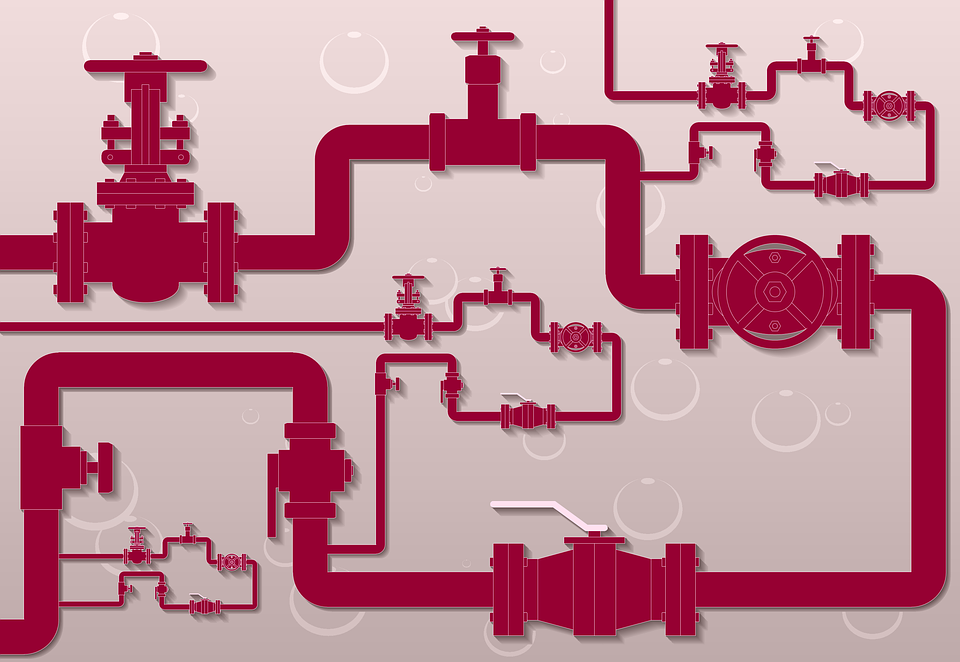
VALLOT Christophe

MANE Jeff Donicson

DOUALI Cédric



***Projet : Compteur d’eau connecté***

***Sommaire***

Table des matières

[Le cahier des charges 3](#_Toc34580679)

[Diagramme UML 4](#_Toc34580680)

[Points et contraintes fonctionnelles et techniques 5](#_Toc34580681)

[L’identification du travail à réaliser 6](#_Toc34580682)

[Maquette de l’IHM 7](#_Toc34580683)

# Le cahier des charges

|  |
| --- |
| Une maison secondaire de bord de mer est équipée d’une installation domotique. Celle-ci nécessite la mesure de la consommation d’eau et la détection d’éventuelles fuites d’eau. |

