Dossier de conception d'un compteur d'énergies connecté

Description :

Dans les ports de plaisance, les yachts ont besoin d'énergies lors de leur amarrage sur le quai du port.

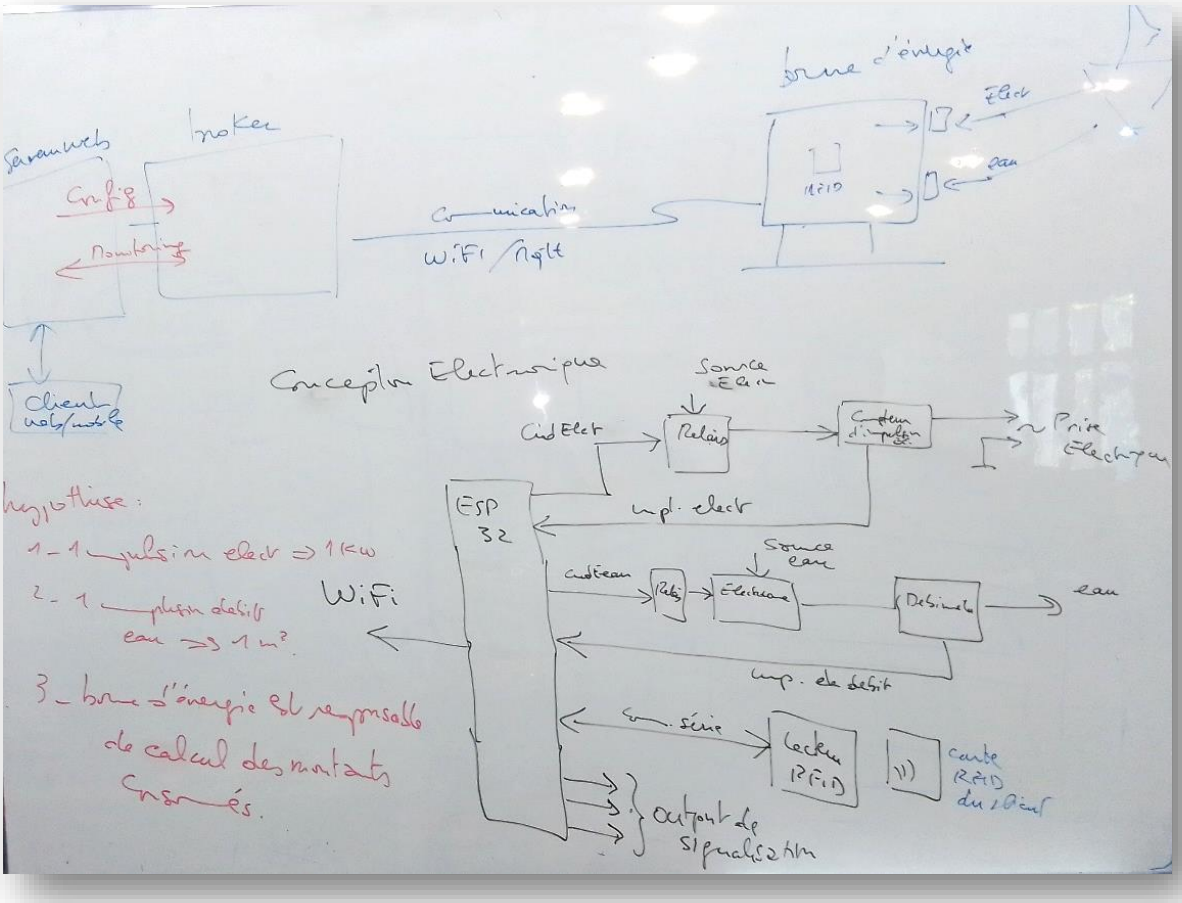
Chaque yacht sera connecté à une source multiple d'énergie mais, au même temps, elle doit être contrôlé.

