ÉTÉ 2024

PFE 024 - Systèmes de tests IoT

Par

Armand Dim Dim
Maximilien Blanchard-Bizien

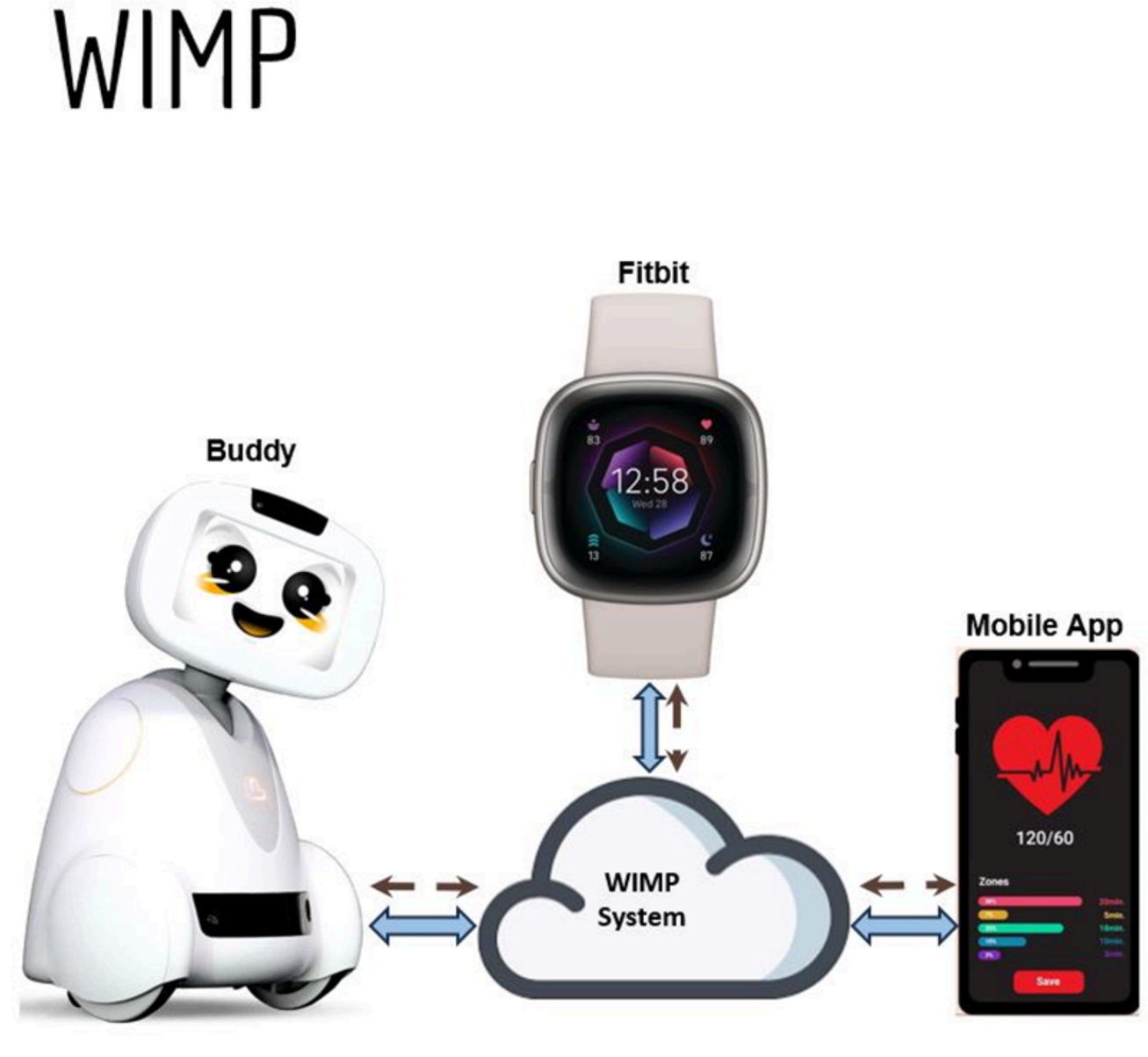
Professeure: Moha, Naouel

ÉTS

Agenda

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

WIMP (Application)

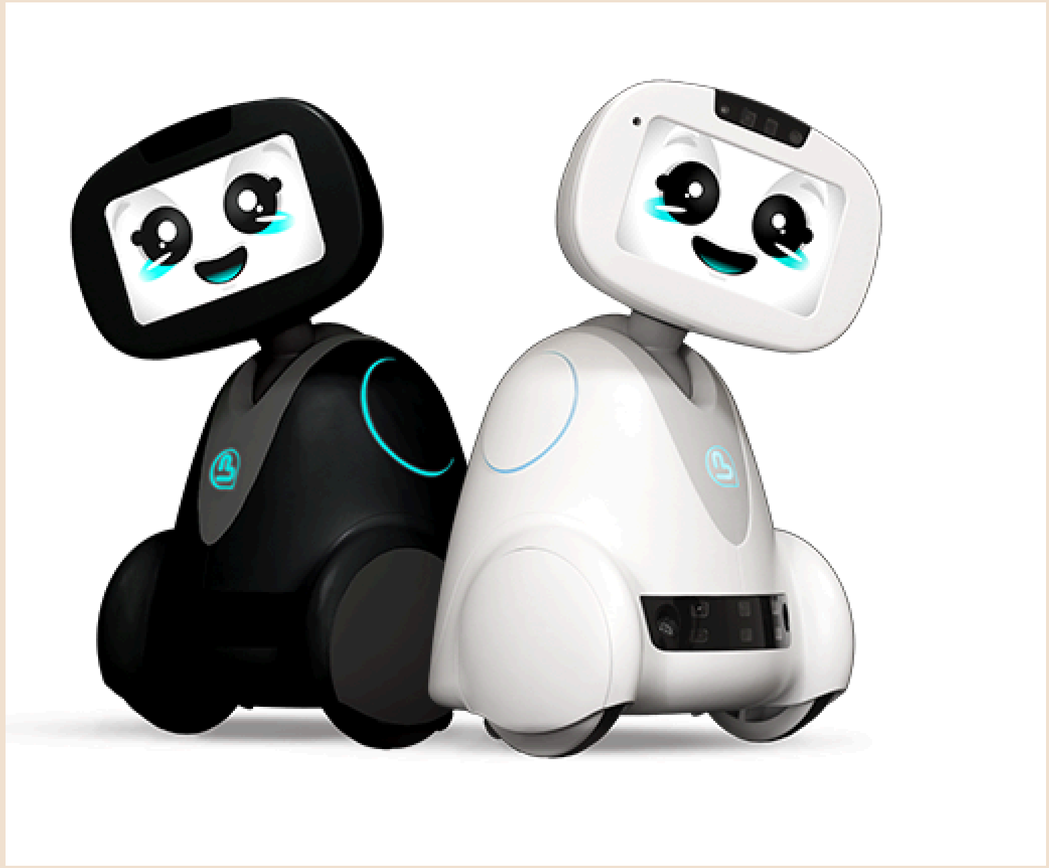


Développé par l’équipe Ptidej de Concordia

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 développé 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:

- Tests des dispositifs IoT complexes
- Nécessite une approche multi-dimensionelle
- Dispositif IoT pas toujours adapté
- 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

