REPUBLIQUE ALGERIENNE DEMOCRATIQUE POPULAIRE

MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITE ABDELHAMID IBN BADIS MOSTAGANEM

FACULTE DES SCIENCES EXACTE ET DE L'INFORMATIQUE

MEMOIRE DE FIN D'ETUDES

En vue de l'obtention d'un diplôme du Master en Informatique
Option : réseaux et systèmes

LA MISE EN ŒUVRE D'UNE SOLUTION OPEN SOURCE POUR LA GESTION DES LABORATOIRES DISTANTS

Réalisé par : BELGHIT Ismail.

Sous la direction de MR. MOUSSA Mohamed.

Année universitaire : 2018-2019

Remerciement

Louange à Dieu qui m'a donné la force, le courage, et l'espoir nécessaire Pour accomplir ce travail et surmonter l'ensemble des difficultés.

Je tiens tout particulièrement à remercier mon encadrant Monsieur MOUSSA Mohamed, pour son aide, son soutien, ses conseils, sa patience et sa générosité. Son ouverture d'esprit, sa disponibilité et ses analyses pertinentes ont contribué à rendre cette étude agréable et enrichissante.

Je tiens à remercier tous ceux qui ont contribué d'une façon ou d'une autre à la réalisation de ce mémoire.

Dédicaces

A MA à ma chère famille.

À tous mes chers amis et mes collègues ;

Et à tous ce qui ont enseigné moi au long de ma vie scolaire

ABSTRACT

In scientific fields, laboratories Experimentation has always been a key concept in the learning process. However, with the explosion of communication technologies, new methods of implementing laboratories have been invented. Virtual or remote laboratories accessible via internet developed for online use have modified the definition of practical experiments, from practical on-site laboratories to virtual/remote online laboratories that are completely independent of the site.

The goal of this work is the design and implementation of an open source solution as a Remote Lab Management Systems management for those laboratories (remote labs).

Keywords:

Remote Laboratories, virtual laboratories, Experimentation, RLMS.

RÉSUMÉ

L'expérimentation dans les domaines scientifiques a toujours été un concept clé dans le processus d'apprentissage. Cependant, avec l'explosion des technologies de l'information et de la communication, des nouvelles méthodes d'implémentation des laboratoires ont été utilisées. Des laboratoires virtuels ou distants accessibles via internet développés afin d'être utilisé en ligne ont modifié la définition des expérimentations pratiques, passant de laboratoires pratiques sur site à des laboratoires en ligne virtuels ou distants totalement indépendants du lieu.

L'objectif de ce travail est la conception et la mise en œuvre d'une solution open source pour la gestion des laboratoires distants et la réalisation d'un prototype système de gestion des laboratoires distants (RLMS *or Remote Laboratories Management System*)

Mots clés

- Laboratoires Distants, Laboratoires Virtual, Expérimentation, RLMS.

SOMMAIRE

LISTE DES FIGURES

LISTE DES ABREVIATIONS

- **IEEE: I**nstitute of **E**lectrical and **E**lectronics **E**ngineers.
- **IoT : I**nternet **o**f **T**hings.
- **RL : R**emote **L**aboratories
- **RLMS : R**emote **L**aboratories **M**anagement **S**ystem.
- **SGLD : S**ystèmes de **G**estion des **L**aboratoires **D**istants.
- **TIC : T**echnologies de l'**I**nformation et de la **C**ommunication.
- **API : A**pplication **P**rogramming **I**nterface.
- **REST : R**epresentational **S**tate **T**ransfer.
- **TLS : T**ransport **L**ayer **S**ecurity.
- **HTTP : H**ypertext **T**ransfer **P**rotocol.
- **CoAP : Co**nstrained **A**pplication **P**rotocol.
- **MQTT: M**essage **Q**ueuing **T**elemetry **T**ransport**.**
- **UX : U**ser e**X**perience.
- **HTML : H**yper**T**ext **M**arkup **L**anguage.
- **CSS : C**ascading **S**tyle **S**heets.
- **JS : J**ava**S**cript.