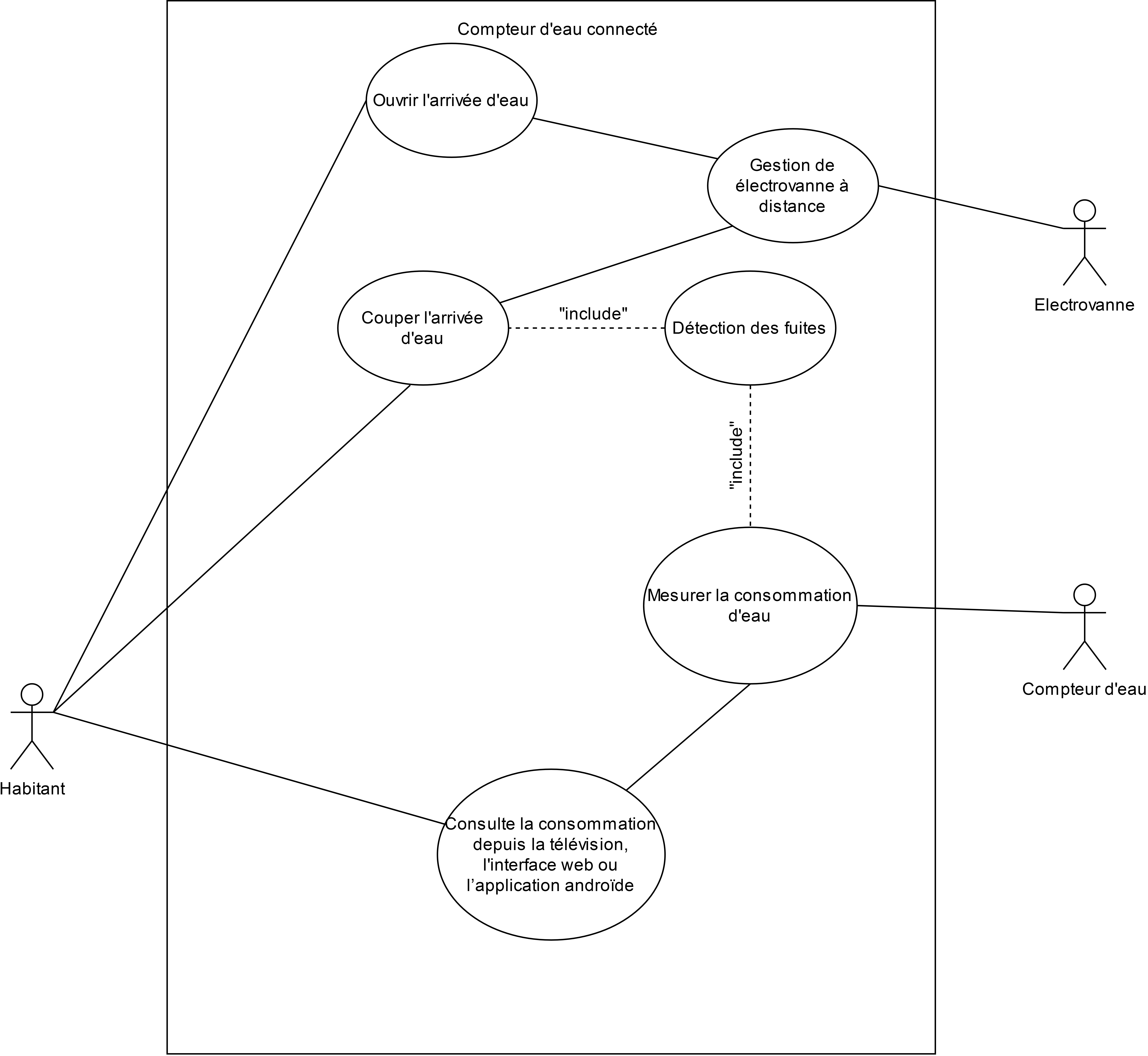
|  |
| --- |
| Dans le cas où une fuite est détectée, une électrovanne coupe l’alimentation en eau. La fonction est assurée par un compteur d’eau à impulsion et une électrovanne connectée |

|  |
| --- |
| sur une carte ESP8266 nodeMCU V3. Les données sont transmises à un serveur web, broker, mqtt en wifi. |

Ces données seront affichées par l’intermédiaire du navigateur du téléviseur du salon et depuis n’importe quel poste connecté sur internet. De plus, elles seront accessibles depuis une application Android

# Diagramme UML

Cas d’utilisation :



# Points et contraintes fonctionnelles et techniques

Points techniques :

Configuration de la carte [ESP8266](https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=11&cad=rja&uact=8&ved=2ahUKEwik7sqcp4joAhUF3IUKHTh8CtIQFjAKegQIBRAB&url=https%3A%2F%2Ffr.wikipedia.org%2Fwiki%2FESP8266&usg=AOvVaw0ci9AGVGpRbaKi_gsjv66d) :

* Librairie pour le wifi « ESP8266WiFi.h »
* Librairie pour le [MQTT](https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=9&cad=rja&uact=8&ved=2ahUKEwju-ILi6IroAhUBxIUKHd4lBZQQFjAIegQIAhAB&url=http%3A%2F%2Fmqtt.org%2F&usg=AOvVaw1kJdOPEcd5GGWuRdKvE4DU) « PubSubClient.h »
* Programmation sur l’IDE Arduino

Configuration du Raspberry :