La source d'énergie est fournie par une borne d'énergies connecté. Cette borne fournie :

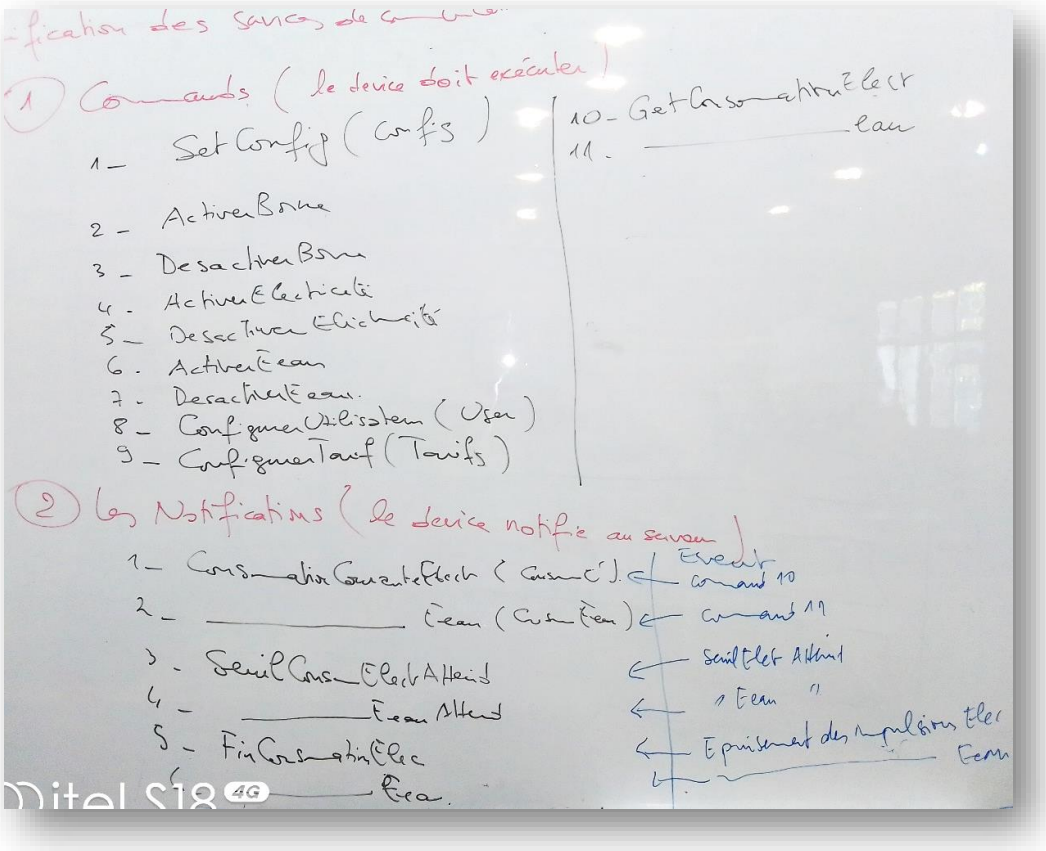
- Une prise d'énergie électrique
- Une vanne d'eau
- Un système de comptage de consommation de chaque type d'énergie
- Un système d'activation / désactivation par carte RFID
- Un système de communication pour sa configuration et son monitoring.

L'objectif de ce projet est de concevoir et développer cette borne.