I. INTORDUCTION GENERALE

L'expérimentation dans les laboratoires dans les domaines scientifiques a toujours été un concept clé dans le processus d'apprentissage. Cependant, avec l'explosion des technologies de l'information et de la communication, des nouvelles méthodes d'implémentation des laboratoires ont été utilisées. Des laboratoires virtuels ou distants accessibles via internet développés afin d'être utilisé en ligne ont modifié la définition des expérimentations pratiques, passant de laboratoires pratiques sur site à des laboratoires en ligne virtuels ou distants totalement indépendants du lieu. (1)

Avec cette nouvelle approche, les étudiants accédant aux laboratoires sans les limites spatio-temporelles. Un autre avantage est la réduction des coûts de déploiement des sur le budget des universités, ainsi la réduction des problèmes de maintenance. (2)

Dans ce travail, nous essayons de donner une étude élargie de ce type des laboratoires, de la façon dont ils sont mis en œuvre, gérés et partagés. Avec un scénario réaliste, en réalisant un laboratoire électronique à distance pour la création et la manipulation des circuits électriques.

Ce travail est divisé en trois chapitres :

- Chapitre 01 : Généralités sur les laboratoires distants

Donne une introduction générale au domaine des laboratoires en ligne en générale est les laboratoires distants plus précisément.

- *Chapitre 02 : Conception et réalisation.*

Dans ce chapitre nous allons présenter et discuter la conception et la réalisation de notre solution proposée.

II. CHAPITRE 01

Généralités sur les laboratoires distants.

1. Introduction

Les laboratoires en lignes (*Remote Laboratories*) RLs ont commencés à se développer dans la fin des années quatre-vingt (3) (4) (5)suite à l'explosion des technologies de l'information et de la communication (TIC) ; Aujourd'hui les RLs son implémentés partout par des universités dans le monde. Dans ce chapitre nous donnons une définition à partir d'une étude tirée de la littérature scientifique sur les RLs.

En effet, Les RLs ont été décrits en détail dans la littérature, en tant qu'outil alternatif aux sessions pratiques (6).Il est généralement classé comme le deuxième outil après celui en présentiel (7), et parfois un emplacement de laboratoires existants (face-à-face) ou en complément des laboratoires *Blended learning* (Modèle hybride d'apprentissage).

2. Classification des laboratoires en ligne

Les laboratoires en ligne peuvent être divisés en deux groupes principaux en basent sur les natures et modèles de conception de ces laboratoires. (6) (8) (9)

- Laboratoire virtuels (Simulation d'une expérience).
- Laboratoire disant (Accès réel aux instruments).

1. **Laboratoires virtuels ou simulations** : où tous les résultats sont calculés d'une manier approximative et non testés dans un environnement réel (par exemple, un simulateur de physique ou électronique, où l'étudiant teste des concepts et le simulateur simule les interactions physiques pour fournir le résultat attendu voir

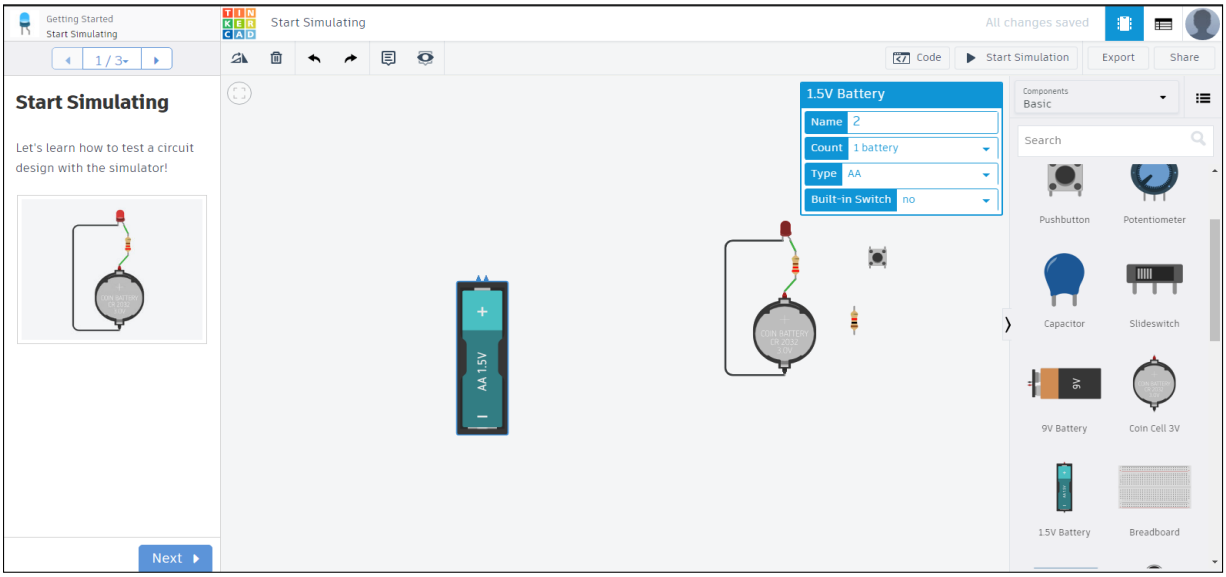


Figure 1 : "Tinkercad Circuits" laboratoire d'électronique virtuel par autodesk.

