

PFE 024 - Systèmes de tests IoT

Par

Armand Dim Dim
Maximilien Blanchard-Bizien

Professeure: Moha, Naouel



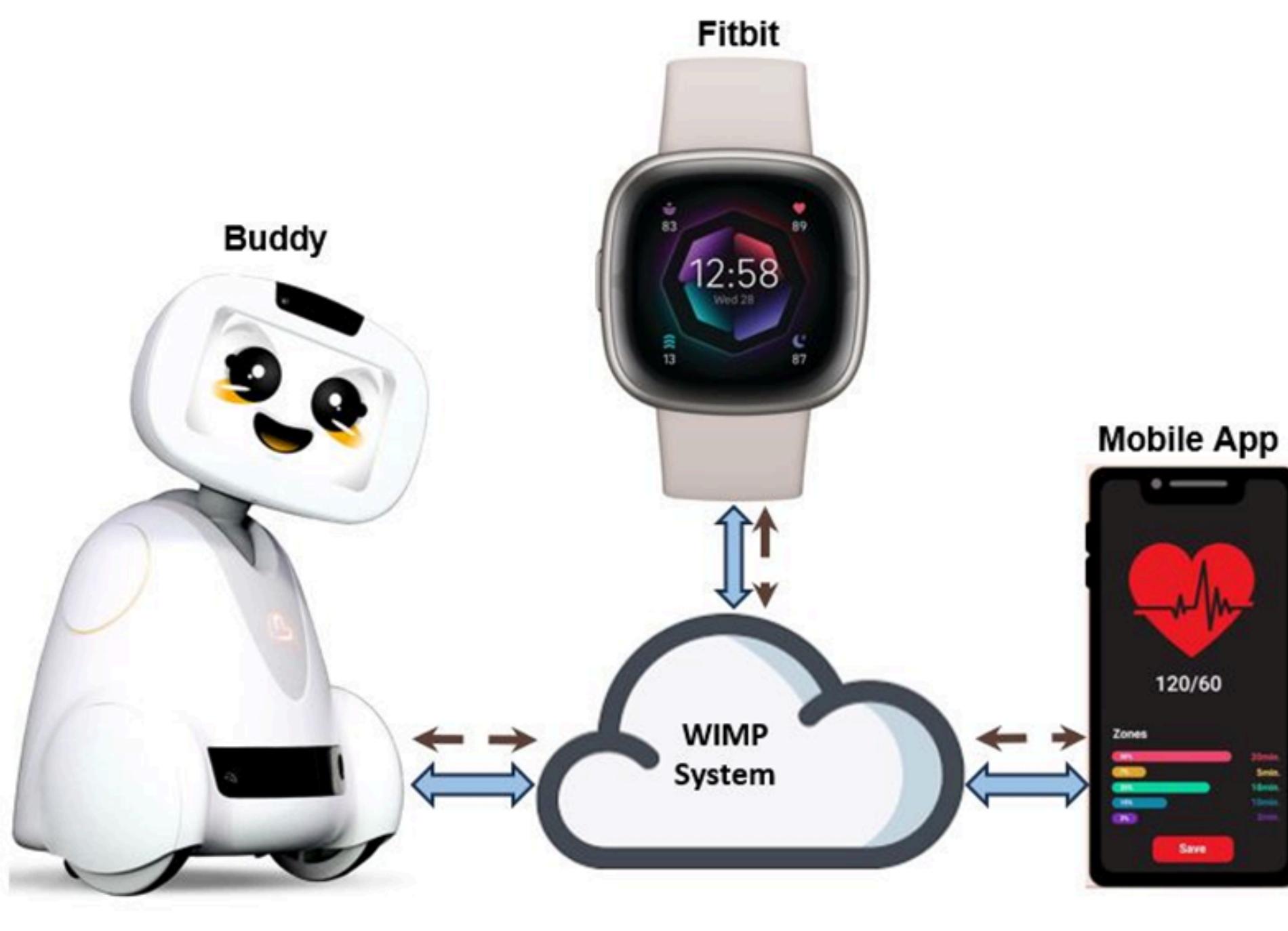
Agenda

- | | | | |
|---|---|----|-------------------------------|
| 1 | Présentation de WIMP | 6 | Modélisation |
| 2 | Équipements IoT : Buddy robot et Fitbit | 7 | Description des microservices |
| 3 | Contexte, problématique et Objectifs | 8 | Résultats et améliorations |
| 4 | Plan de gestion du projet | 9 | Défis |
| 5 | Analyse et conception | 10 | Conclusion |

WIMP (Application)

WIMP

WIMP



Développé par l'équipe Ptidej de Concordia

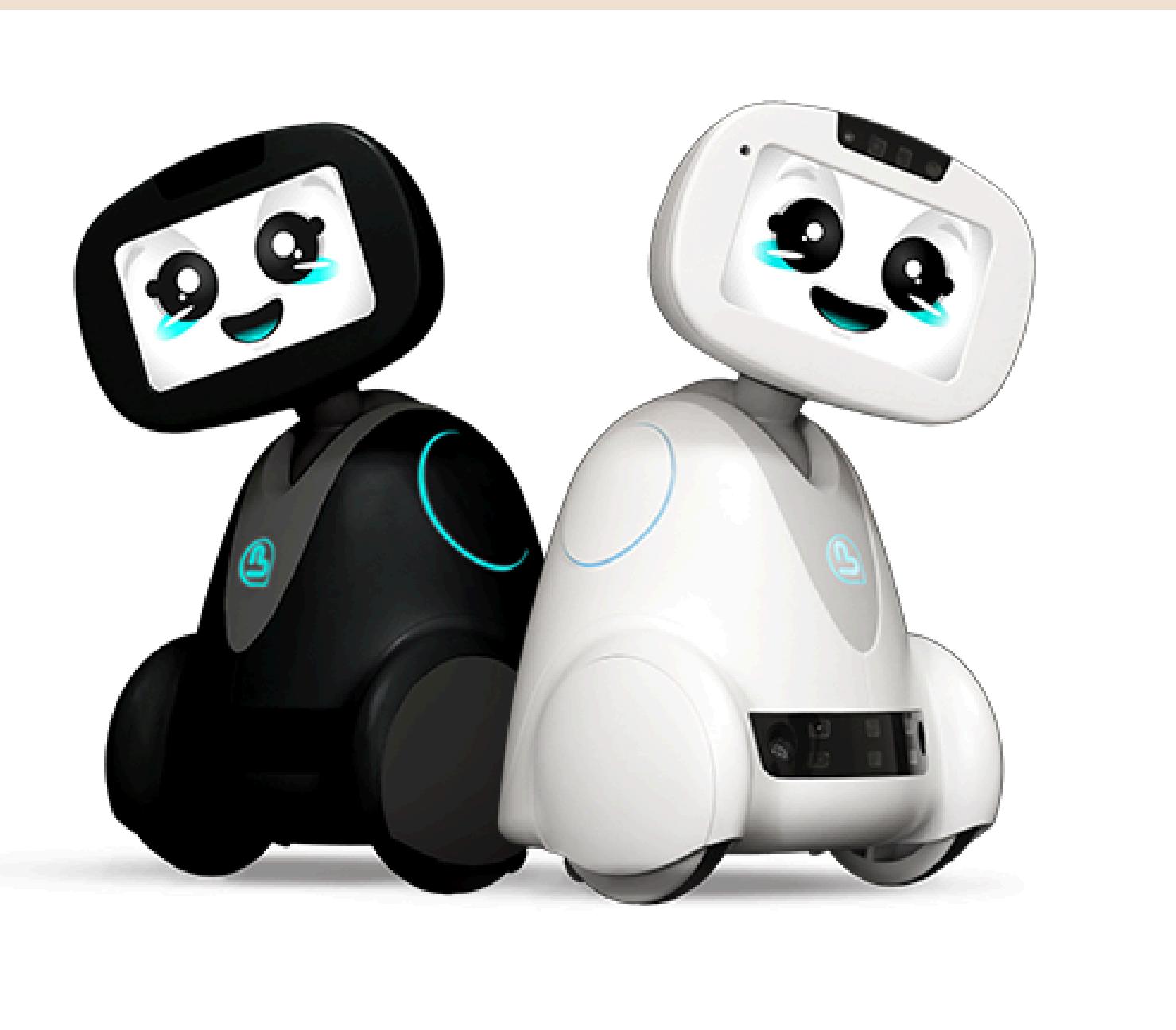


Utilise robot Buddy pour déterminer la disponibilité de l'enseignant



Utilise aussi les données de la montre Fitbit

Equipements IoT: Robot Buddy



E Enablewheels

T Turn

M Move

C Change

R Rotate

sT Stop

S Speak

Mood

Le Blue Frog Robotics Buddy Pro pour Développeurs (SDK) est un robot compagnon innovant offrant interaction, éducation, soins, soutien émotionnel et connectivité. Sert de plateforme ouverte et évolutive pour les développeurs.