* Configuration du [SSH](https://fr.wikipedia.org/wiki/Secure_Shell)
* Apache, [phpMyAdmin](https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&cad=rja&uact=8&ved=2ahUKEwiqrJqJ7YroAhW85OAKHbfEBb4QFjAAegQIEhAC&url=https%3A%2F%2Fwww.phpmyadmin.net%2F&usg=AOvVaw00sojKkLR7MeGR2Z0A1YhA) (Base de donnée [MySQL](https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&cad=rja&uact=8&ved=2ahUKEwjN1u767IroAhVLAWMBHUkeBWcQFjAAegQIBBAB&url=https%3A%2F%2Fwww.mysql.com%2Ffr%2F&usg=AOvVaw1xHe9NLinx_ajFTyW34S9_))
* librairie [PHP](https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=2&cad=rja&uact=8&ved=2ahUKEwi2wO6FqojoAhUh0uAKHc50AUIQFjABegQICRAB&url=https%3A%2F%2Fwww.php.net%2F&usg=AOvVaw1JqmoYucRr7TaiUciQETQq) pour le protocole [MQTT](https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=9&cad=rja&uact=8&ved=2ahUKEwju-ILi6IroAhUBxIUKHd4lBZQQFjAIegQIAhAB&url=http%3A%2F%2Fmqtt.org%2F&usg=AOvVaw1kJdOPEcd5GGWuRdKvE4DU)
* Broker [Mosquitto](https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&cad=rja&uact=8&ved=2ahUKEwiMpoqw6YroAhVRXBoKHSgpCHAQFjAAegQIAxAB&url=https%3A%2F%2Fmosquitto.org%2F&usg=AOvVaw3_kUQEdq1eLHouykhO8uyk)

Création de l’application Android :

* Installation de Android studio
* Création d’une [API REST](https://fr.wikipedia.org/wiki/Representational_state_transfer)

Contraintes fonctionnelles :

* Le développement web doit être en [HTML](https://fr.wikipedia.org/wiki/Hypertext_Markup_Language), [CSS](https://fr.wikipedia.org/wiki/Feuilles_de_style_en_cascade), [JavaScript](https://fr.wikipedia.org/wiki/JavaScript), [Ajax](https://fr.wikipedia.org/wiki/Ajax_(informatique)), [PHP](https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=2&cad=rja&uact=8&ved=2ahUKEwi2wO6FqojoAhUh0uAKHc50AUIQFjABegQICRAB&url=https%3A%2F%2Fwww.php.net%2F&usg=AOvVaw1JqmoYucRr7TaiUciQETQq)
* Application Android doit être crée sur Android studio
* La carte [ESP8266](https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=11&cad=rja&uact=8&ved=2ahUKEwik7sqcp4joAhUF3IUKHTh8CtIQFjAKegQIBRAB&url=https%3A%2F%2Ffr.wikipedia.org%2Fwiki%2FESP8266&usg=AOvVaw0ci9AGVGpRbaKi_gsjv66d) nodeMCU V3 est connectée en wifi
* Utilisation du protocole [MQTT](https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=9&cad=rja&uact=8&ved=2ahUKEwju-ILi6IroAhUBxIUKHd4lBZQQFjAIegQIAhAB&url=http%3A%2F%2Fmqtt.org%2F&usg=AOvVaw1kJdOPEcd5GGWuRdKvE4DU)
* On doit fournir une maquette fonctionnelle
* Gestion de l’adresse IP du Raspberry

# L’identification du travail à réaliser

Pour le projet « Compteur d’eau connecté », chaque étudiant c’est vu attribuer une tache spécifique :