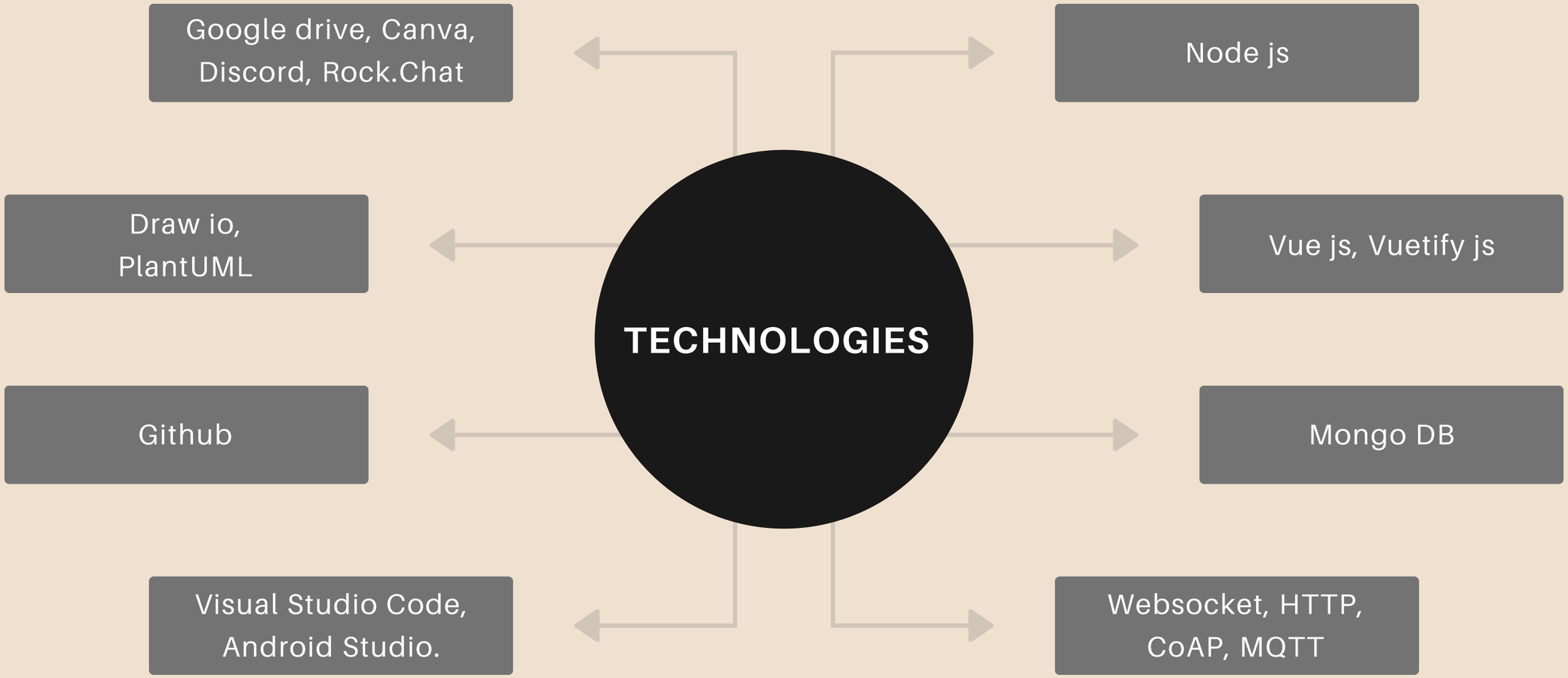
- Mettre en place différents types de protocoles pour diverses applications IoT
- L’envoie des requêtes et la gestion des réponses, y compris les erreurs
- L’analyse et le stockage des résultats pour un usage ultérieur

Objectifs

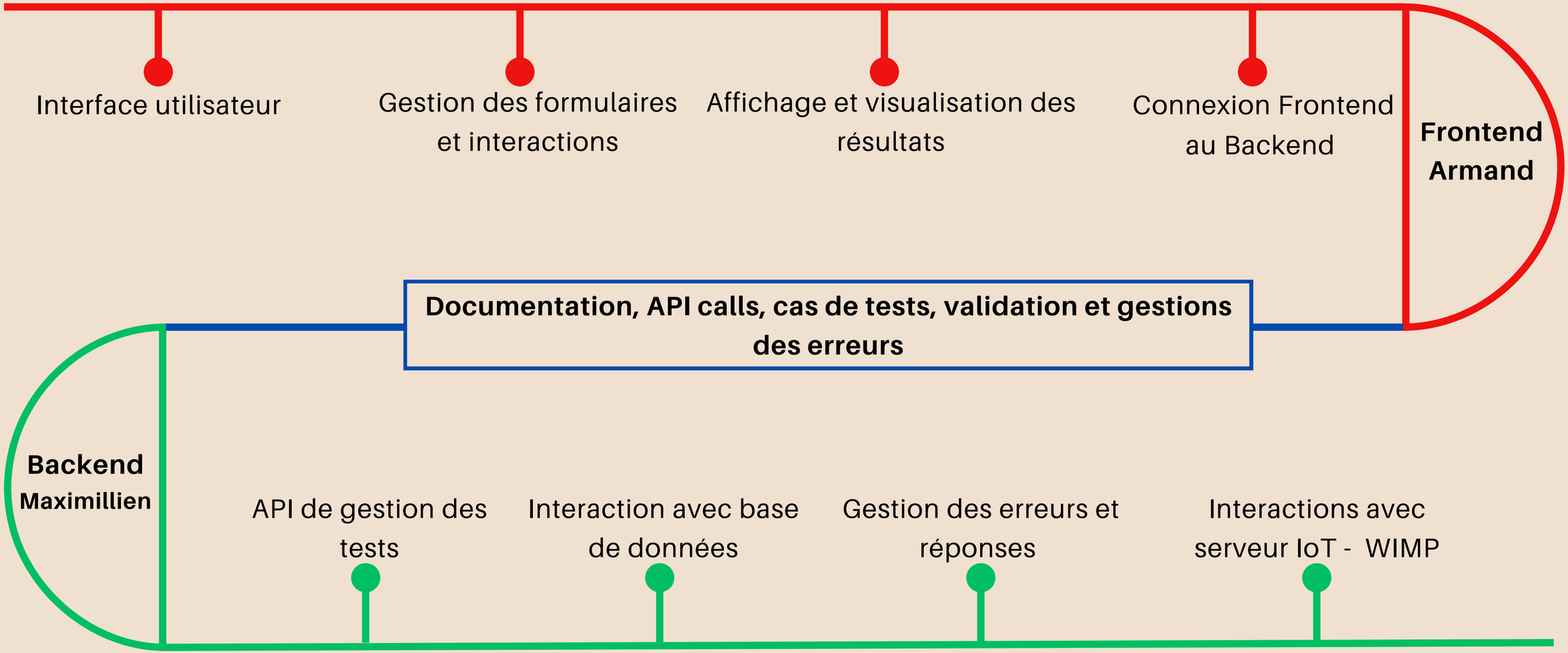
- Développer un outil de test automatisé
- Analyser le payload
- Ajouter des protocoles de communication
- Exécuter des tests et analyser les résultats