Equipements IoT : Fitbit



Montre intelligente développée par Fitbit



Monitore fonctions du corps



Utiliser pour déterminer actions à prendre

Fitbit fournit des insights précieux basés sur les données collectées par ses dispositifs portables. le cas ici avec buddy robot. Par exemple, WIMP peut aller chercher le battement de coeur du professeur, et déterminer si le robot buddy se déplacer, ou déplacer sa tête.

Contexte



Générer des cas de tests pour une application IoT:

- ➊ Tests des dispositifs IoT complexes
- ➋ Nécessite une approche multi-dimensionnelle
- ➌ Dispositif IoT pas toujours adapté
- ➍ Connexion nécessaire pour intéragir avec outils

Problématique



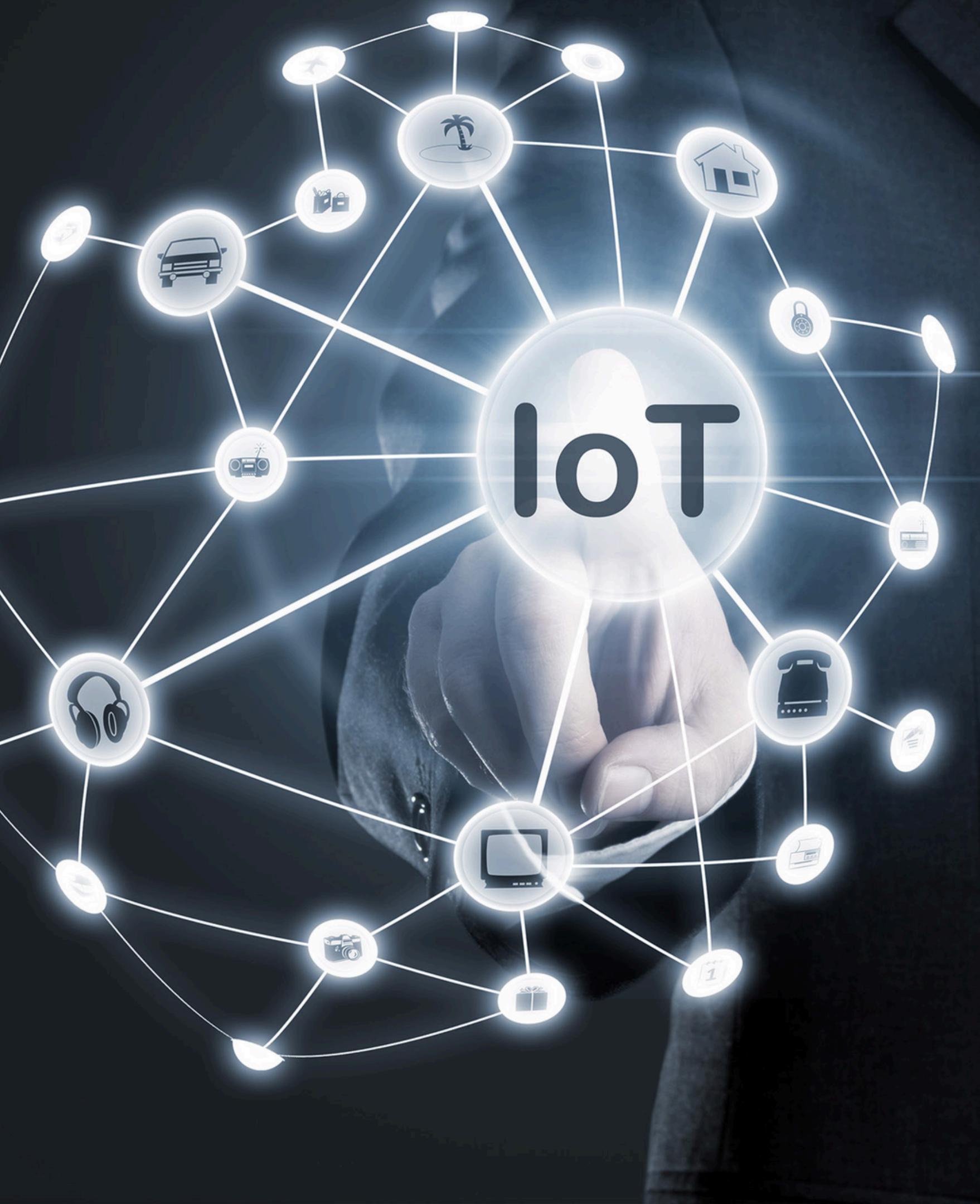
Comment tester et gérer efficacement des tests pour assurer le fonctionnement de l'application WIMP et son système IoT pour la gestion de la disponibilité des professeurs?



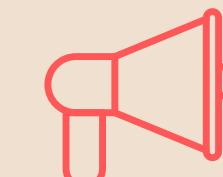
Approche

- ▶ Mettre en place différents types de protocoles pour diverses applications IoT
- ▶ L'envoie des requêtes et la gestion des réponses, y compris les erreurs
- ▶ L'analyse et le stockage des résultats pour un usage ultérieur