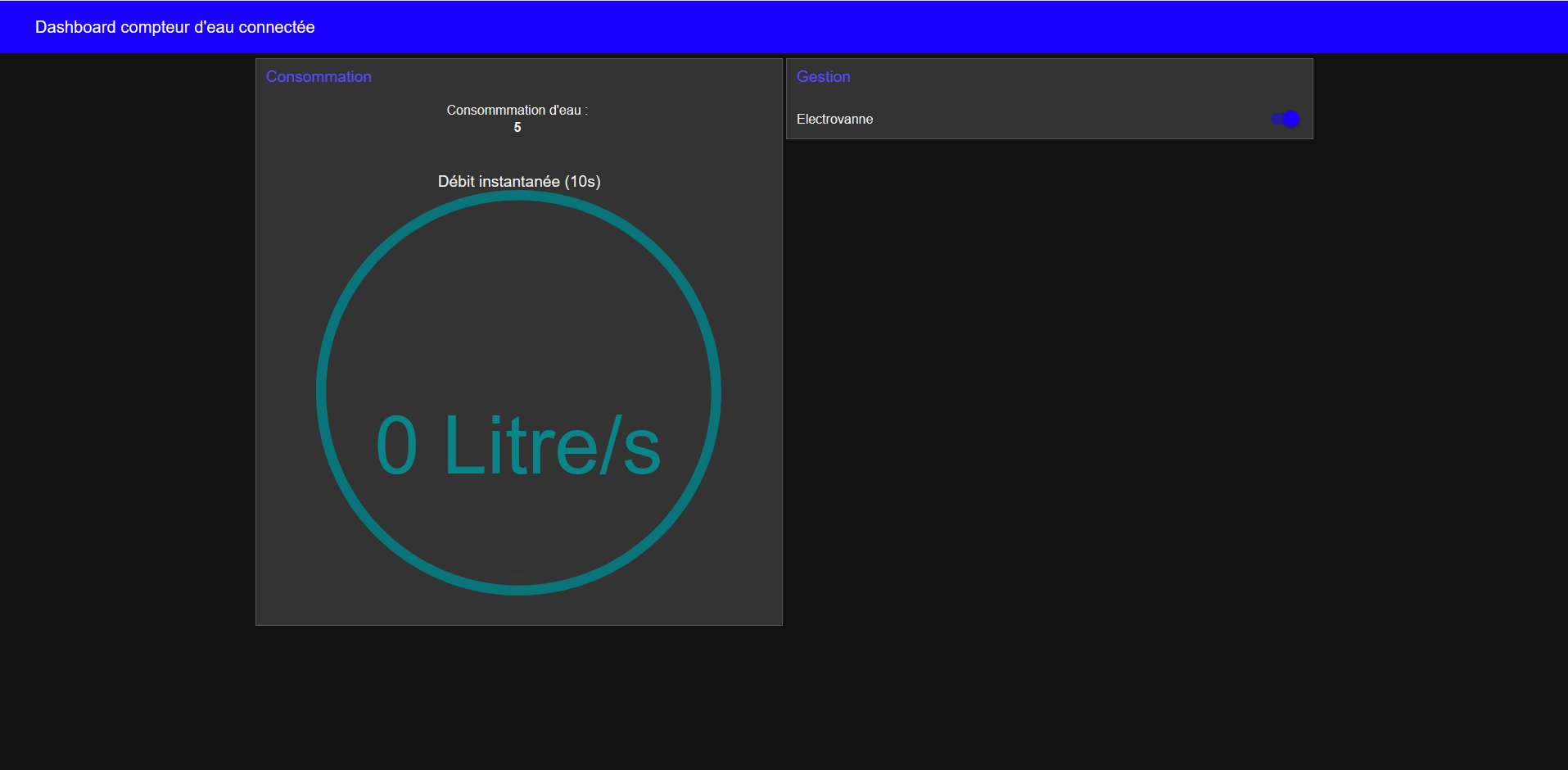
L’étudiant 1 doit paramétrer la carte Heltec [ESP8266](https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=11&cad=rja&uact=8&ved=2ahUKEwik7sqcp4joAhUF3IUKHTh8CtIQFjAKegQIBRAB&url=https%3A%2F%2Ffr.wikipedia.org%2Fwiki%2FESP8266&usg=AOvVaw0ci9AGVGpRbaKi_gsjv66d) afin de la connecter au wifi. La carte Heltec doit être raccordée au compteur d’eau et à l’électrovanne. Elle doit pouvoir arrêter la circulation de l’eau en cas de fuite. La communication entre la carte et le Raspberry sera effectué par le protocole [MQTT](https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=9&cad=rja&uact=8&ved=2ahUKEwju-ILi6IroAhUBxIUKHd4lBZQQFjAIegQIAhAB&url=http%3A%2F%2Fmqtt.org%2F&usg=AOvVaw1kJdOPEcd5GGWuRdKvE4DU). Le Raspberry doit entrer les informations dans la base donnée a l’aide d’un script Shell exécutable au démarrage. L’étudiant 1 a à sa charge de coder une [A](https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=14&cad=rja&uact=8&ved=2ahUKEwjL9Iyzp4joAhVhx4UKHWP5BRMQFjANegQIBxAB&url=https%3A%2F%2Ffr.wikipedia.org%2Fwiki%2FRepresentational_state_transfer&usg=AOvVaw30VLb0PRFUxcN0n_S2DcpT)PI REST en [PHP](https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=2&cad=rja&uact=8&ved=2ahUKEwi2wO6FqojoAhUh0uAKHc50AUIQFjABegQICRAB&url=https%3A%2F%2Fwww.php.net%2F&usg=AOvVaw1JqmoYucRr7TaiUciQETQq). Elle doit se connecter à la base de données et afficher les informations demandées par le client. Elle doit pouvoir renvoyer uniquement les données demander et contrôler l’électrovanne en [MQTT](https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=9&cad=rja&uact=8&ved=2ahUKEwju-ILi6IroAhUBxIUKHd4lBZQQFjAIegQIAhAB&url=http%3A%2F%2Fmqtt.org%2F&usg=AOvVaw1kJdOPEcd5GGWuRdKvE4DU). La transmission des données depuis l’api sera sous format [JSON](https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&cad=rja&uact=8&ved=2ahUKEwjEvZaUqYjoAhX1D2MBHYk5DuMQFjAAegQIAxAB&url=https%3A%2F%2Fwww.json.org%2Fjson-fr.html&usg=AOvVaw0JCNgNN9CAiDCvkR6e7E-X). Le Raspberry utilisera un Dashboard [Node-RED](https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&cad=rja&uact=8&ved=2ahUKEwj82P7_nYjoAhXpA2MBHTYuBhwQFjAAegQIARAB&url=https%3A%2F%2Fnodered.org%2F&usg=AOvVaw20oC2WulwYcBnUuJlWjzaH) contrôlant l’électrovanne et affichant les informations sur le système.