2. **Laboratoires distants** : où toutes les interactions sont toujours effectuées dans un environnement réel distant. Cela peut être en temps réel (par exemple, un laboratoire interactif) ou de manière asynchrone (par exemple, un laboratoire en batch, où les étudiants soumettent des instructions et que le laboratoire traite).

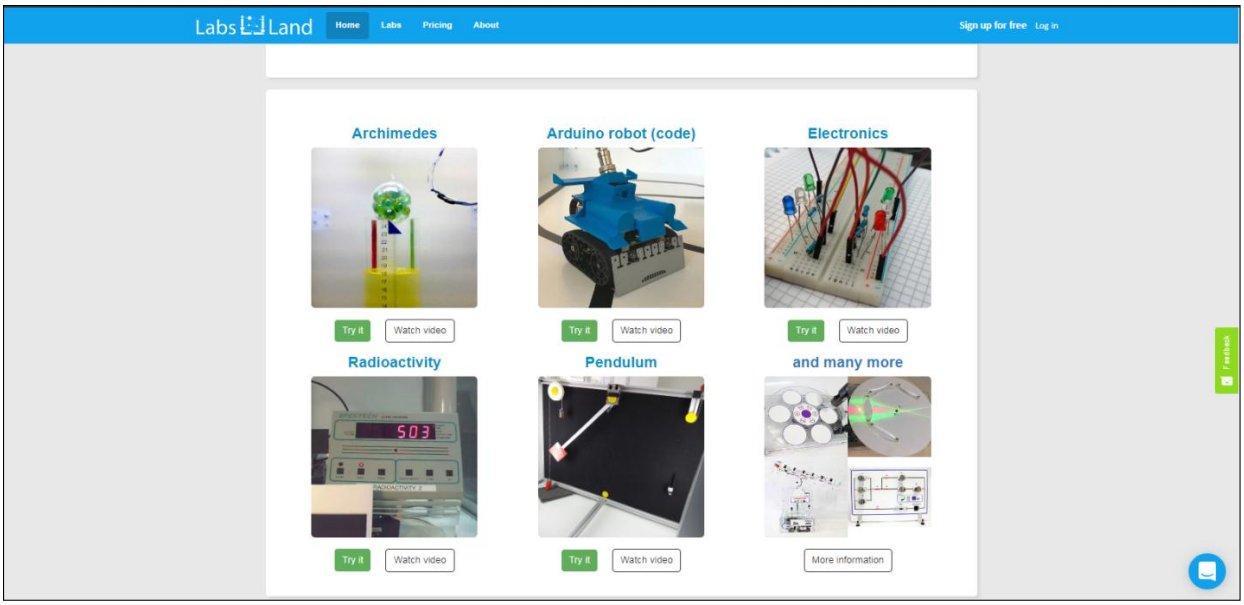


Figure 2 :"LabsLand Portal", portail en ligne pour les laboratoires a distants.

3. Laboratoires distants

Un laboratoire distant est une combinaison d'outils logiciels et matériels qui permet aux étudiants d'accéder et de manipuler à distance des expériences physiques réelles situées dans un laboratoire distant de l'université d'accueil ou dans un local technique relié à Internet. (10)

Ce type de laboratoires vise à fournir aux utilisateurs une expérience indépendante de la localisation et du temps, avec la même qualité que s'ils participaient à une session d'expérimentation en laboratoire réel. Les laboratoires distants ont été considérés comme faisant partie des cinq changements majeurs dans l'éducation en ingénierie. (Numéro spécial du centenaire des travaux de l'IEEE (11), en ce qui concerne l'influence d'informatique et les technologies de la communication)

Aujourd'hui, des laboratoires distants de différentes disciplines ont été déployés par des universités à travers le monde : électronique, robotique, physique, chimie, etc. (8)

3.1. Composition des laboratoires distants

Comme nous l'avons déjà cité, les laboratoires distants combinent deux composants (ou outils) principales différents, un matériel et logiciel.