Objectifs



Développer un outil de test automatisé



Analyser le payload

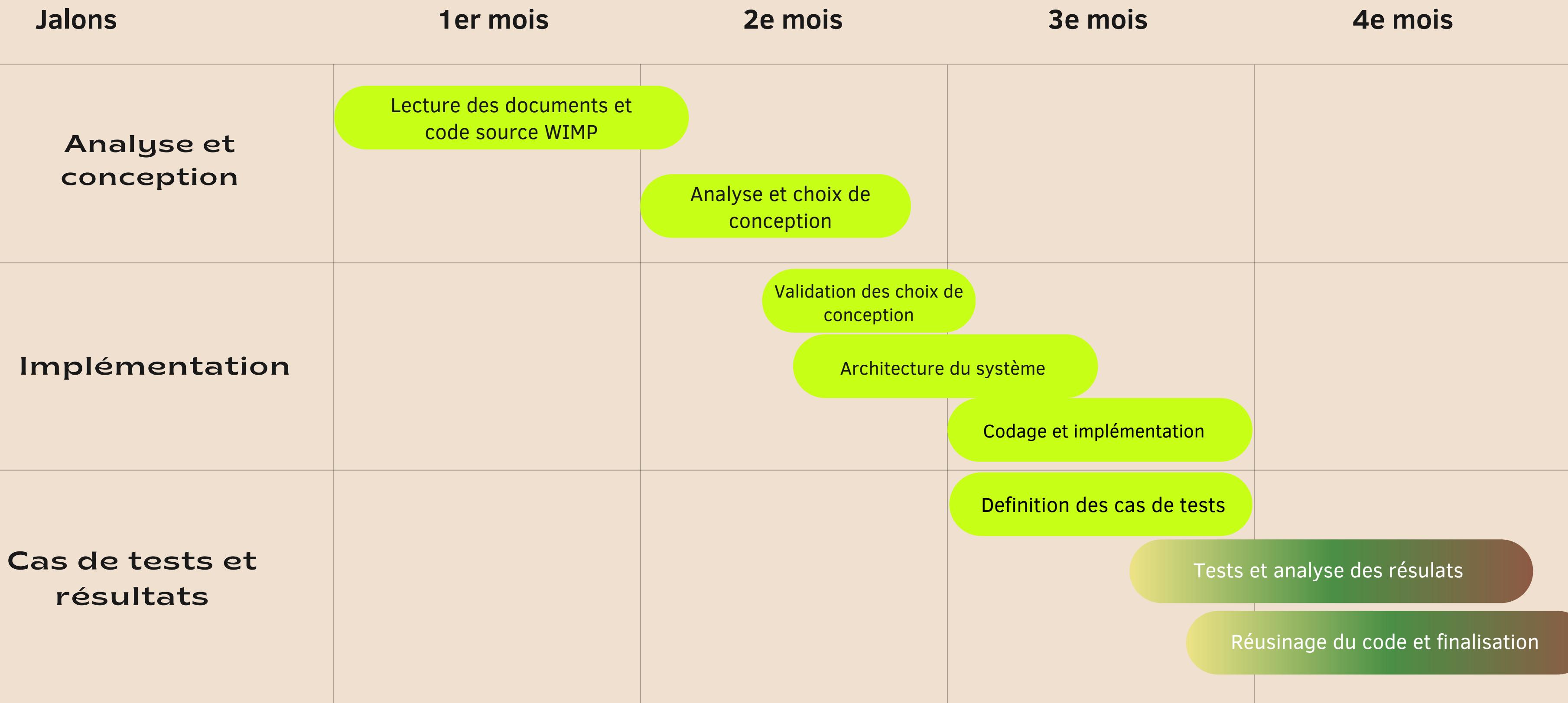


Ajouter des protocoles de communication

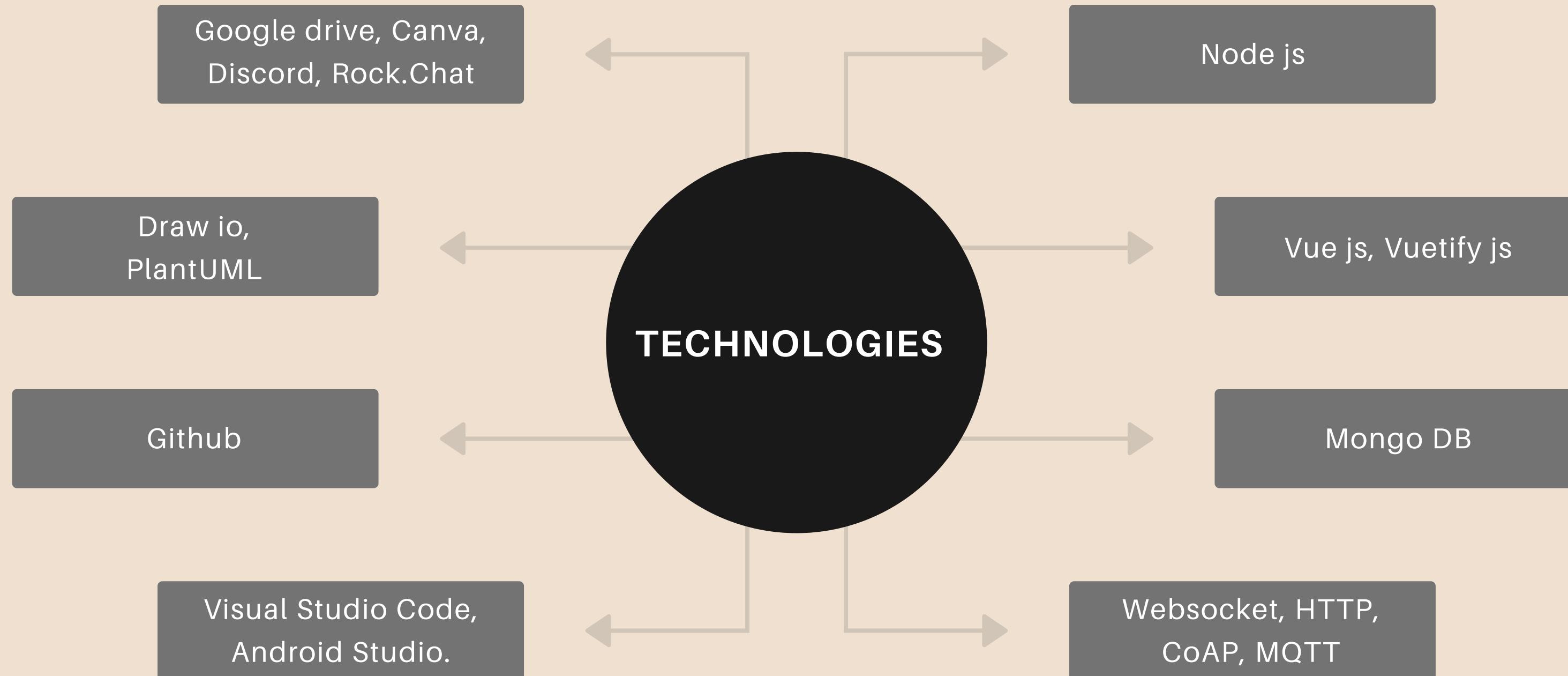


Exécuter des tests et analyser les résultats

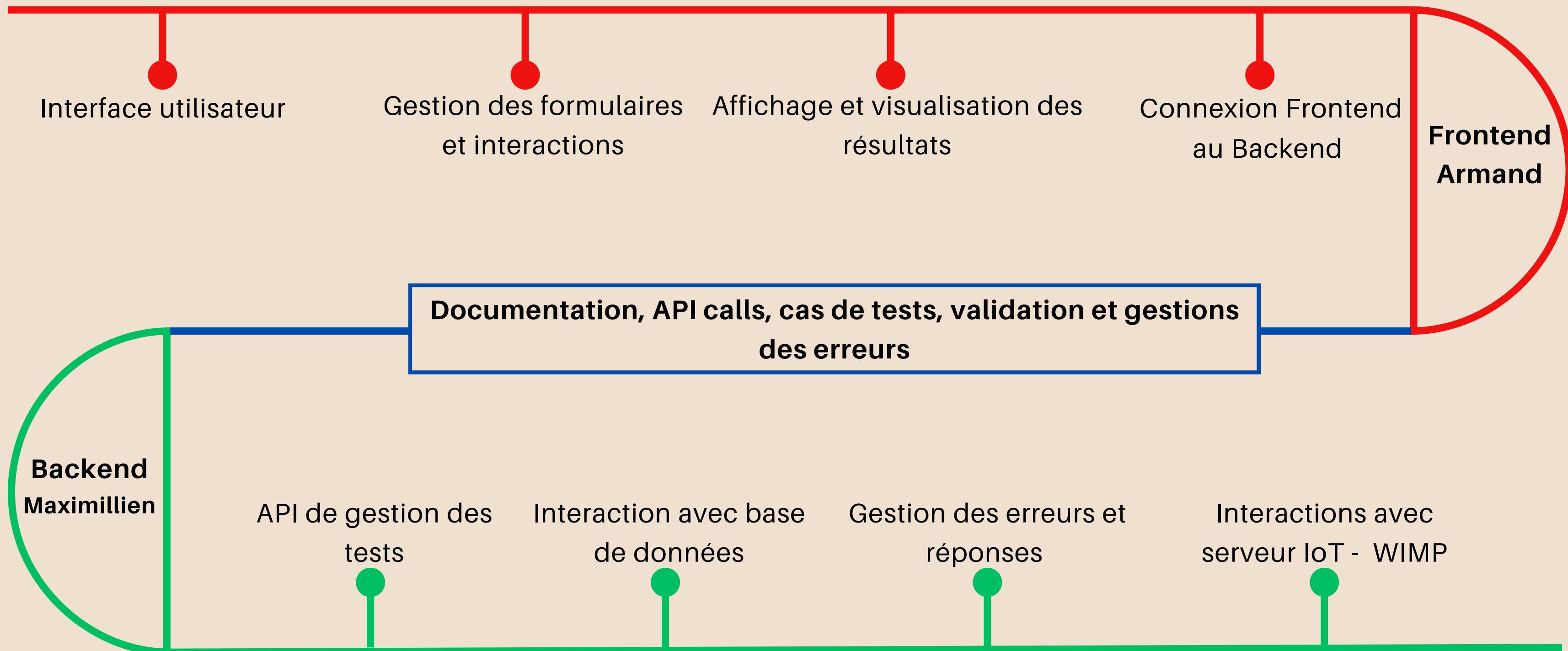
Étapes du projet



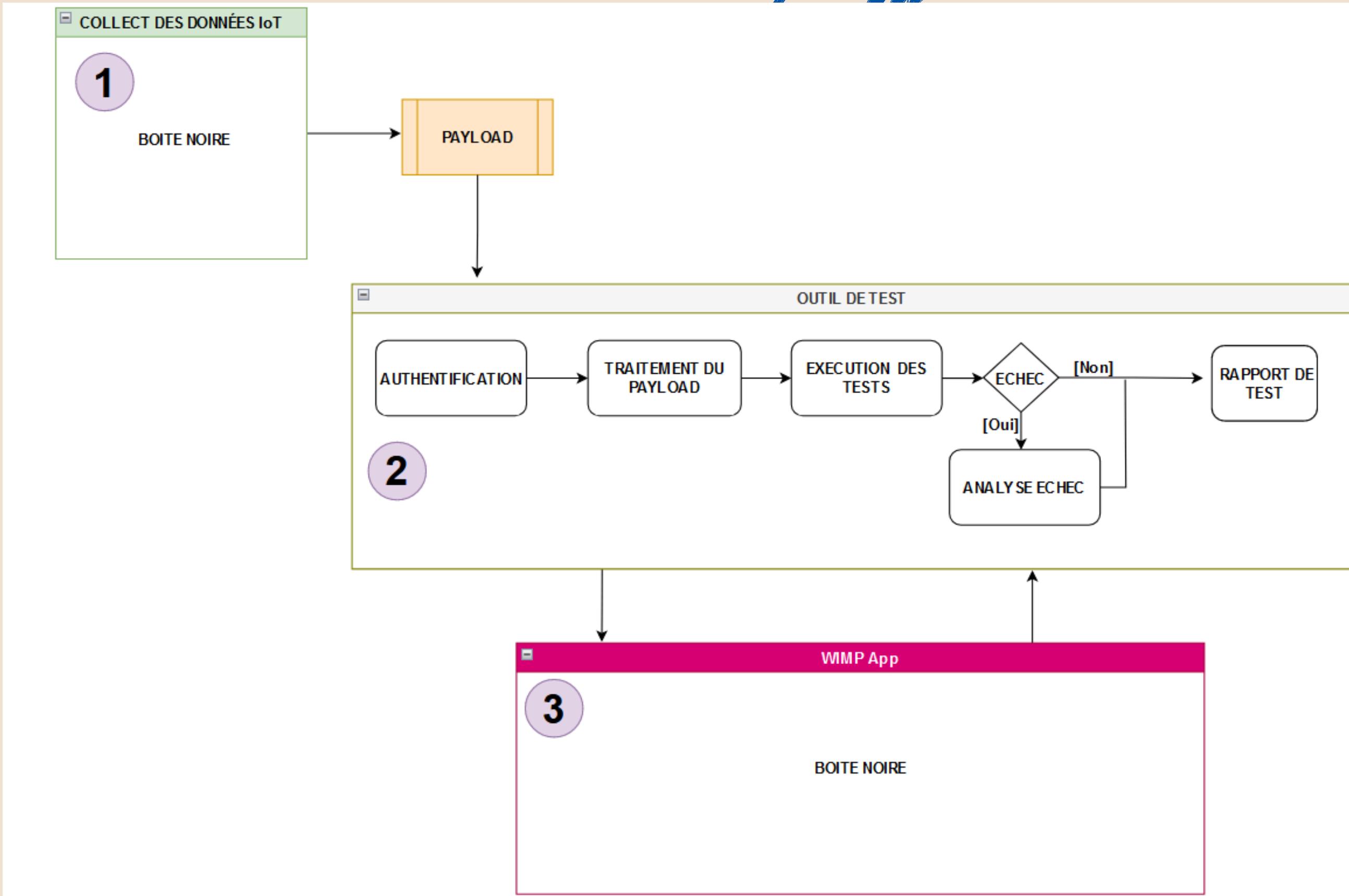
Technologies



Séparation des tâches



Analyse et conception



Description des composants

1. Génération du payload
(Collecte des données des équipement IoT)
2. Outil de test vue en processus
3. WIMP App

Payloads

```
"TC_ID": "TC001",
"name": "move buddy successfully",
"steps": [
{
  "operation": "getData",
  "target": {
    "protocol": "HTTP",
    "method": "POST",
    "name": "Fitbit"
  },
  "inputs": {
    "datatype": "bpm"
  },
  "expectations": {
    "msg": "Receive Fitbit Data"
  }
},
{
  "operation": "determineBuddy",
  "target": null,
  "inputs": {
    "data": 130
  },
  "expectations": {
    "msg": "robot must move"
  }
},
{
  "operation": "Enable_Wheels",
  "target": {
    "protocol": "Websocket",
    "method": "send",
    "name": "Buddy"
  },
  "inputs": {
    "left": 1,
    "right": 1
  },
  "expectations": {
    "msg": "WHEEL_MOVE_FINISHED"
  }
}
```

Un payload est un ensemble de données transmises au sein d'une requête ou d'un message entre différents systèmes ou applications.