Étapes du projet

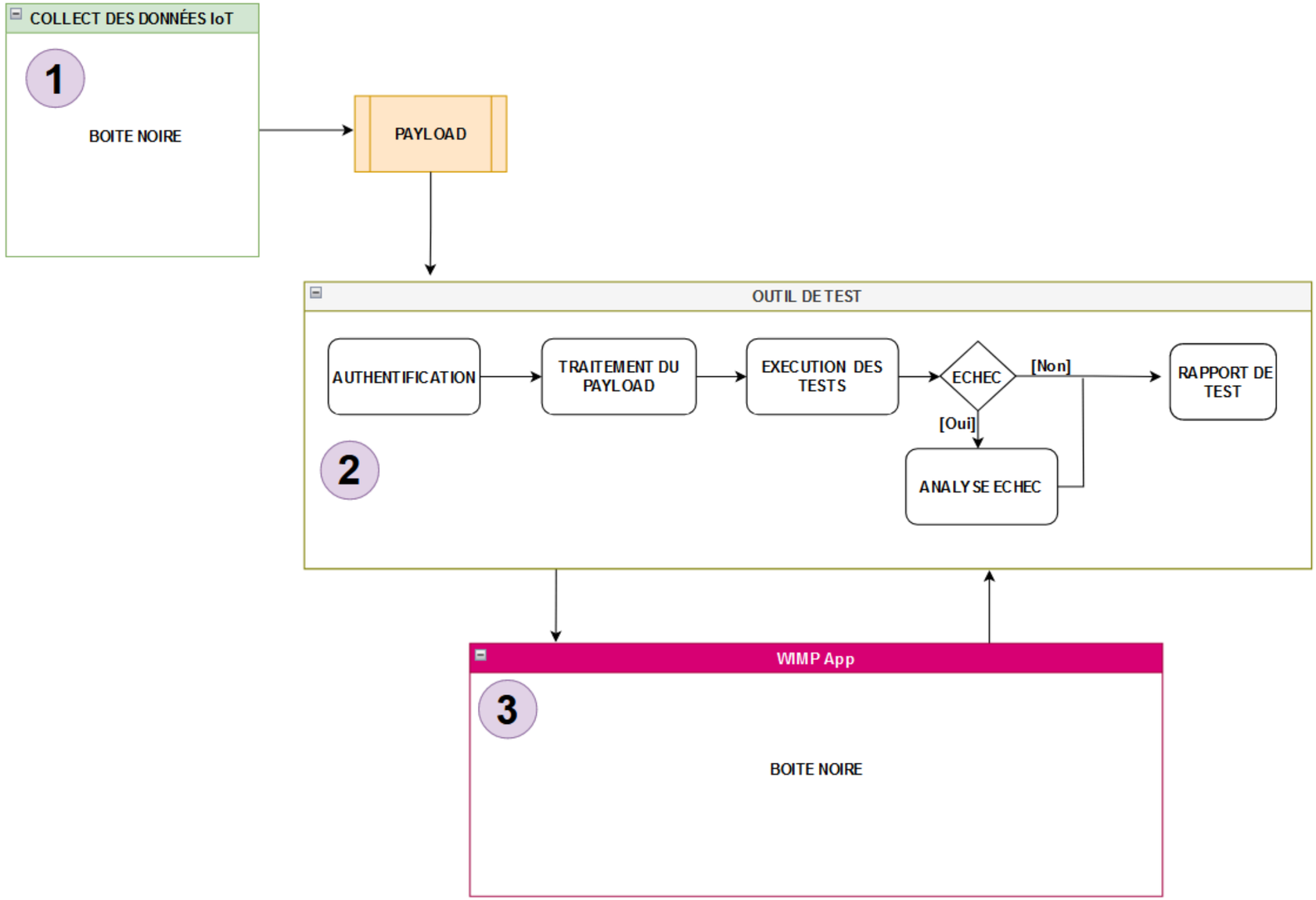
Technologies



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"
      }
```

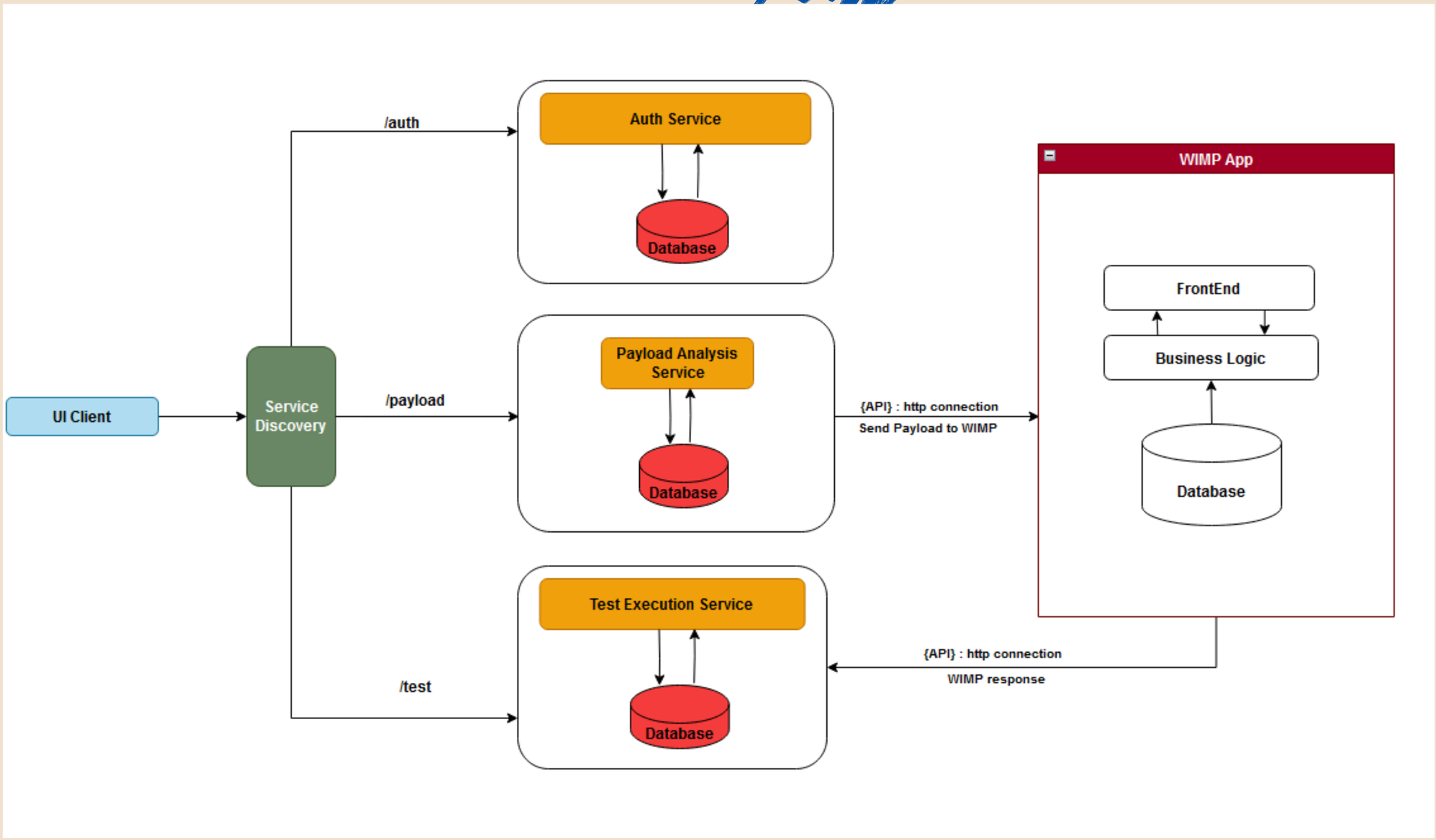

Un payload est un ensemble de données transmises au sein d'une requête ou d'un message entre différents systèmes ou applications.

Contenue dans un fichier JSON

Envoyé à WIMP, qui traite les données

Test case pour l’application

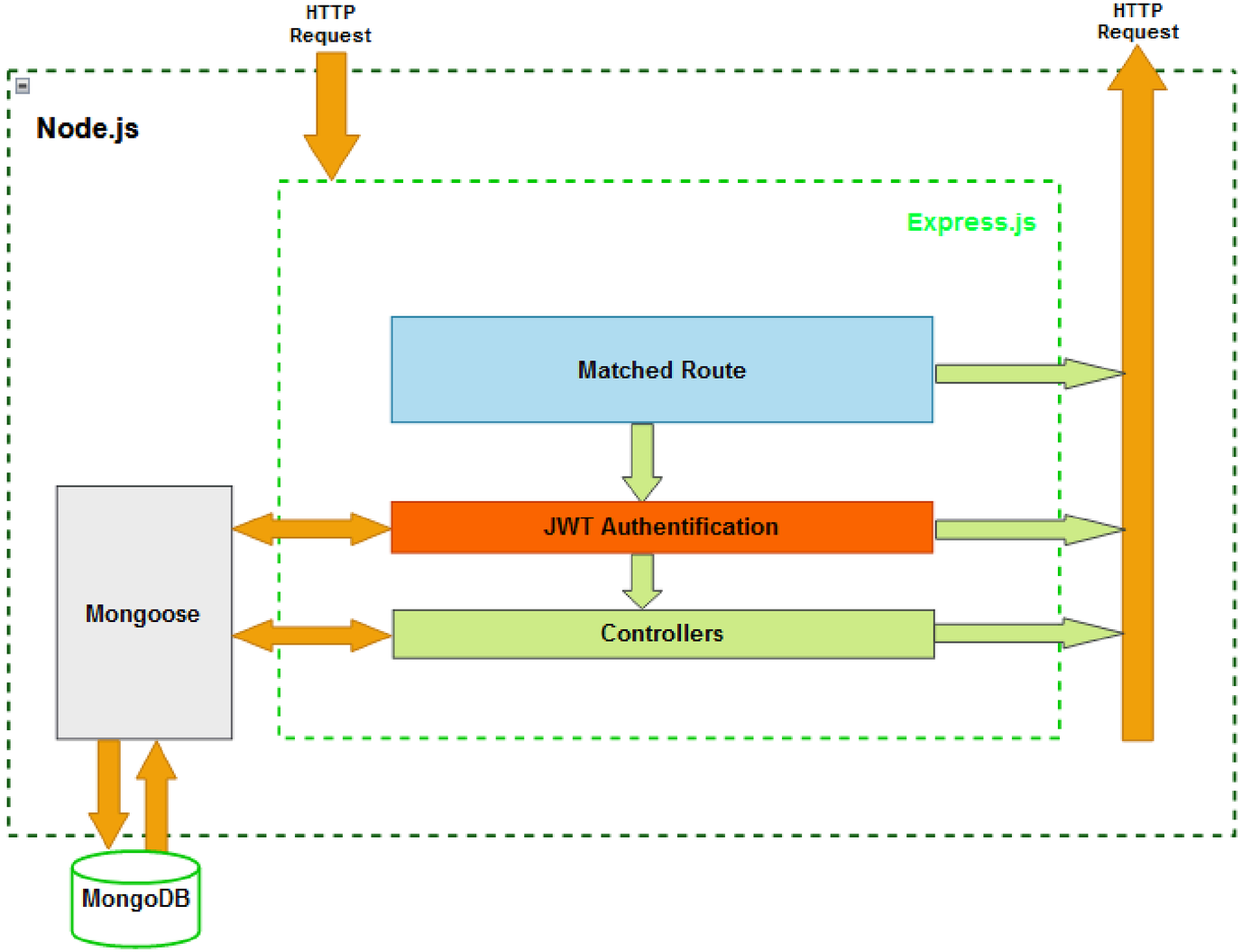
Architecture du système



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

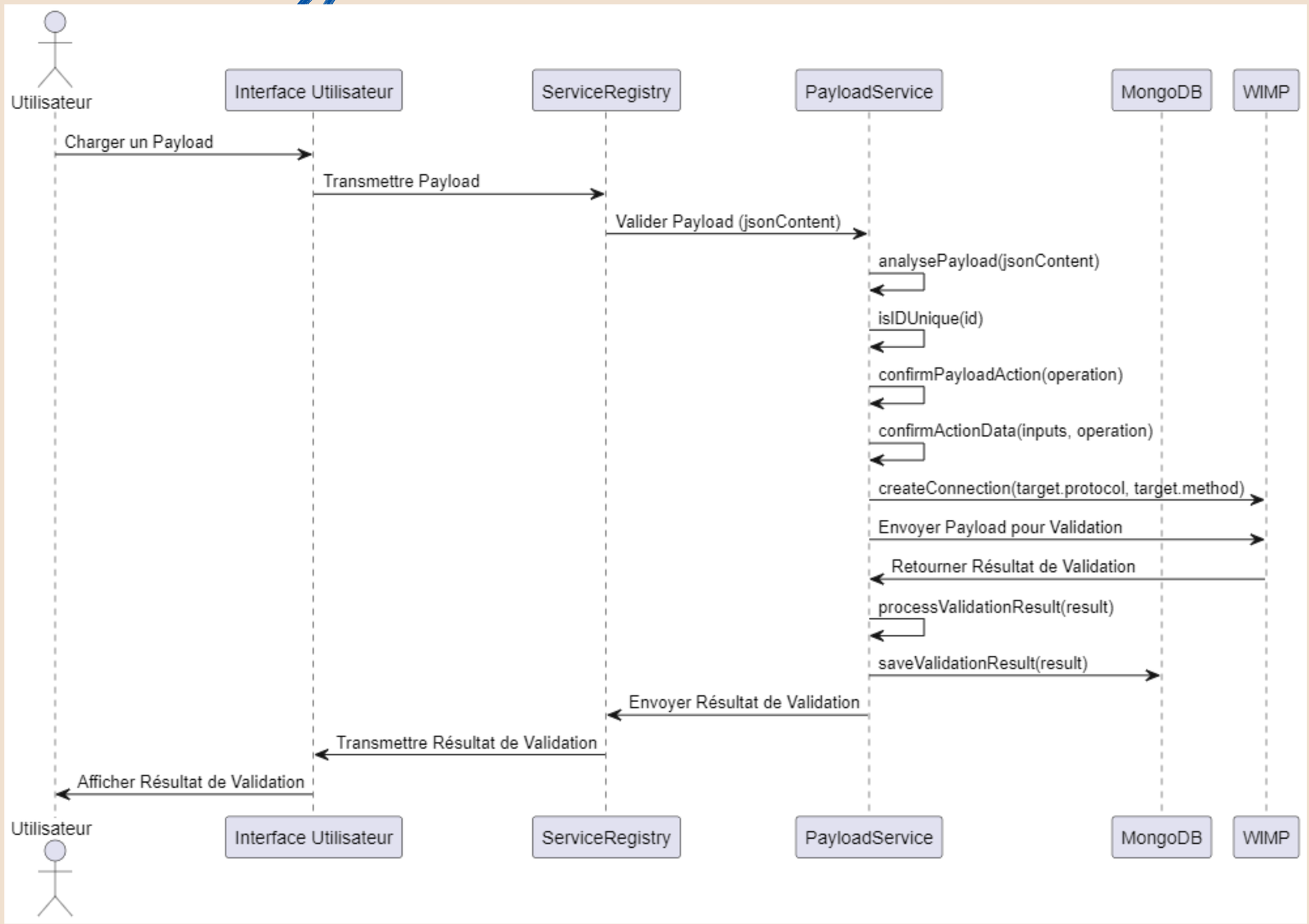
- Express.js fait correspondre les requêtes HTTP avec les routes définies

Mongoose