Architecture générale et schéma bloc de la borne

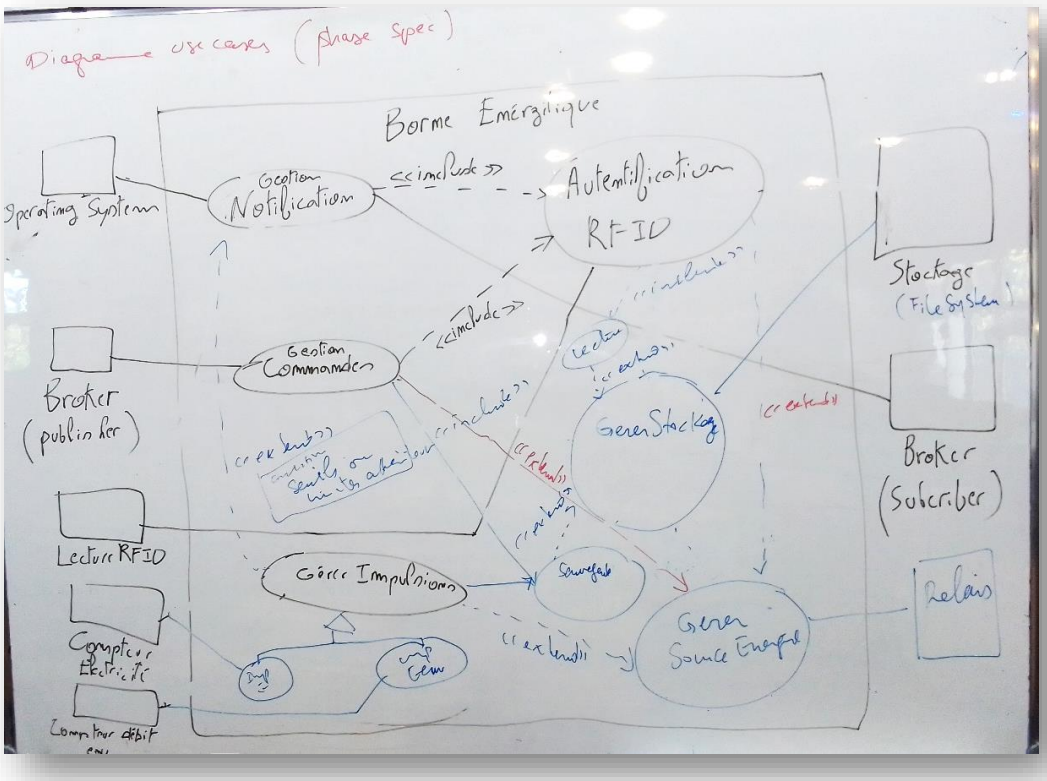


Spécification des services de communication



Dans la suite de ce rapport, on s'intéresse au composant logiciel « Borne d'énergie »