Contenue dans un fichier JSON

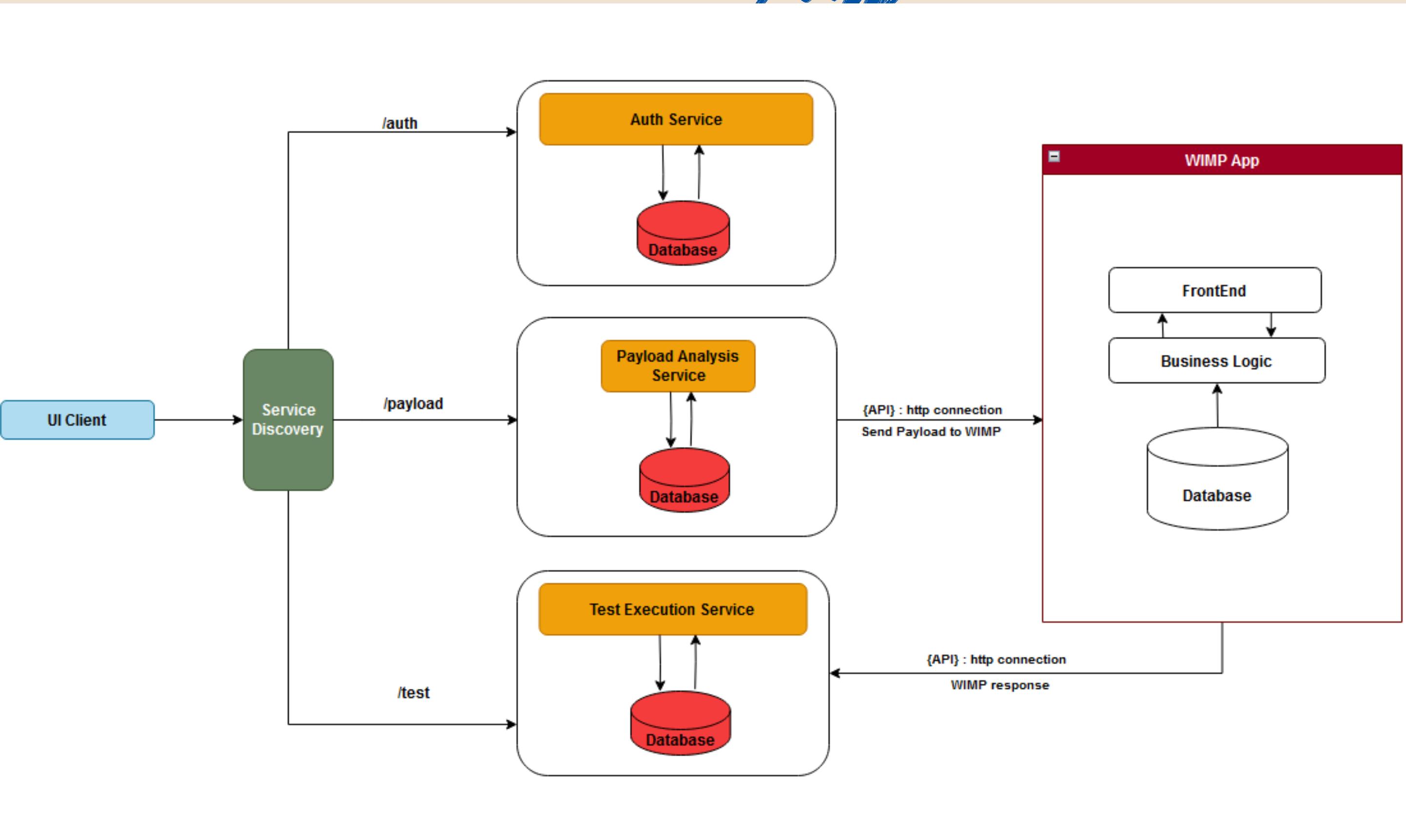


Envoyé à WIMP, qui traite les données



Test case pour l'application

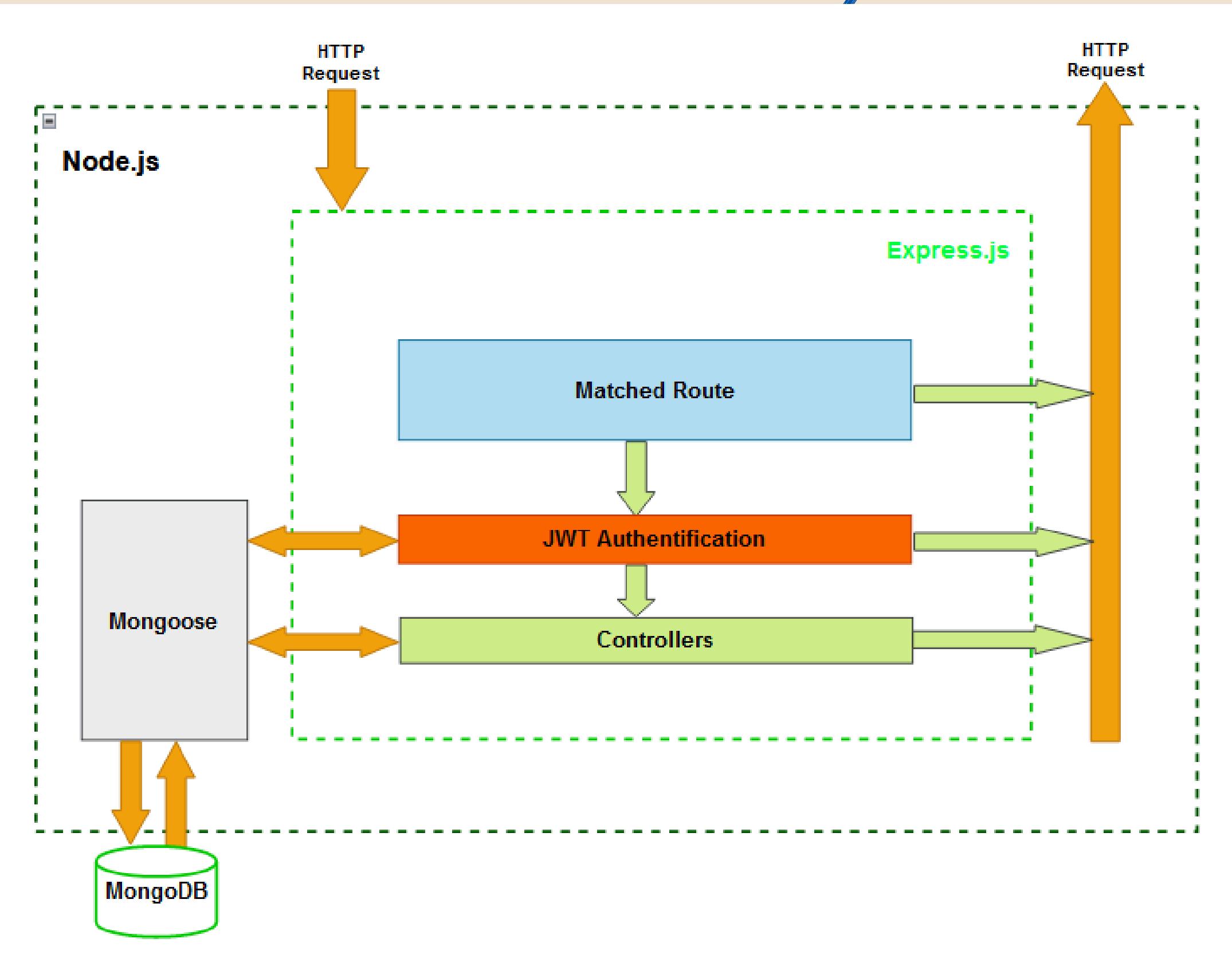
Architecture du système



Architecture basée microservice

- Interface client
- Service d'authentification
- Service d'analyse de payload
- Service d'exécution des tests
- Service registry

Architecture du système (suite)



Fonctionnement d'un microservice

Node.js

- Le cadre principal qui exécute l'application
- Reçoit les requêtes HTTP entrantes

Express.js

- Framework web pour Node.js
- Gère des requêtes et des routes

Route

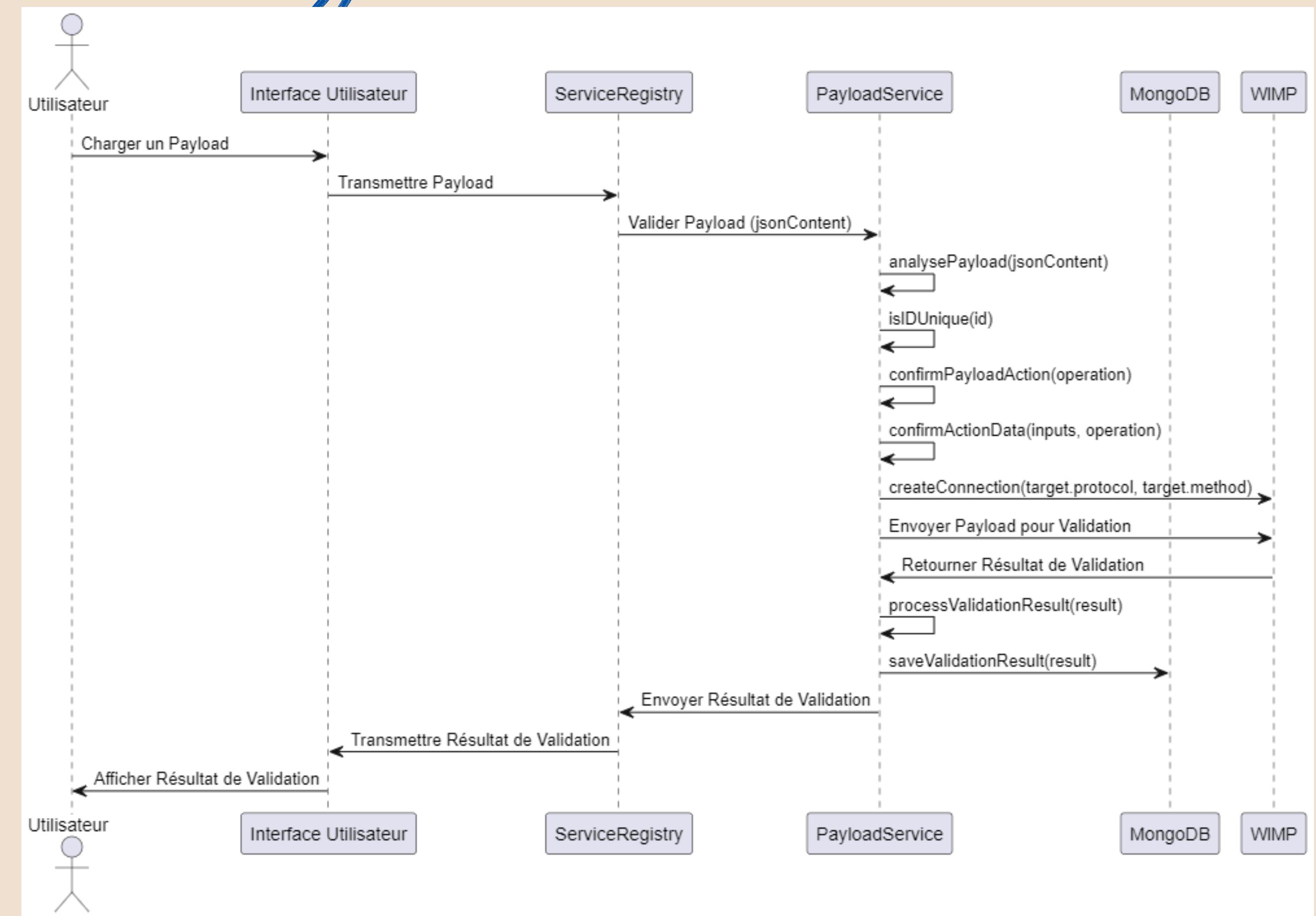
- Express.js fait correspondre les requêtes HTTP avec les routes définies