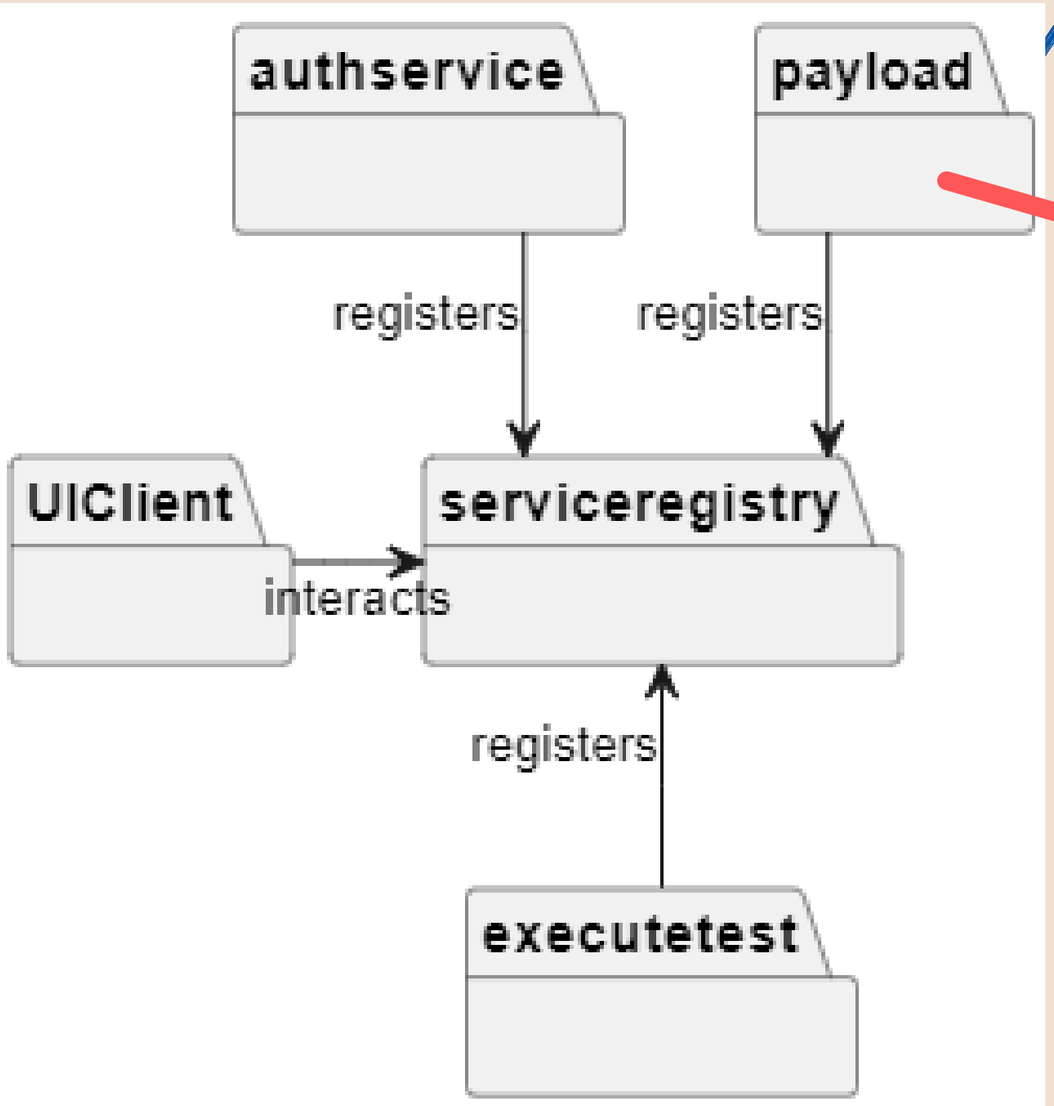
- ODM (Object Data Modeling) pour MongoDB et Node.js
- Interagie avec MongoDB pour les opérations de base de données

Modélisation

Diagramme de séquence envoie et exécution et sauvegarde du payload



Modélisation (suite)



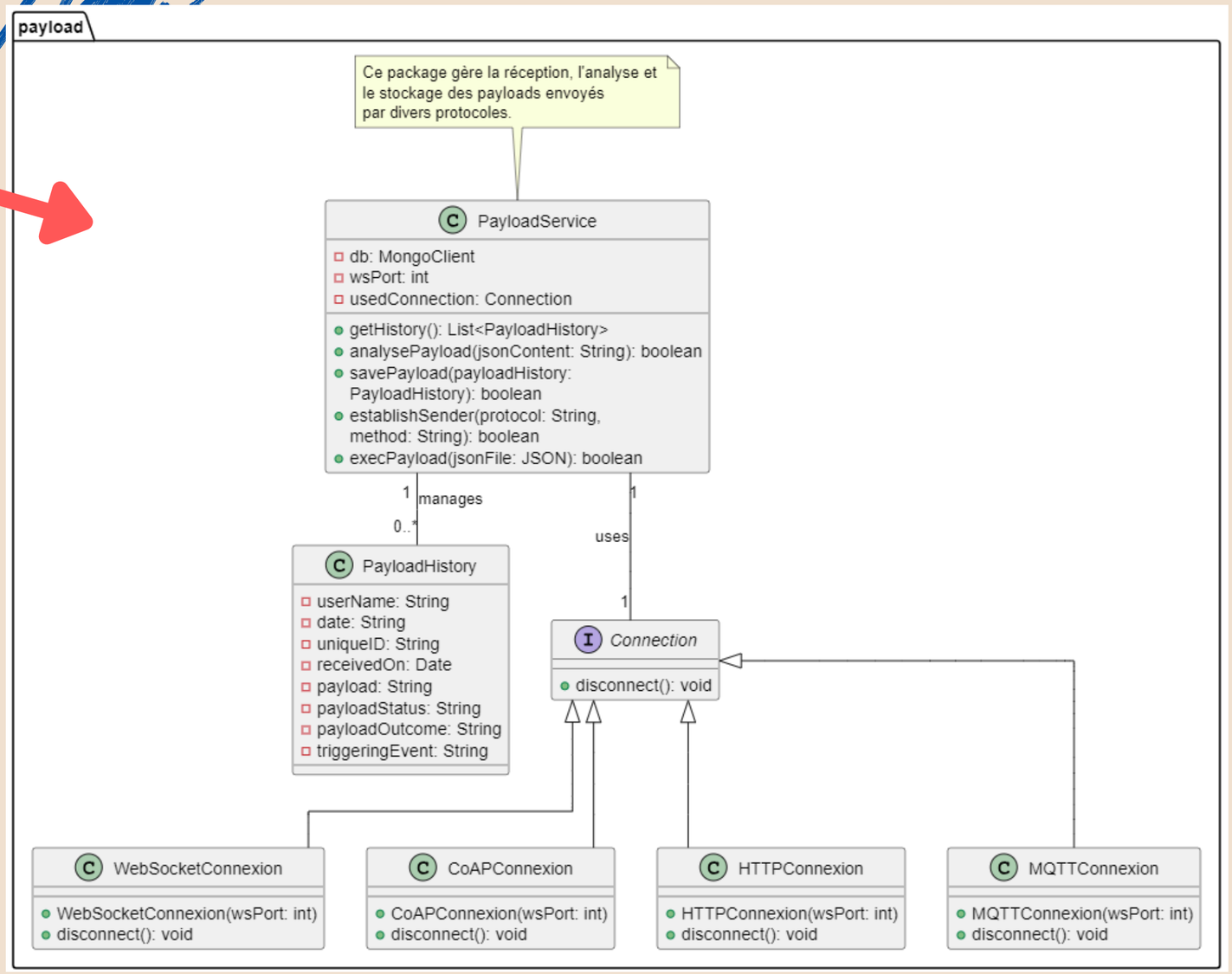
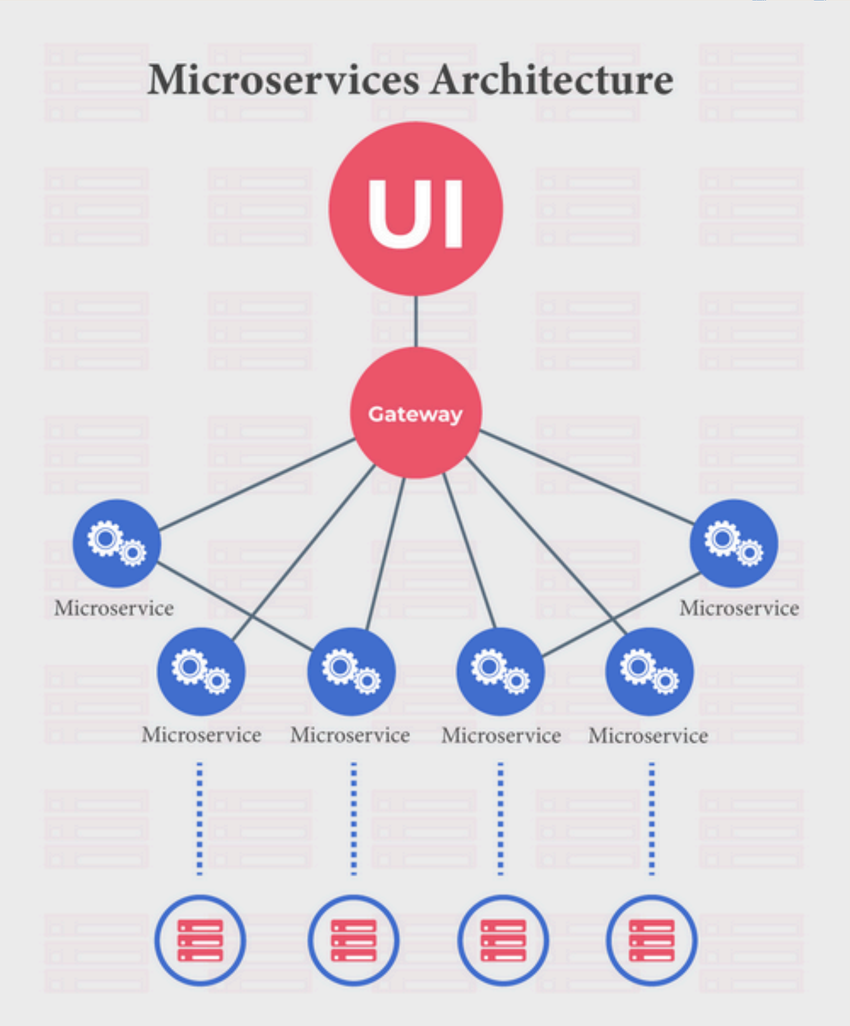


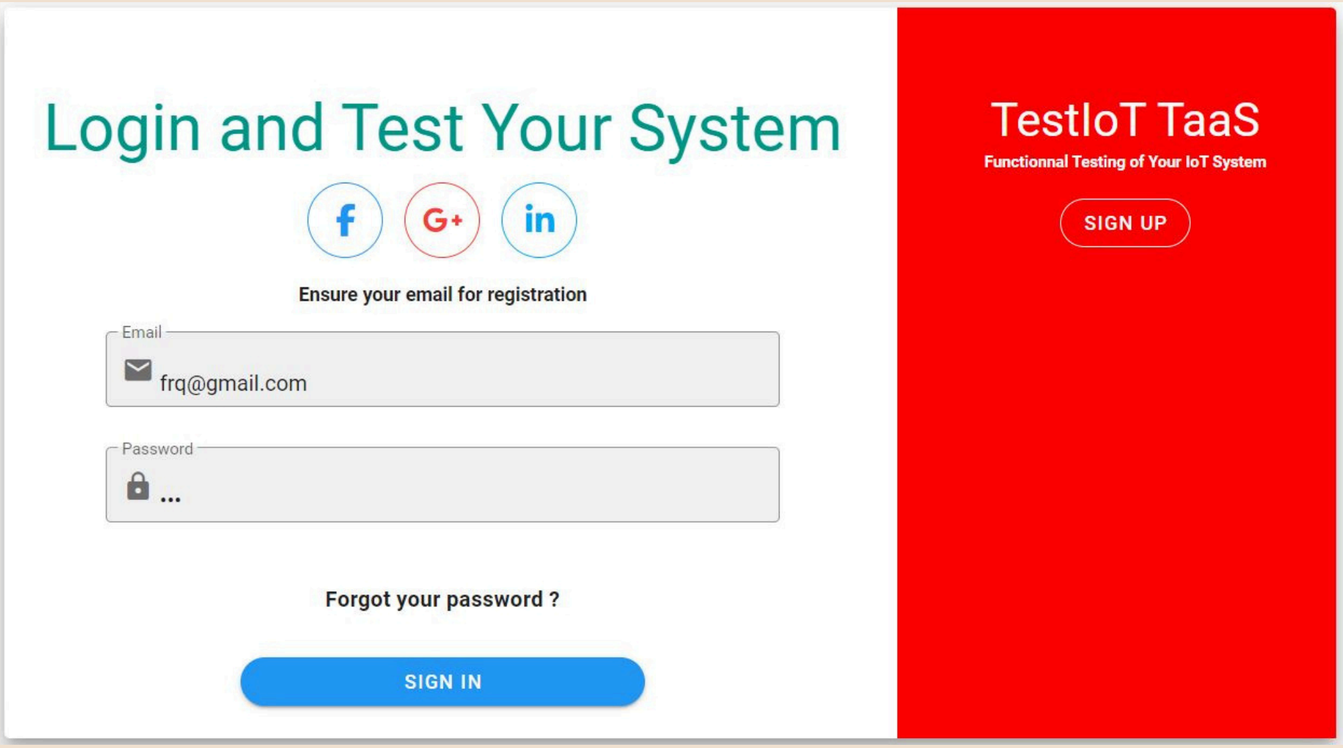
Diagramme de package système et diagramme de classe du microservice Payload

Description des microservices



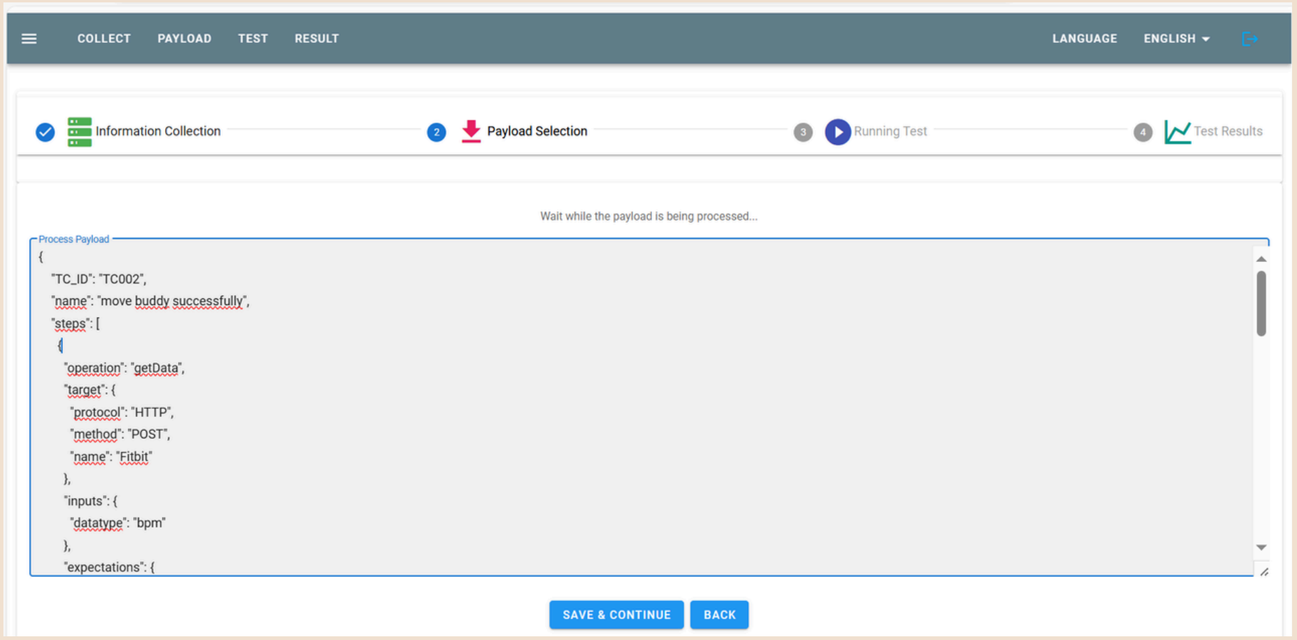
- Service d’authentification
- Service d’analyse du payload
- Service d’exécution des tests

Service d'authentification



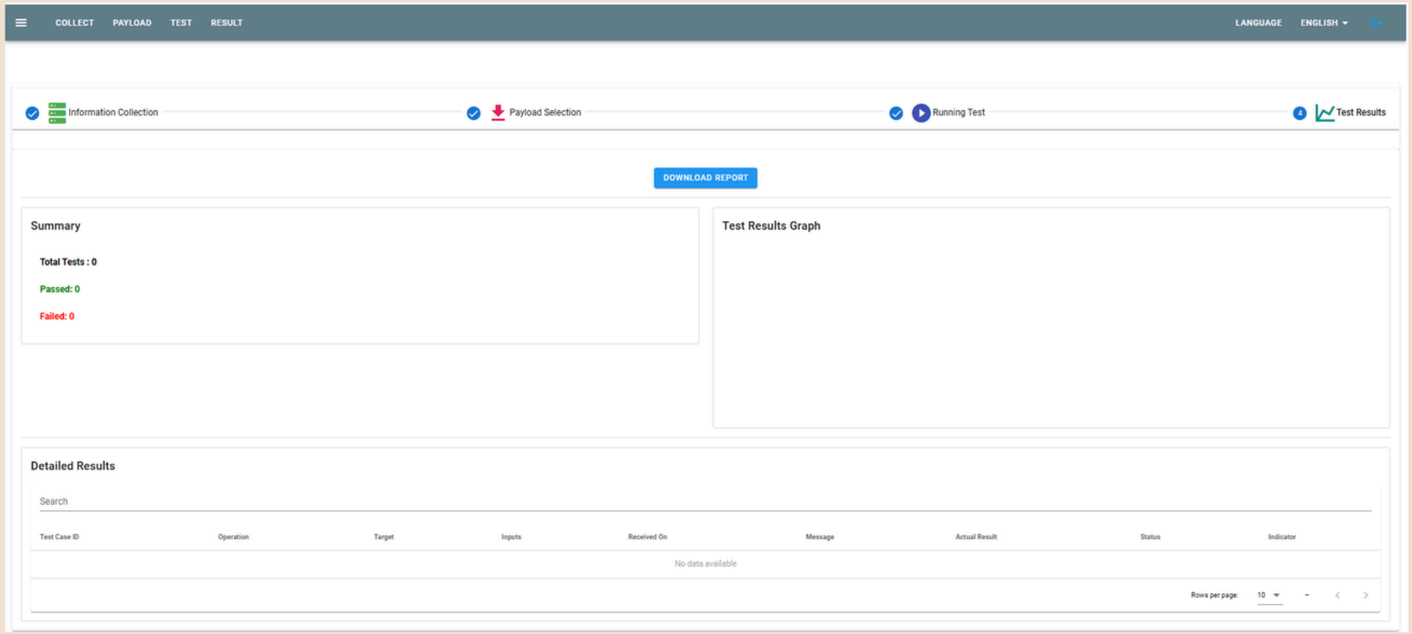
