.getString("Full Name") + ";" + rs.getString("class");
        }
    }
}

```

```

        }
    } catch (SQLException e) {
        e.printStackTrace();
    }
    return null;
}

// Style the return button
private void styleButton(JButton button, Color bgColor) {
    button.setBackground(bgColor);
    button.setForeground(Color.WHITE);
    button.setFocusPainted(false);
    button.setFont(new Font("Segoe UI", Font.BOLD, 12));
    button.setBorderPainted(false);
    button.setOpaque(true);
    button.setPreferredSize(new Dimension(100, 30));
    button.setMaximumSize(new Dimension(100, 30));
    button.setMinimumSize(new Dimension(100, 30));
}

// Close MySQL connection
private void closeConnection() {
    try {
        if (conn != null && !conn.isClosed()) conn.close();
    } catch (SQLException ignored) {}
}

// Called when a client connects to the server
@Override
public void onClientConnected(Socket client) {
    SwingUtilities.invokeLater(() -> statusLabel.setText("Client connected: " +
        client.getInetAddress()));
}

// Called when a client disconnects from the server
@Override
public void onClientDisconnected(Socket client) {
    SwingUtilities.invokeLater(() -> statusLabel.setText("Client disconnected: " +
        client.getInetAddress()));
}

// Registers this GUI to listen for RFID events from the server
public void attachToServer() {
    if (rfidServer != null) {
        rfidServer.addRfidListener(this);
    }
}
}

```

*code 1 class RFID*

Cette interface utilise la classe suivante :RFIDServer.

```

package jframe;
import java.io.*;
import java.net.ServerSocket;
import java.net.Socket;
import java.sql.*;
import java.util.Map;
import java.util.concurrent.*;
/**
 * RFIDServer is a server that listens for incoming RFID messages from clients.
 * It can notify listeners when RFID tags are received and return the corresponding
 * student's name.

```

```

*/
publicclass RFIDServer implements Runnable {
// Port used for client connections
privatestaticfinalintPORT = 65100;
// Server state flag
privatevolatilebooleanrunning;
// Server socket for incoming connections
private ServerSocket serverSocket;
// List of listeners interested in RFID events
privatefinal CopyOnWriteArrayList<RfidListener>listeners = new
CopyOnWriteArrayList<>();
// Stores the time of the last message (heartbeat) from each client
privatefinal Map<Socket, Long>lastHeartbeat = new ConcurrentHashMap<>();
// Constants for heartbeat checking
privatestaticfinallongHEARTBEAT_TIMEOUT_MS = TimeUnit.HOURS.toMillis(8);
privatestaticfinallongHEARTBEAT_CHECK_INTERVAL_MS = TimeUnit.MINUTES.toMillis(1);
// Database connection information
privatestaticfinal String URL =
"jdbc:mysql://127.0.0.1/pointeuse?characterEncoding=UTF-8";
privatestaticfinal String USER = "root";
privatestaticfinal String PASS = "";
// Interface to handle RFID-related events
publicinterface RfidListener {
void onRfidReceived(String rfid);
void onClientConnected(Socket client);
void onClientDisconnected(Socket client);
}
/**
 * Starts the server and begins listening for clients.
 */
publicsynchronizedvoid start() throws IOException {
if (running) return;
serverSocket = new ServerSocket(PORT);
serverSocket.setReuseAddress(true);
running = true;
// Start the main server thread
new Thread(this, "GG-MainThread").start();
// Start a scheduler to check for inactive clients (heartbeat)
ScheduledExecutorService scheduler =
Executors.newSingleThreadScheduledExecutor();
scheduler.scheduleAtFixedRate(this::monitorHeartbeats,
HEARTBEAT_CHECK_INTERVAL_MS,
HEARTBEAT_CHECK_INTERVAL_MS,
TimeUnit.MILLISECONDS);
}
/**
 * Stops the server.
 */
publicsynchronizedvoid stop() {
if (!running) return;
running = false;
try {
if (serverSocket != null&& !serverSocket.isClosed()) {
serverSocket.close();
}
} catch (IOException e) {
System.err.println("Error stopping server: " + e.getMessage());
}
}

```