Mongoose

- ODM (Object Data Modeling) pour MongoDB et Node.js
- Interagie avec MongoDB pour les opérations de base de données

Modélisation

Diagramme de séquence envoie et exécution et sauvegarde du payload



Modélisation (suite)

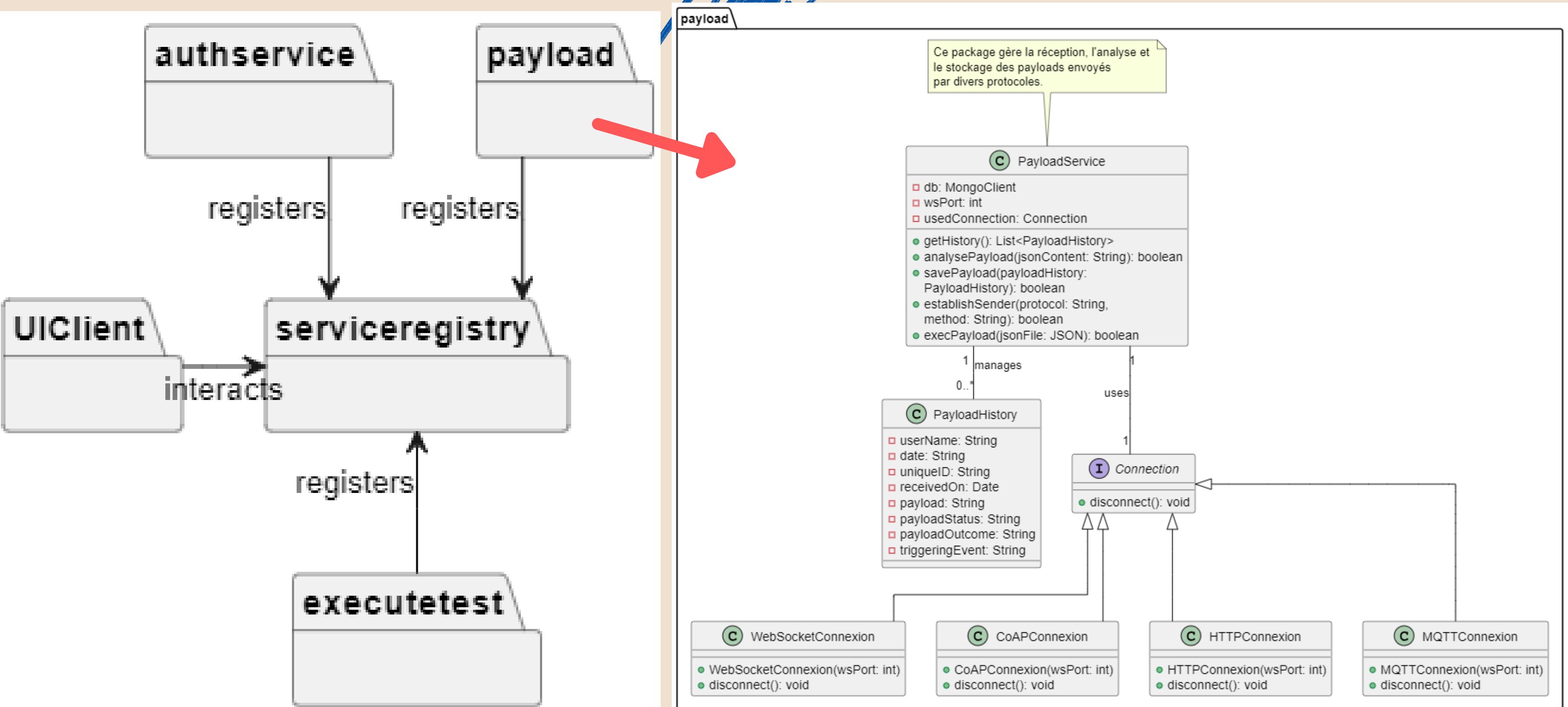
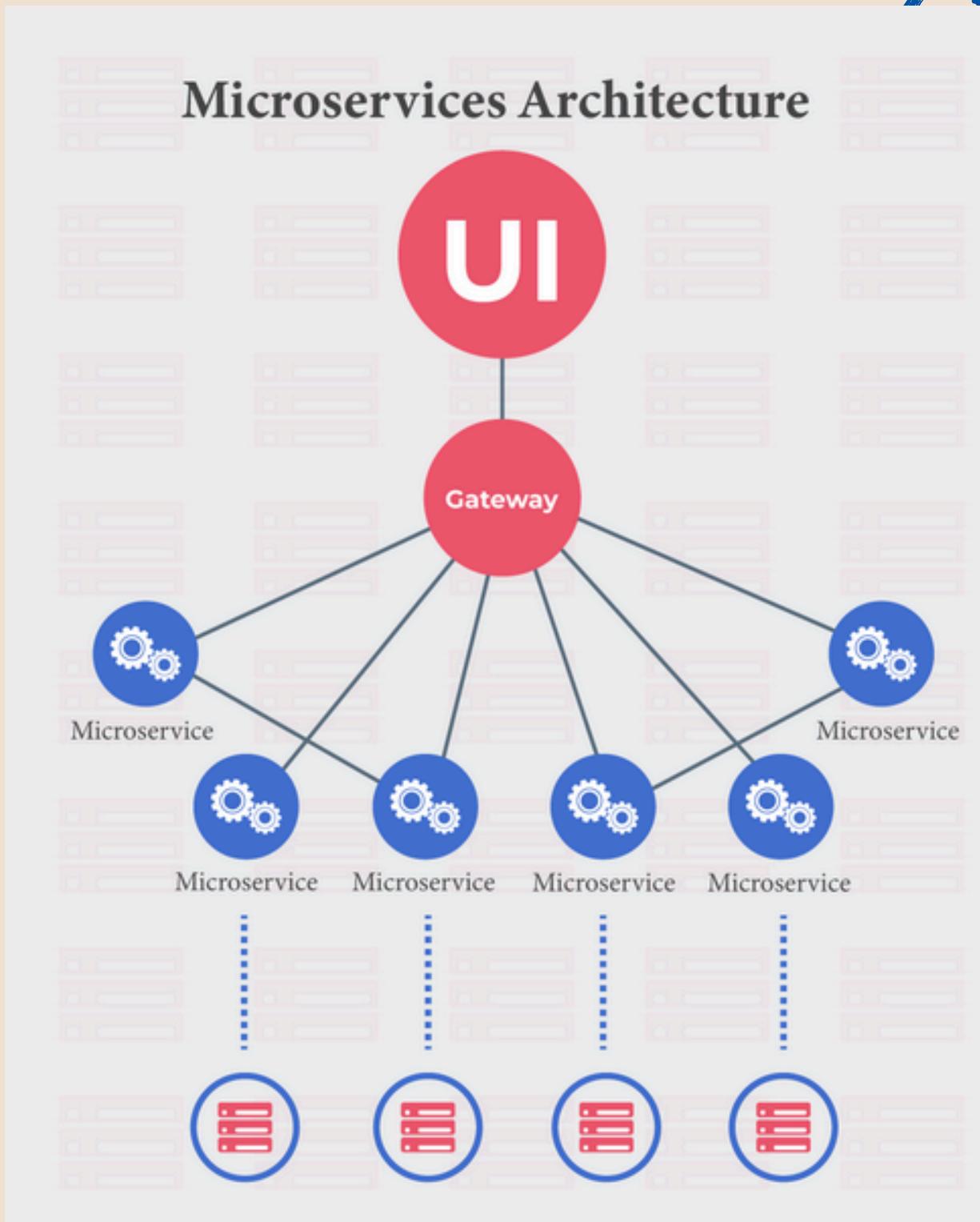


Diagramme de package système et diagramme de classe du microservice
Payload

Description des microservices



- Service d'authentification
- Service d'analyse du payload
- Service d'exécution des tests

Service d'authentification



Login and Test Your System

Ensure your email for registration

Email: frq@gmail.com

Password: ...