- Gérer connexion et utilisateurs
- Ajouter de nouveaux utilisateurs
- Encrypter mots de passes
- Générer les jetons

Service de gestion du payload



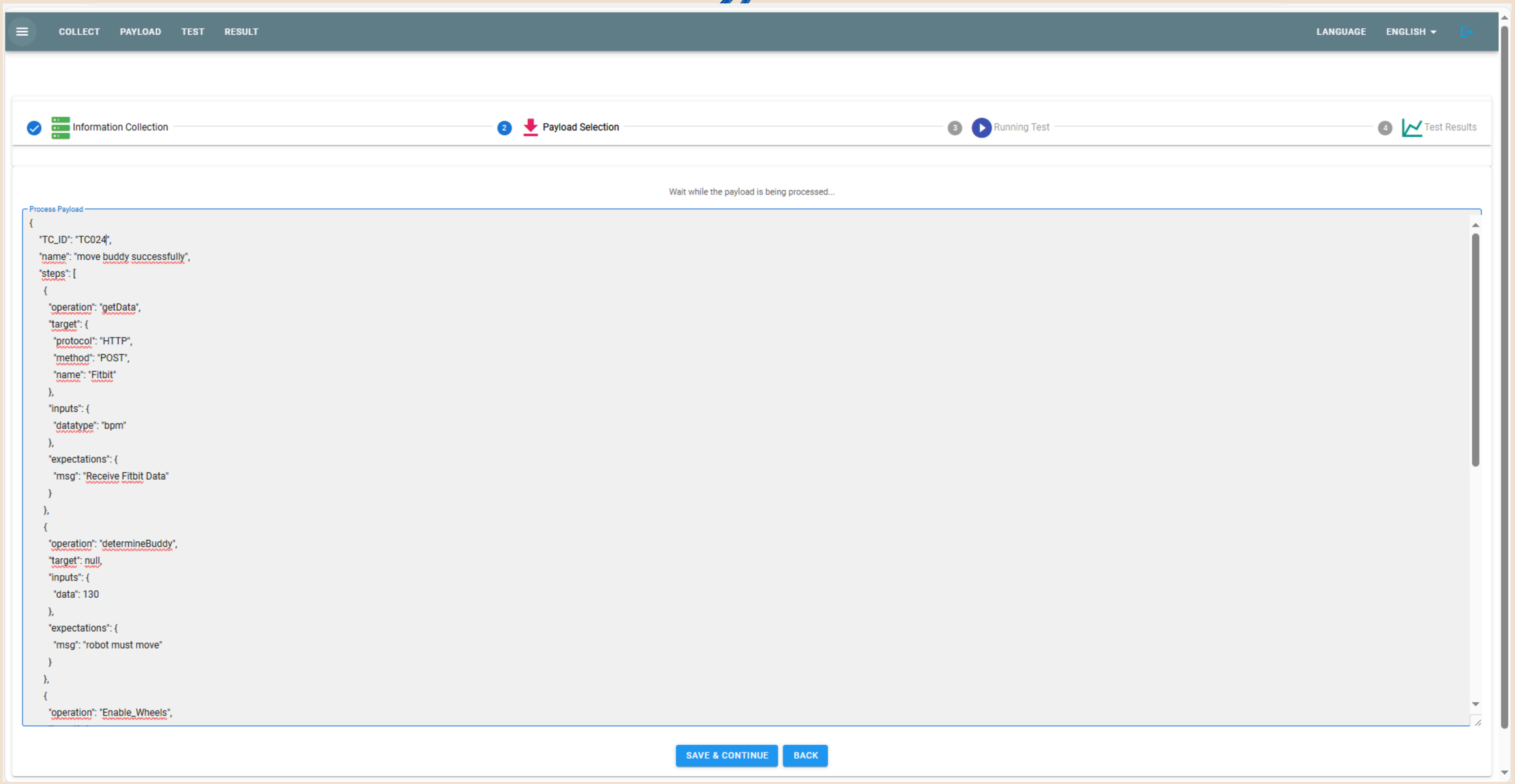
- Analyser le payload
- Envoyer le payload application IoT
- Sauvegarder les informations dans la base de données

Service des tests

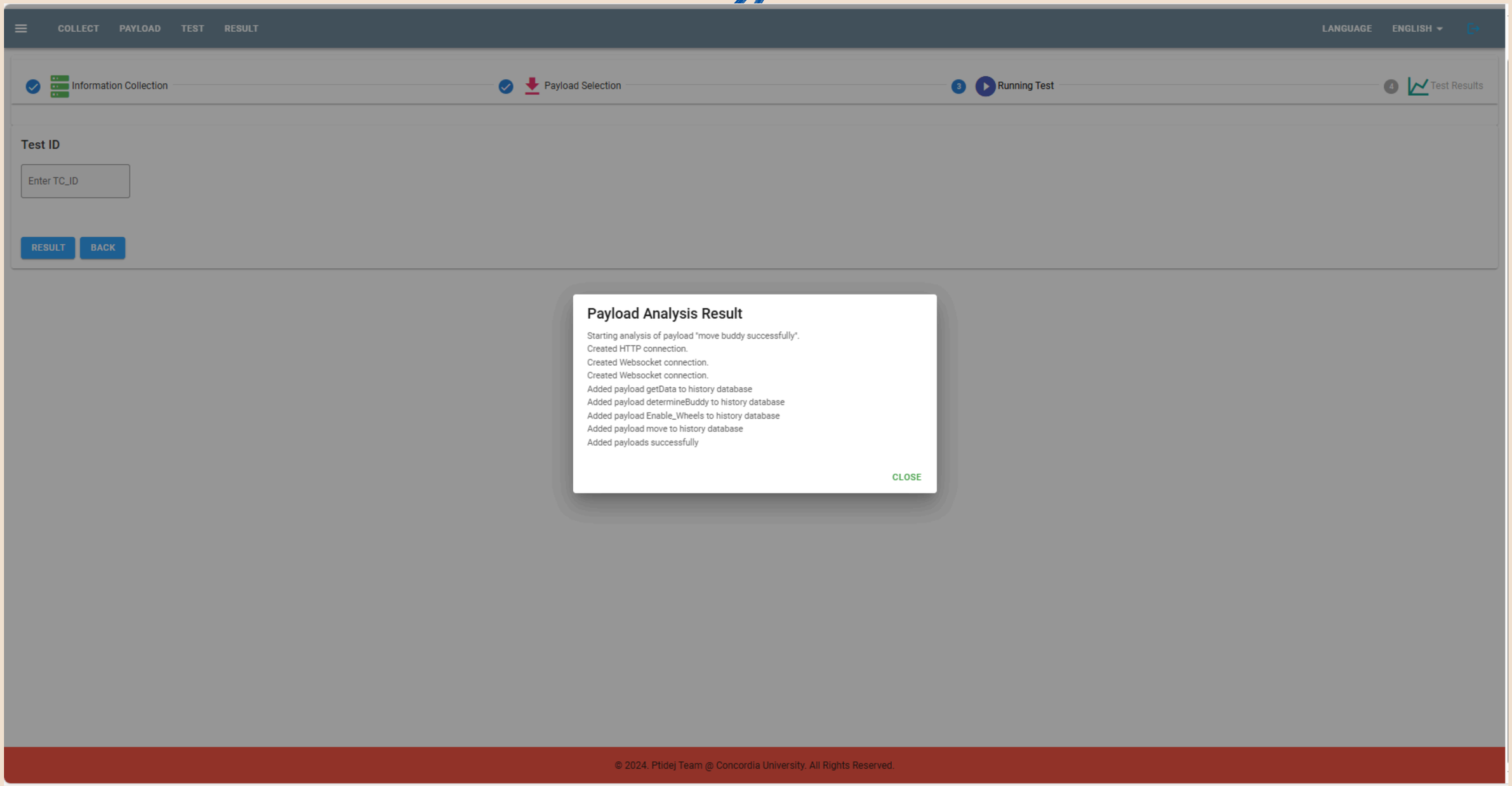


- Gérer tests sauvegardées
- Obtenir résultats de l’application IoT
- Retourner les résultats à utilisateur

Résultat des tests



Résultat des tests



Résultat des tests

COLLECT PAYLOAD TEST RESULT LANGUAGE ENGLISH

Information Collection — Payload Selection — Running Test — 4 Test Results