Phase spécification : Diagramme de cas d'utilisation

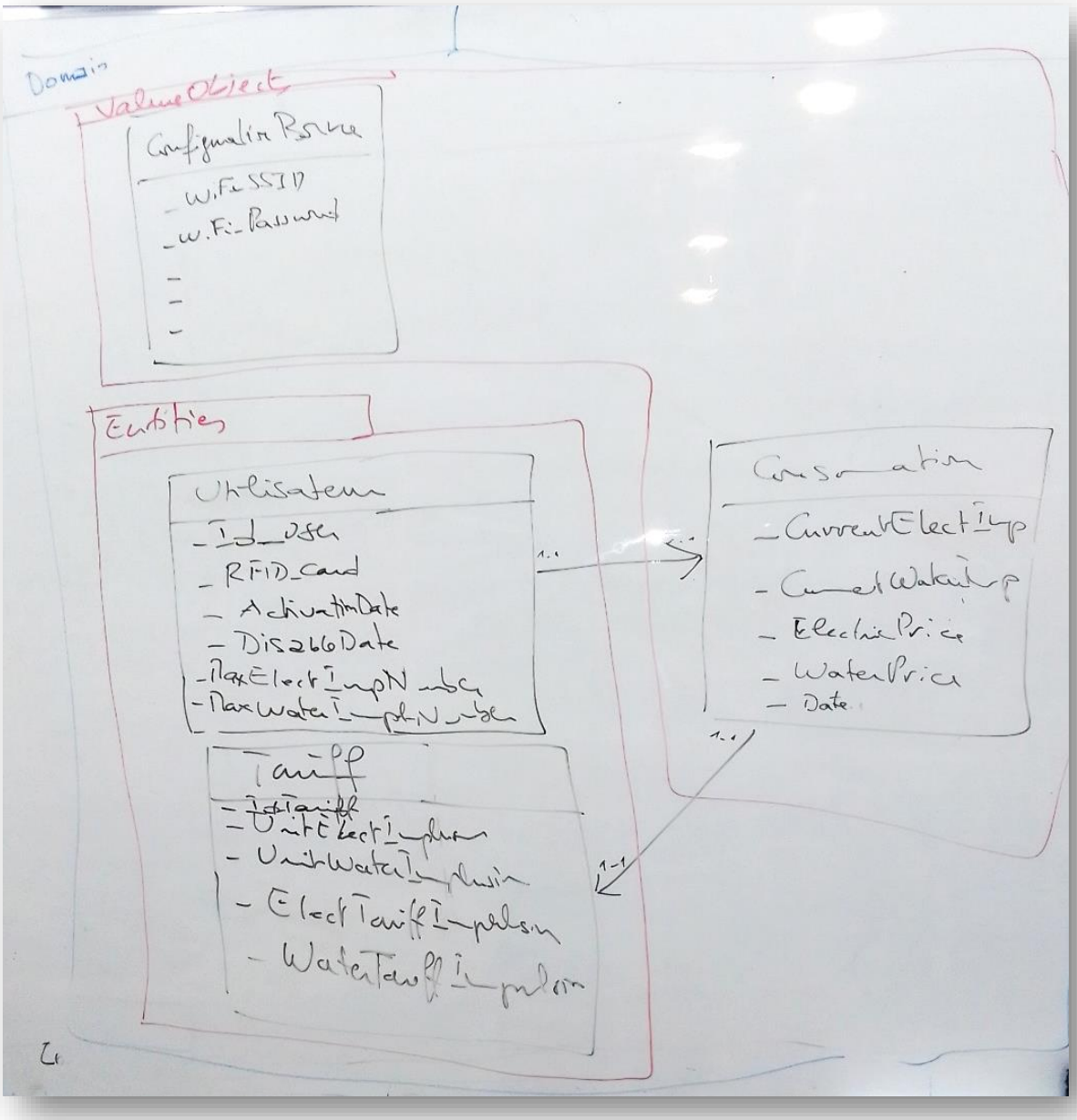


Il faut modifier les points suivants :

« Gestion notification » et « Gestion commandes » n'inclus pas « Authentification »