Forgot your password ?

SIGN IN

TestIoT TaaS
Functional Testing of Your IoT System

SIGN UP

- Gérer connexion et utilisateurs
- Ajouter de nouveaux utilisateurs
- Encrypter mots de passes
- Générer les jetons

Service de gestion du payload



The screenshot shows a user interface for managing test cases. At the top, there are tabs: 'COLLECT', 'PAYLOAD', 'TEST', and 'RESULT'. The 'PAYLOAD' tab is active, indicated by a blue circle with the number '2' and a red downward arrow icon. Below the tabs, a progress bar indicates the current step: 'Information Collection' (step 1), 'Payload Selection' (step 2), 'Running Test' (step 3), and 'Test Results' (step 4). A message 'Wait while the payload is being processed...' is displayed above a large text area. The text area contains a JSON payload configuration:

```
Process Payload
{
  "TC_ID": "TC002",
  "name": "move buddy successfully",
  "steps": [
    {
      "operation": "getData",
      "target": {
        "protocol": "HTTP",
        "method": "POST",
        "name": "Fitbit"
      },
      "inputs": {
        "datatype": "bpm"
      },
      "expectations": {}
    }
  ]
}
```

At the bottom of the interface are two buttons: 'SAVE & CONTINUE' and 'BACK'.

- Analyser le payload
- Envoyer le payload application IoT
- Sauvegarder les informations dans la base de données

Service des tests



- Gérer tests sauvegardées
- Obtenir résultats de l'application IoT
- Retourner les résultats à utilisateur

Résultat des tests

The screenshot shows a software interface for test configuration, specifically for a payload process. The top navigation bar includes 'COLLECT', 'PAYLOAD', 'TEST', 'RESULT', 'LANGUAGE' (set to ENGLISH), and a help icon.

The main area displays a four-step process:

- Information Collection (Step 1)
- Payload Selection (Step 2)
- Running Test (Step 3)
- Test Results (Step 4)

A progress message "Wait while the payload is being processed..." is visible above the payload details.

The payload configuration section, titled "Process Payload", contains the following JSON code:

```
{  
  "TC_ID": "TC024",  
  "name": "move buddy successfully",  
  "steps": [  
    {  
      "operation": "getdata",  
      "target": {  
        "protocol": "HTTP",  
        "method": "POST",  
        "name": "Fitbit"  
      },  
      "inputs": {  
        "datatype": "bpm"  
      },  
      "expectations": {  
        "msg": "Receive Fitbit Data"  
      }  
    },  
    {  
      "operation": "determineBuddy",  
      "target": null,  
      "inputs": {  
        "data": 130  
      },  
      "expectations": {  
        "msg": "robot must move"  
      }  
    },  
    {  
      "operation": "Enable_Wheels",  
      "target": null,  
      "inputs": {}  
    }  
  ]  
}
```

At the bottom, there are "SAVE & CONTINUE" and "BACK" buttons.

Résultat des tests

The screenshot shows a user interface for managing test cases. At the top, there is a navigation bar with tabs: COLLECT, PAYLOAD, TEST, and RESULT. On the right side of the header, there are buttons for LANGUAGE (set to ENGLISH) and a refresh icon.

The main area displays four status steps:

- Information Collection (Step 1, green checkmark)
- Payload Selection (Step 2, green checkmark)
- Running Test (Step 3, blue play button)
- Test Results (Step 4, green chart icon)

Below these steps, there is a section labeled "Test ID" with a text input field containing "Enter TC_ID".

At the bottom left, there are two buttons: "RESULT" and "BACK".

A modal window titled "Payload Analysis Result" is displayed in the center. It contains the following text:

Starting analysis of payload "move buddy successfully".
Created HTTP connection.
Created Websocket connection.
Created Websocket connection.
Added payload getData to history database
Added payload determineBuddy to history database
Added payload Enable_Wheels to history database
Added payload move to history database
Added payloads successfully

At the bottom right of the modal, there is a "CLOSE" button.

At the very bottom of the page, there is a red footer bar with the copyright text: "© 2024. Ptidej Team @ Concordia University. All Rights Reserved."

Résultat des tests

COLLECT PAYLOAD TEST RESULT LANGUAGE ENGLISH ▾

Information Collection Payload Selection Running Test Test Results

DOWNLOAD REPORT

Summary

Total Tests : 7
Passed: 5
Failed: 2

Test Results Graph

The graph displays the distribution of test results. The y-axis represents the number of tests from 0 to 5.0. The x-axis shows two categories: 'Passed' and 'Failed'. A light blue bar reaches the 5.0 mark on the y-axis under 'Passed', indicating 5 successful tests. A pink bar reaches the 2.0 mark on the y-axis under 'Failed', indicating 2 failed tests.