```

    }
    public boolean isRunning() {
        return running;
    }
    /**
     * Main loop for accepting new client connections.
     */
    @Override
    public void run() {
        while (running) {
            try {
                Socket clientSocket = serverSocket.accept(); // Accept new client
                lastHeartbeat.put(clientSocket, System.currentTimeMillis()); // Record time of
                connection
                notifyConnected(clientSocket); // Notify listeners
                new Thread(() -> handleClient(clientSocket), "ClientHandler-" +
                clientSocket.getPort()).start(); // Start a new thread for client
            } catch (IOException e) {
                if (running) {
                    System.err.println("Accept error: " + e.getMessage());
                }
            }
        }
    }
    /**
     * Handles communication with a connected client.
     */
    private void handleClient(Socket socket) {
        try {
            BufferedReader reader = new BufferedReader(new
            InputStreamReader(socket.getInputStream()));
            BufferedWriter writer = new BufferedWriter(new
            OutputStreamWriter(socket.getOutputStream()));
            Connection conn = DriverManager.getConnection(URL, USER, PASS)
            ) {
                String line;
                while (running && !socket.isClosed() && (line = reader.readLine()) != null) {
                    String msg = line.trim();
                    if (!msg.isEmpty()) {
                        lastHeartbeat.put(socket, System.currentTimeMillis()); // Update heartbeat
                        if (!"PING".equalsIgnoreCase(msg)) {
                            notifyListeners(msg); // Notify listeners of received RFID
                            // Look up student name in the database
                            String name = lookupStudentName(conn, msg);
                            // Send the student name back to the client
                            writer.write(name + "\n");
                            writer.flush();
                        }
                    }
                }
            } catch (IOException | SQLException e) {
                if (running) {
                    System.err.println("Client handler error: " + e.getMessage());
                }
            } finally {
                cleanupClient(socket); // Remove client on disconnect
            }
        }
    }

```

```

/**
 * Searches the database for a student's full name by RFID tag.
 */
private String lookupStudentName(Connection conn, String rfidTag) {
    String sql = "SELECT `Full Name` FROM students WHERE RFID_tag = ?";
    try (PreparedStatement pst = conn.prepareStatement(sql)) {
        pst.setString(1, rfidTag);
        try (ResultSet rs = pst.executeQuery()) {
            if (rs.next()) {
                return rs.getString("Full Name");
            }
        } catch (SQLException e) {
            System.err.println("DB lookup error: " + e.getMessage());
        }
    }
    return "Unknown";
}

/**
 * Checks if any clients have stopped sending messages and disconnects them if so.
 */
private void monitorHeartbeats() {
    long now = System.currentTimeMillis();
    for (Map.Entry<Socket, Long> entry : lastHeartbeat.entrySet()) {
        Socket sock = entry.getKey();
        long last = entry.getValue();
        if (now - last > HEARTBEAT_TIMEOUT_MS) {
            cleanupClient(sock); // Disconnect inactive client
        }
    }
}

/**
 * Cleans up a client socket and notifies listeners.
 */
private void cleanupClient(Socket socket) {
    lastHeartbeat.remove(socket);
    notifyDisconnected(socket);
    try {
        socket.close();
    } catch (IOException ignored) {}
}

/**
 * Notifies listeners when a client connects.
 */
private void notifyConnected(Socket client) {
    listeners.forEach(l -> l.onClientConnected(client));
}

/**
 * Notifies listeners when a client disconnects.
 */
private void notifyDisconnected(Socket client) {
    listeners.forEach(l -> l.onClientDisconnected(client));
}

/**
 * Notifies listeners when an RFID tag is received.
 */
private void notifyListeners(String rfid) {
    listeners.forEach(l -> l.onRfidReceived(rfid));
}

```

```
/**
 * Registers a new listener for RFID events.
 */
public void addRfidListener(RfidListener listener) {
    listeners.add(listener);
}