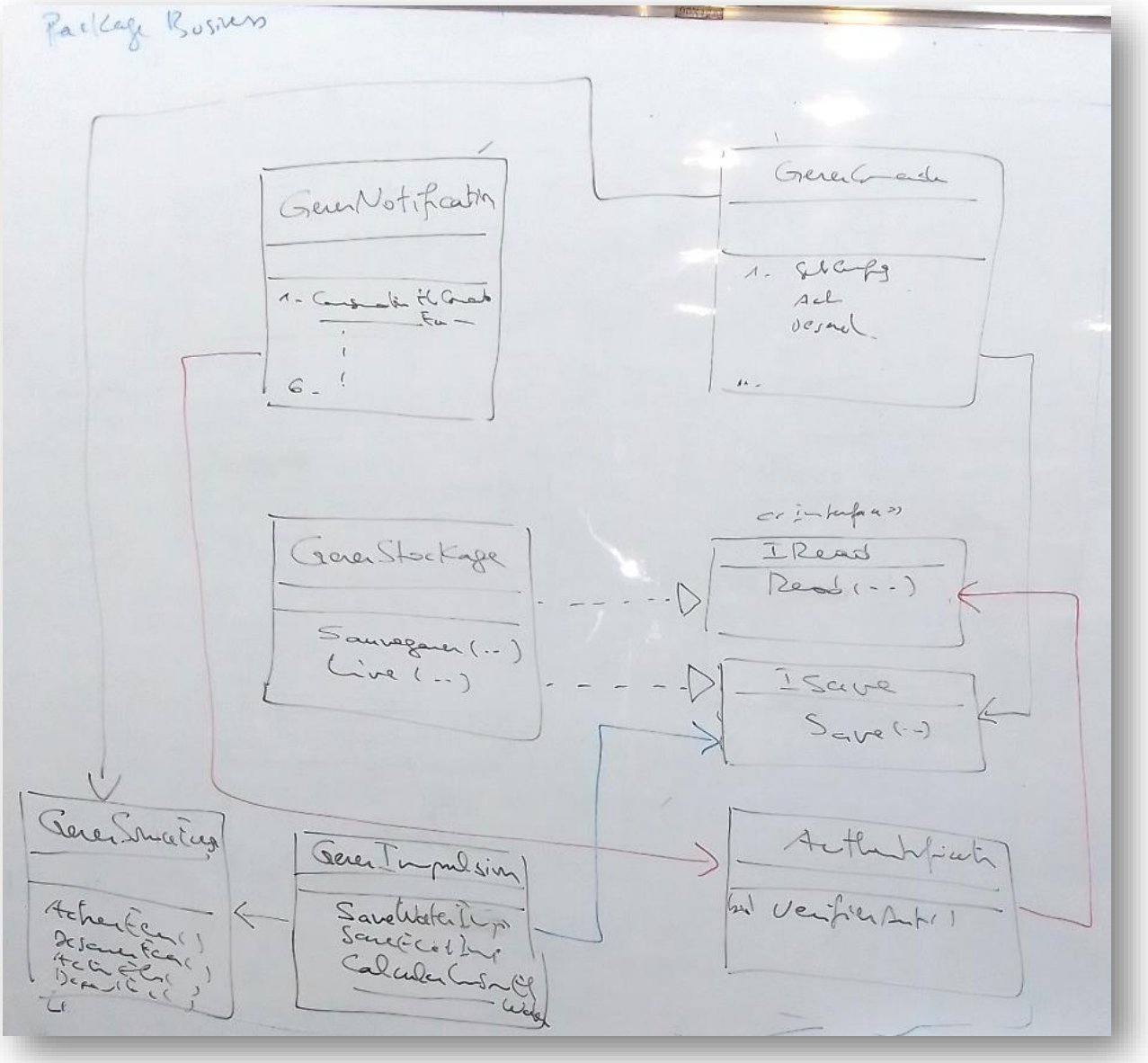
« Gérer Impulsion » inclus « Authentification »

Phase analyse : Diagramme de classes du domaine



Phase conception : Diagramme de classes participatives

(Niveau PIM : Platform Independant Model)



Notes :

Suite aux modifications du diagramme de cas d'utilisation, les modifications suivantes doivent être pris en considération :

1- Supprimer la relation entre « Gérer Notification » et « Authentification »
2- Ajouter la relation de type « has » entre « Gérer Impulsion » et « Authentification »

Phase conception : Diagramme de classes de conception

(Niveau PSM : Platform Specific Model)

Decisions technologiques :

Hardware: ESP32 (architecture ARM)

OS : FreeRtos

Framework: Arduino

Langage: C++

Architecture de notre composant logiciel : clean architecture

Patron d'architecture : ? (Aucun)