DOWNLOAD REPORT

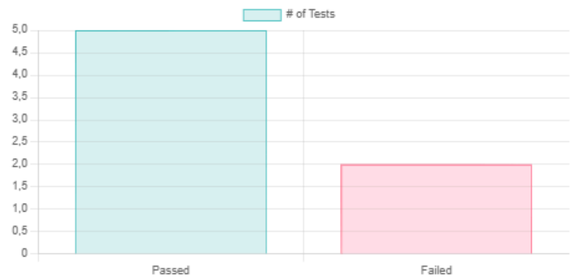
Summary

Total Tests : 7

Passed: 5

Failed: 2

Test Results Graph



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	
TC002	Enable_Wheels	{ "protocol": "Websocket", "method": "send", "name": "Buddy" }	{ "left": 1, "right": 1 }	2024-08-07	WHEELS_ENABLE	WHEELS_ENABLE_FINISHED	Fail	
TC002	Enable_Wheels	{ "protocol": "Websocket", "method": "send", "name": "Buddy" }	{ "left": 1, "right": 1 }	2024-08-07	WHEELS_ENABLE_FINISHED	WHEELS_ENABLE_FINISHED	Success	
TC002	Enable_Wheels	{ "protocol": "Websocket", "method": "send", "name": "Buddy" }	{ "left": 1, "right": 1 }	2024-08-07	WHEELS_ENABLE_FINISHED	WHEELS_ENABLE_FINISHED	Success	