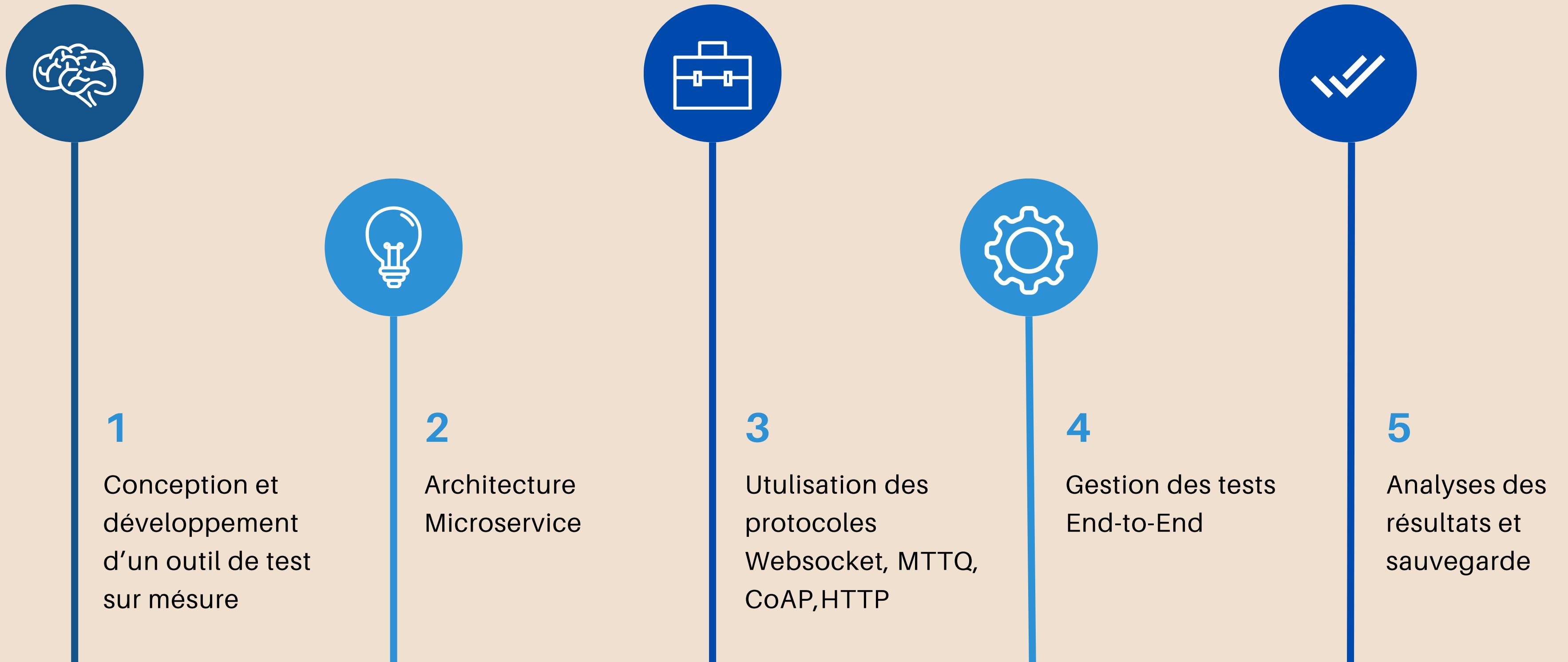
Detailed Results

Search

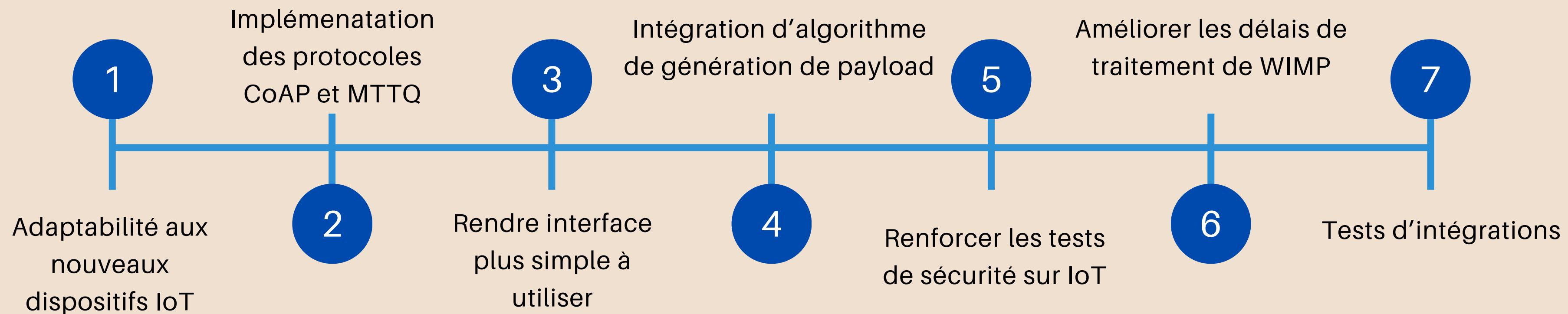
Test Case ID	Operation	Target	Inputs	Received On	Message	Actual Result	Status	Indicator
TC002	Enable_Wheels	{ "protocol": "WebSocket", "method": "send", "name": "Buddy" }	{"left": 1, "right": 1}	2024-08-07	WHEELS_ENABLE_FINISHED	WHEELS_ENABLE_FINISHED	Success	Green
TC002	Enable_Wheels	{ "protocol": "WebSocket", "method": "send", "name": "Buddy" }	{"left": 1, "right": 1}	2024-08-07	WHEELS_ENABLE	WHEELS_ENABLE_FINISHED	Fail	Red
TC002	Enable_Wheels	{ "protocol": "WebSocket", "method": "send", "name": "Buddy" }	{"left": 1, "right": 1}	2024-08-07	WHEELS_ENABLE_FINISHED	WHEELS_ENABLE_FINISHED	Success	Green
TC002	Enable_Wheels	{ "protocol": "WebSocket", "method": "send", "name": "Buddy" }	{"left": 1, "right": 1}	2024-08-07	WHEELS_ENABLE_FINISHED	WHEELS_ENABLE_FINISHED	Success	Green
TC002	Enable_Wheels	{ "protocol": "WebSocket", "method": "send", "name": "Buddy" }	{"left": 1, "right": 1}	2024-08-07	WHEELS_ENABLE	WHEELS_ENABLE_FINISHED	Fail	Red
TC002	Move	{ "protocol": "WebSocket", "method": "send", "name": "Buddy" }	{"distance": 2, "speed": 1}	2024-08-11	WHEEL_MOVE_FINISHED	WHEEL_MOVE_FINISHED	Success	Green
TC002	Move	{ "protocol": "WebSocket", "method": "send", "name": "Buddy" }	{"distance": 3, "speed": 1}	2024-08-11	WHEEL_MOVE_FINISHED	WHEEL_MOVE_FINISHED	Success	Green

Rows per page: 10 ▾ 1-7 of 7 < > 24

Solutions



Améliorations et Évolutions



Défis



MICROSERVICE D'AUTHENTIFICATION

Les mots de passes sont maintenant hachés.

FRONTEND

Interface implémenté. Certains détails mineurs manquants.

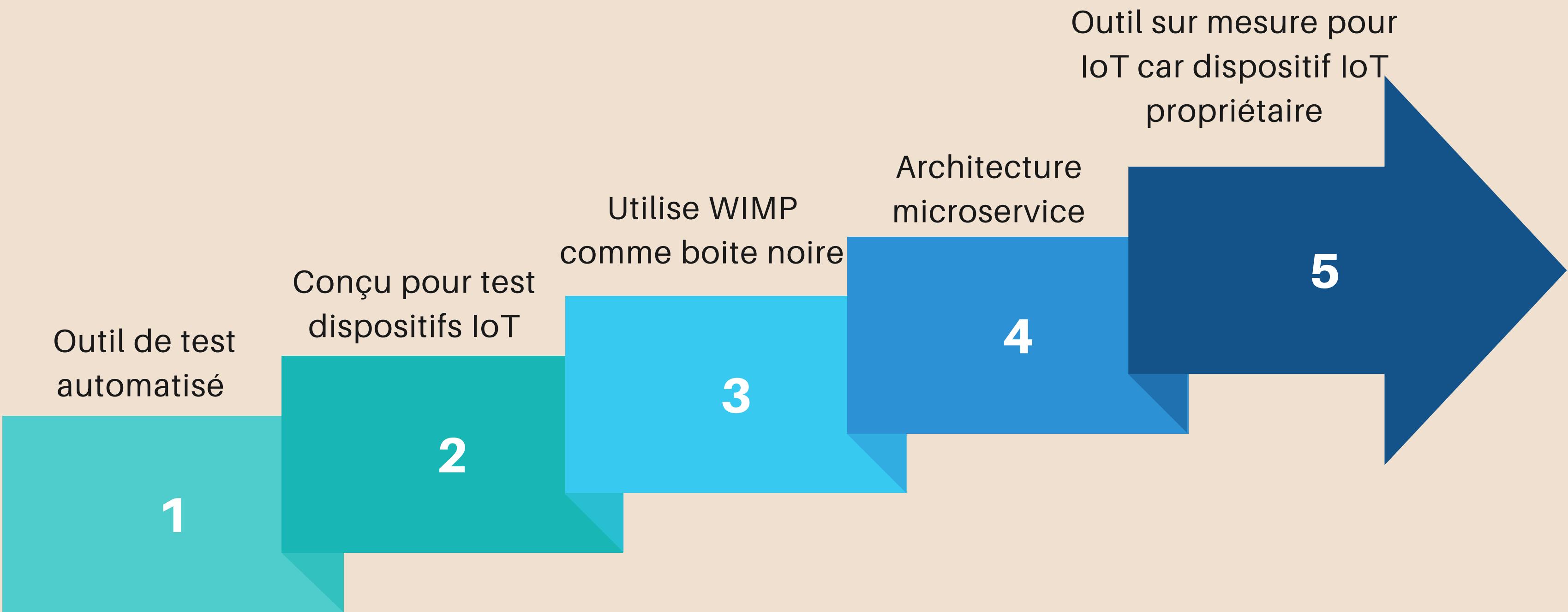
ANALYSE DU PAYLOAD

Autres connexions ont été implémentés, mais manque des application de tests.

RÉSEAU IOT-3

Problèmes de connexions avec WIMP.

Conclusion



Questions



?