Patron de conception : observable

…

Diagramme de paquetage

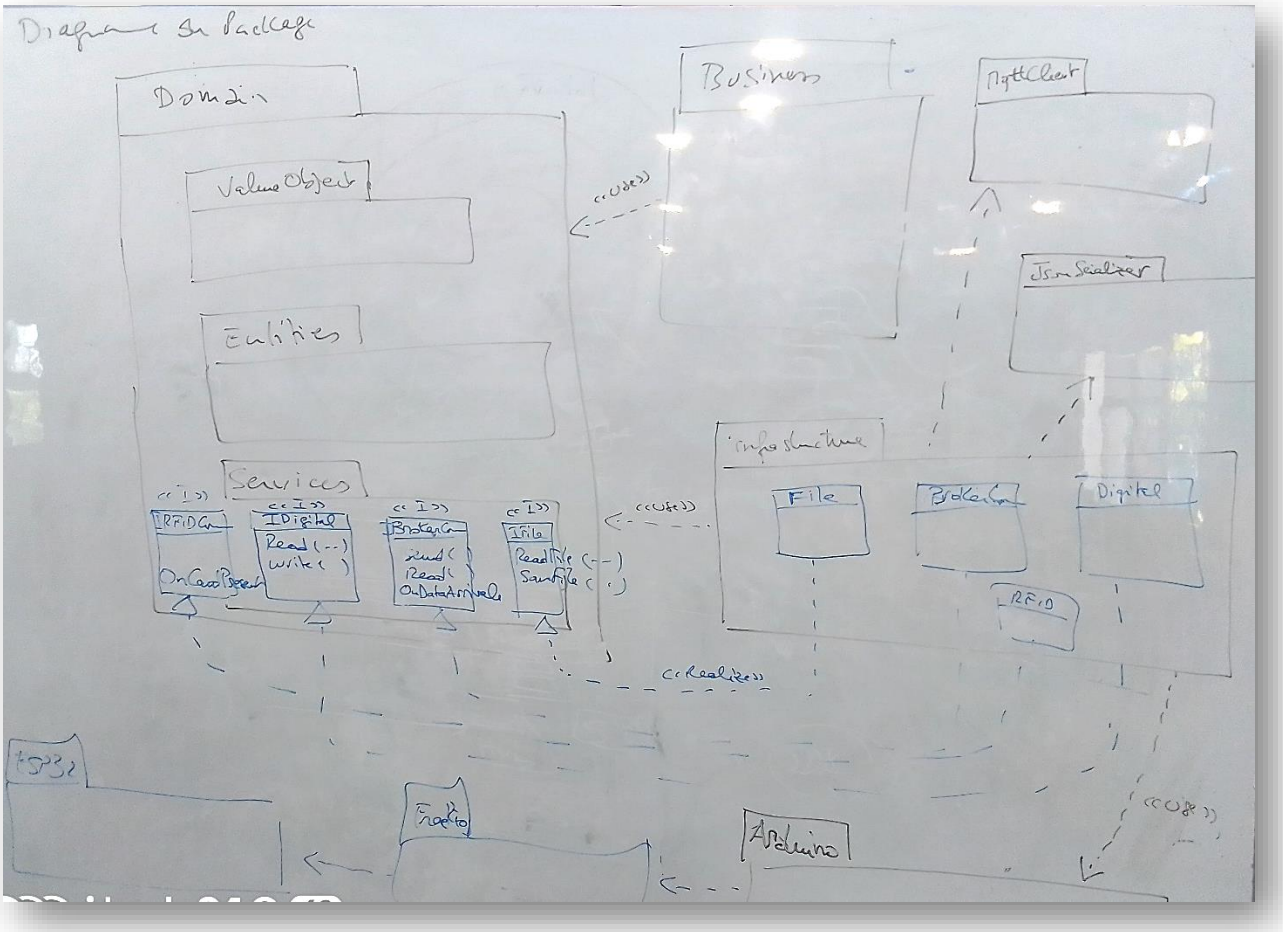


Diagramme d'état / transition (FSM)

Deux machines à état fini à définir:

1- Dans la classe GérerImpulsion : FsmImpulsion()
2- Dans la classe Rfid : FsmRfid()
Pour chaque FSM, il faut définir ses états et ses transitions.

Tasks :

Une seule tâche à créer : ImpulsionTask -> execute périodéquement (temps ?) la méthode FsmImplusion() de la classe GérerImplusion.

Timer :

Un timer à utiliser pou la scrutation du lecteur Rfid : exécution de laméthode FsmRfid() de la classe Rfid.