Structure

Dépendances

La librairie fait appel à de nombreuses librairies :

- M5Unified : bibliothèque standard utilisée pour contrôler et interagir avec les M5
- M5GFX : bibliothèque graphique (normalement standard) permettant d'utiliser des fonctions d'affichage avancées afin de créer une UI
- UNIT_RFID_UHF : bibliothèque officielle de M5Stack afin de gérer le module RFID UHF
- MFRC522_I2C : bibliothèque officielle de M5Stack fin de gérer le module RFID (attention, il existe plusieurs versions de cette bilbiothèque et seule celle disponible sur le GitHub de M5Stack fonctionne)
- Wifi : bibliothèque standard permettant de contrôler la connexion WLAN du M5
- HTTPClient : bibliothèque permettant de faire des requêtes HTTP, notamment à l'API TimeTonic
- ArduinoJson : parseur JSON optimisé pour les ESP32, permet de manipulé les objets JSON retourné par l'API de TimeTonic
- Wire : bibliothèque standard permettant d'initialiser et de contrôler la connexion I2C du M5

Structure de code

La bibliothèque **M5lib** est conçue pour uniformiser le comportement de plusieurs dispositifs **M5Stack Core2** dans l'environnement du FabLab, chacun jouant un rôle défini (accueil, servante, machine, etc.). Elle repose sur une architecture orientée objet claire et modulaire, facilitant l’évolution, la maintenance et l'intégration au back-end (TimeTonic).

Structure de la Bibliothèque **M5lib**

1. **Classe de base abstraite : `M5lib`**

- Représente l’**interface** commune à tous les types de M5.

- Contient les méthodes de base utilisées par tous les modules : lecture de carte RFID (`scancard`), scan UHF (`scanuhf`), interaction utilisateur (`showchoice`, `getuser`), etc.
- Déclare des **méthodes virtuelles pures** (polymorphisme) à implémenter dans les classes dérivées : `uploadlog()` et `changestatus()`.

2. **Classes dérivées spécialisées**

Chaque type de M5 (défini dans `M5type`) possède une classe dédiée, dérivée de `M5lib` :

Classe	Hérite de	Spécificités
`accueil`	`M5lib`	Méthodes supplémentaires : `entree()`, `sortie()`, `regcard()`. Menu de motifs de visite.
`servante`	`M5lib`	Gère la traçabilité des servantes utilisées.
`machine`	`M5lib`	Pour l'enregistrement d’utilisation de machines.
`ordinateur`	`M5lib`	Associe cartes/scans à une session ordinateur précise.
`materiel`	`M5lib`	Permet le prêt/retour d’outils ou matériels via `borrow()`.

Chaque classe surcharge au moins :

- `uploadlog`: envoie les journaux d’usage à l’API TimeTonic avec une structure propre (identifiants de champs différents selon le type).
- `changestatus`: placeholder actuel (pour la plupart renvoie `200`) servant potentiellement à signaler un changement d’état matériel (ex : verrouillage machine, etc.).

3. **Composants globaux manipulés**

- **Périphériques** :
  - `MFRC522 mfrc522` : pour la lecture RFID classique (Mifare).
  - `Unit_UHF_RFID uhf` : pour les lectures UHF/EPC.
- **API & Données** :
  - `HTTPClient http` : communication HTTP avec TimeTonic.
  - `JsonDocument doc` : stockage de la réponse JSON.
- **Accès réseau** :
  - Configuration du Wi-Fi intégrée dans `setupstd()`.

Résumé Visuel Simplifié

```
      +--------------------------+
      |         M5lib            |  <--- classe abstraite
      +--------------------------+
   /      |      |      |       \
  /       |      |      |        \
accueil   servante  machine ordinateur  materiel  <--- classes concrètes

Toutes implémentent :
   + uploadlog()
   + changestatus()

Chaque classe possède des méthodes propres, par exemple :
accueil fournit aussi :
   + entree()
   + sortie()
   + regcard()
```

Une **évolution fluide** : il suffit d’ajouter une nouvelle classe dérivée et d’enrichir la factory pour intégrer un nouveau comportement ou un nouveau rôle matériel dans le système du FabLab.

Revision #1
Created 21 July 2025 12:39:16 by Eyglier De Rooster Mathieu
Updated 21 July 2025 13:05:27 by Eyglier De Rooster Mathieu