/**
 * Removes a registered listener.
 */
public void removeRfidListener(RfidListener listener) {
    listeners.remove(listener);
}
}
```

### III.5.10. Conclusion

En conclusion, nous avons mis en œuvre concrètement le système de pointage que nous avons conçu. Nous avons commencé par découvrir les composants matériels essentiels — notamment l’ESP32, le module RFID RC522 et l’écran LCD en expliquant leur utilité et comment ils s’intègrent dans l’ensemble du montage.

La partie matérielle a été assemblée avec soin, à partir de schémas de branchement clairs et d’un circuit imprimé pensé pour optimiser les connexions. Ensuite, nous avons configuré le tout via l’environnement Arduino, en installant les bibliothèques nécessaires et en développant le programme embarqué qui pilote l’appareil.

En parallèle, une application Java complète a été développée. Elle permet d’afficher et de gérer les données collectées, à travers différentes interfaces ergonomiques pour la gestion des présences, des étudiants, des emplois du temps et des examens. Chaque partie de l’application a été pensée pour être simple à utiliser et efficace.

L’intégration du serveur RFID marque l’aboutissement de cette partie pratique, en assurant la communication entre le dispositif physique et la base de données. Le résultat final est un système cohérent, moderne, et facile à prendre en main, capable de répondre aux besoins quotidiens d’un établissement d’enseignement.

### III.5.11. Perspectives de développement

Dans le cadre de l’évolution continue du système, plusieurs pistes d’amélioration peuvent être envisagées. Il serait possible d’adapter ce travail pour le marquage de présence des enseignants, des employés ainsi que des doctorants. Par ailleurs, une version plus complète de l’application pourrait être développée afin d’assurer une gestion intégrée du département, incluant la création et le suivi des listes de professeurs, la planification des séances de rattrapage, la gestion des notes des étudiants, ainsi

que le contrôle de présence lors des examens. Ces évolutions visent à renforcer la digitalisation des processus pédagogiques et administratifs au sein de l'établissement.



*Conclusion  
Générale*

En conclusion générale, nous avons exploré, conçu et mis en œuvre un système complet de gestion de présence basé sur la technologie RFID. À travers trois chapitres structurés, nous avons successivement introduit les fondements de la RFID, détaillé les outils logiciels et matériels utilisés (comme l'ESP32, le module RC522 et l'écran LCD), et finalement décrit de manière approfondie la réalisation pratique de notre système, tant sur le plan matériel que logiciel. L'intégration de Java avec MySQL, l'utilisation de modules embarqués, ainsi que la communication en temps réel entre les composants nous ont permis de bâtir une solution fiable, efficace et évolutive. Ce système permet de suivre la présence des étudiants de façon automatisée, tout en centralisant les informations dans une application graphique conviviale.

Mais au-delà du contexte académique, ce projet peut être élargi à d'autres secteurs : il représente une base solide pour une application dans les entreprises, les établissements publics, les institutions de santé, les centres de formation ou toute autre structure ayant besoin de gérer les présences de manière automatisée. Grâce à sa flexibilité, il peut être adapté aux employés, enseignants, visiteurs ou même aux patients.

Ce travail ouvre également des perspectives de développement intéressantes, telles que la gestion complète du département, l'intégration de plannings de rattrapage, de notes, ou encore le marquage de présence lors des examens. L'ajout d'un accès multi-utilisateur ou encore la compatibilité mobile pourraient renforcer son adoption à plus grande échelle. En somme, ce projet constitue une première étape vers une digitalisation plus intelligente et automatisée de la gestion de présence, avec un potentiel d'adaptation à une grande variété d'environnements professionnels.

## Références bibliographiques

- [1] G. Martin, Conception de systèmes RFID passifs pour la logistique hospitalière, Thèse de Doctorat, 2018, p. 72.
- [2] A. Lefèvre, Sécurité des données dans les réseaux RFID industriels, Thèse de Doctorat, 2021, p. 155.
- [3] L. Dubois, Impact des interférences électromagnétiques sur les performances RFID, Thèse de Doctorat, 2019, p. 89.
- [4] E. M. Amin, Chipless RFID Sensor for Ubiquitous Sensing, Thèse de Doctorat en Génie Électrique, 2017, p. 215.
- [5] ResearchGate, The RFID System Components Architecture, [en ligne], <https://www.researchgate.net/publication/363496040/figure/fig3/AS:11431281106371051@1670641801553/The-RFID-system-components-architecture.png>
- [6] C. Turcu (éd.), RFID Middleware Design and Architecture, IntechOpen, 2011. DOI: 10.5772/16917.
- [7] J. Grimes, RFID Reader Architecture, Google Patents, 2010, <https://patents.google.com/patent/US7890056B2/en>
- [8] R. Want, “An Introduction to RFID Technology”, IEEE Pervasive Computing, vol. 5, no. 1, 2006, pp. 25–33.
- [9] RFID in Agriculture: A Study on Sustainable Management, AgriTech Journal, vol. 12, no. 3, 2022, pp. 45–60.
- [10] L. Dupont, Gestion automatisée des présences, Thèse de Doctorat en Informatique, Université de Lyon, 2021.
- [11] RFID Standards in Modern Logistics, Université de Cambridge, 2022.
- [12] EPCglobal Gen2 et Supply Chains, INSA Lyon, 2020.
- [13] GS1, Global RFID Frequency Regulations, GS1, 2023.
- [14] From RFID to Smart Objects, IEEE IoT Journal, 2022.
- [15] C. J. Date, An Introduction to Database Systems, 8<sup>e</sup> éd., Addison-Wesley, 2003.
- [16] Oracle, Java Platform Standard Edition Documentation, Oracle Help Center, 2023, <https://docs.oracle.com/en/java/>
- [17] J. Gosling et al., The Java Language Specification, 3<sup>e</sup> éd., Addison-Wesley, 2005.
- [18] Oracle, Java Platform, Standard Edition 8 Documentation, Oracle, 2014.
- [19] Oracle Corporation, What Is MySQL?, Oracle, 2024, <https://www.oracle.com/mysql/what-is-mysql/>

- [20] ESP32 WROOM Devkitc v4, Microsoft Bing, <https://www.iottechrends.com/getting-started-with-esp32-wroom-devkitc/>
- [21] Espressif Systems, ESP32 Series Datasheet, Version 3.4, 2023.
- [22] NXP Semiconductors, MFRC522 Standard Performance MIFARE and NTAG Frontend, Rev. 3.9, 2016, <https://www.nxp.com/docs/en/data-sheet/MFRC522.pdf>
- [23] ElectroPeak, Getting Started with LCD 16x2 Using I2C Module, 2021, <https://electropeak.com/learn/interfacing-lcd-1602-with-arduino-using-i2c/>
- [24] D. Ferkous et A. Grana, Surveillance de données biomédicales à distance, Mémoire de fin d'étude, Licence Académique, Université 8 Mai 1945 – Guelma, juin 2023.