L’étudiant 2 doit créer une application Android. Cette dernière doit se connecter à l’API REST et afficher les données de la base de donnée.

L’étudiant 3 doit mettre en place le serveur apache et la base de données. Il doit également créer une interface en [PHP](https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=2&cad=rja&uact=8&ved=2ahUKEwi2wO6FqojoAhUh0uAKHc50AUIQFjABegQICRAB&url=https%3A%2F%2Fwww.php.net%2F&usg=AOvVaw1JqmoYucRr7TaiUciQETQq) capable d’afficher et de mettre en forme le contenu de la base de donnée.

# Maquette de l’IHM

Dashboard Node-red :

******

***Scénario***

Dans le cas où l’utilisateur veut arrêter la circulation de l’eau, il doit se rendre sur l’interface du Raspberry.  
Il doit cliquer sur le slider électrovanne. Celui-ci deviendra gris et affiche l’etat OFF.

Quand l’utilisateur

***Spécification de la base de donnée et des protocoles de communication***

Http utilisé sur l’API REST et le site web, le HTTP est protocole de un protocole de communication client-serveur. Utilisée essentiellement pour le web. Il se trouve sur la couche application du modèle OSI (de l’anglais Open Systems Interconnection). Par défaut le port un serveur HTTP utilise le port 80. L’api rest du projet utilise des requêtes GET qui permettent récupère la ressource depuis un client ici l’application Android

MQTT (ESP8266, API REST, Node red)

Protocole de messagerie publish-subcribe. Il a été conçu pour les communications avec des clients distants où la bande passante du réseau est limitée. Le protocole nécessite la présence d’un serveur appeler broker. Le broker va transmettre les donnés des clients publicher au différent abonné en fonction des Topics concerner. Dans le projet L’ESP8266 fait des requetes publish sur le broker afin d’envoyer les données. La carte subscribe également sur le topic « gestion » qui permet a l’api rest et a l’interface node red de contrôle l’etat de l’électrovanne

TCP/IP ???