TC002	Enable_Wheels	{ "protocol": "Websocket", "method": "send", "name": "Buddy" }	{ "left": 1, "right": 1 }	2024-08-07	WHEELS_ENABLE	WHEELS_ENABLE_FINISHED	Fail	
TC002	Move	{ "protocol": "Websocket", "method": "send", "name": "Buddy" }	{ "distance": 2, "speed": 1 }	2024-08-11	WHEEL_MOVE_FINISHED	WHEEL_MOVE_FINISHED	Success	
TC002	Move	{ "protocol": "Websocket", "method": "send", "name": "Buddy" }	{ "distance": 3, "speed": 1 }	2024-08-11	WHEEL_MOVE_FINISHED	WHEEL_MOVE_FINISHED	Success	

Rows per page: 10 1-7 of 7

Solutions

1

Conception et développement d’un outil de test sur mésure

2

Architecture Microservice

3

Utulisation des protocoles Websocket, MTTQ, CoAP,HTTP

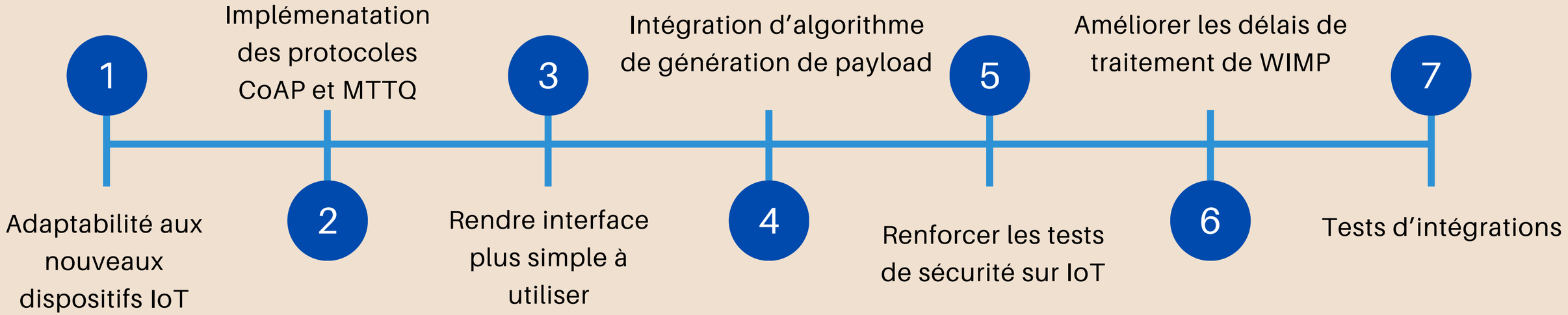
4

Gestion des tests End-to-End

5

Analyses des résultats et sauvegarde

Améliorations et Évolutions



Défis

MICROSERVICE D’AUTHENTIFICATION

Les mots de passes sont maintenant hachés.

FRONTEND

Interface implémenté. Certains détails mineurs manquants.

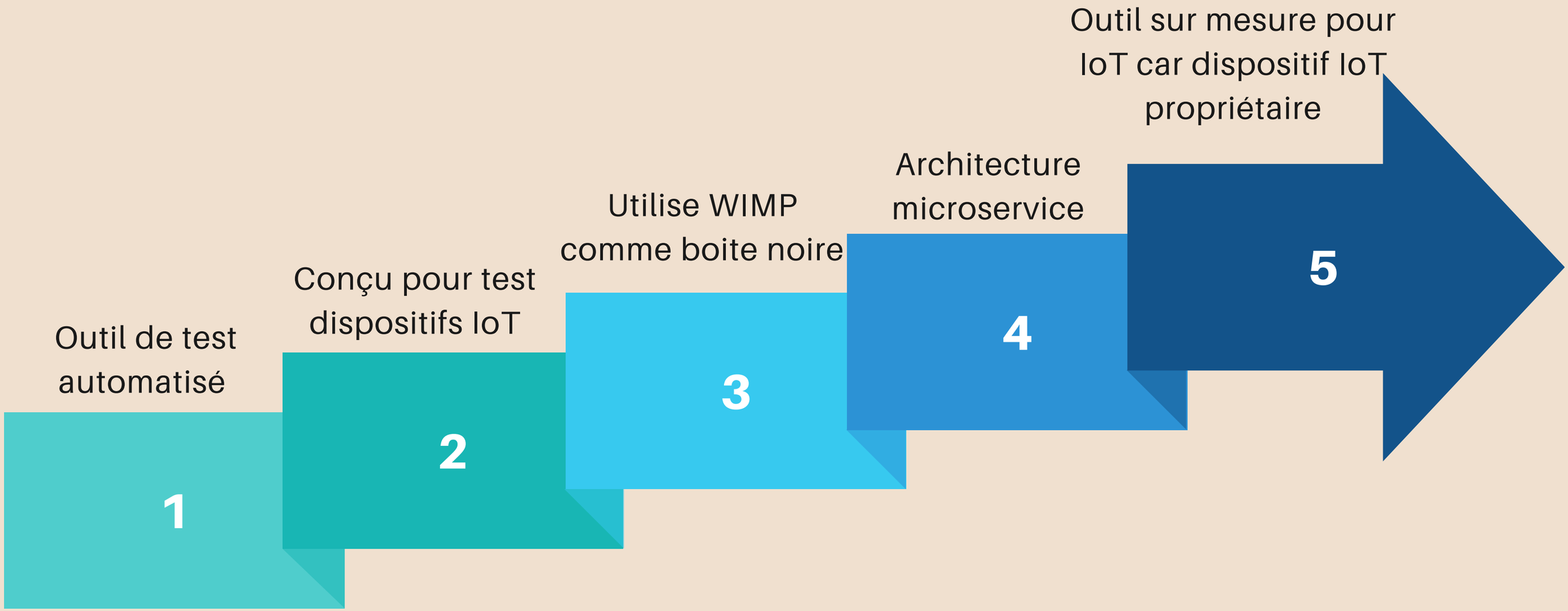
ANALYSE DU PAYLOAD

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

RÉSEAU IOT-3

Problèmes de connexions avec WIMP.

Conclusion


Outil de test automatisé
1
Conçu pour test dispositifs IoT
2
Utilise WIMP comme boite noire
3
Architecture microservice
4
Outil sur mesure pour IoT car dispositif IoT propriétaire
5

Questions

?