- **Matériel** : l'environnement physique (matériel) où les expériences seront exécutées par les utilisateurs distants. Ce matériel inclus tous équipements ou matériaux utilise lors de la réalisation ou l'exécution des expériences ; tel que les instruments de mesures, des objets connectés[1] (IoT), les circuits électroniques, des produits chimiques, etc.
- **Logiciel** : Il peut également être appelé système de gestion de laboratoire distants (SGLD)[2]l'environnement ou la partie logique du laboratoire ; inclus tous composant logique utilise lors de la réalisation du labo, tel que l'application permettant l'utilisateur d'accéder au labo (notamment des applications web),les systèmes de gestion utilisateur, gestion d'authentification, gestion des expériences, etc.

3.2. Avantages des laboratoires distants

La disponibilité des ressources de laboratoire pour les étudiants constitue un problème important dans la formation des ingénieurs. D'une part, la gestion de laboratoire peut être coûteuse en ressources, car il nécessite l'entretien et l'évolution continue du personnel et du matériel, de sorte que le nombre de laboratoires est souvent limité, également en raison de facteurs économiques (duplication) des expériences scientifiques.

L'adoption de modes d'accès alternatifs (mode distants) est de plus en plus envisagée par les universités (8). Et l'utilisation d'un réseau informatique pour relier des étudiants

[1]Voir: A. Kalashnikov and H. Zhang. "Remote laboratory: using Internet-of-Things (IoT) for E-learning." (2017).

[2] Discuté en détails les chapitres suivants.

géographiquement éloignés à l'expérience éducative en laboratoire et un aspect très important (3); permettent les ressources du labo d'être accessible à un plus grand nombre d'étudiants.

En réalité, les laboratoires distants peuvent étendre les capacités d'un laboratoire conventionnel en multipliant le nombre des étudiants et le nombre de session par étudiant (12) (13) et en étendant sa disponibilité à plusieurs étudiants (14).En outre, ils ont le potentiel de fournir des données expérimentales abordables en partageant des équipements de laboratoire coûteux au sein d'un groupe d'utilisateurs plus important (15).Néanmoins, les laboratoires distants ne sont pas une solution simple à implémenter et nécessite des ressources humaines.

Un des aspects majeurs des laboratoires distants c'est qu'ils peuvent facilement être partagés[3] entre des universités.

4. Système de gestion des laboratoires distants

Tout laboratoire distant est une combinaison matérielle (hardware) et logiciel (software)[4] ; Nous nous intéressons dans un premier temps par la partie logiciel ou les systèmes de gestion de ces laboratoires. Dans cette partie nous allons essayer d'élaborer un aperçu de la conception de ces systèmes, comment ils sont implémentés, et aussi les différentes fonctionnalités proposées par ces derniers.

Tout laboratoire distant a besoin d'un système de gestion des laboratoires pour assurer la gestion d'authentification, d'autorisation, d'utilisateurs, des expériences, gestion des files d'attentes pour les utilisateurs souhaitent d'accéder au laboratoire, etc. Ces fonctionnalités sont en générale sont indépendante du type de laboratoires et ils sont souvent des fonctionnalités transversales. (16)

Ce système qui offre ces fonctionnalités pour un laboratoire distant est appelé un système de gestion laboratoire distant SGLD (anglais : *Remote Laboratory Management System RLMS*). Ces systèmes sont en général des applications implémenté comme une seule unité – application-global qui gère tous (application « monolithique »[5], en anglais « monolithic»), ou on plusieurs unités -services- coopératives ou chaque unité –service- est responsable de la gestion une partie du fonctionnement total (approche micro-services[6]) [7] ; exemple, un service pour la gestion d'authentification est un autre pour la gestion des expériences ou la gestion des fils d'attende, etc.

---

[3] Discuté en plus de détails dans : 3. Partage des laboratoires distants.

[4]Voir 3.2. Composition des laboratoires distants.

[5]Voir plus sur : https://en.wikipedia.org/wiki/Monolithic_application.

[6]Voir plus sur : https://fr.wikipedia.org/wiki/Microservices.

[7]La différence entre les deux approches est discutée en plus de détails dans :2.2. Les SGLD, une vue conceptuelle (architectural).

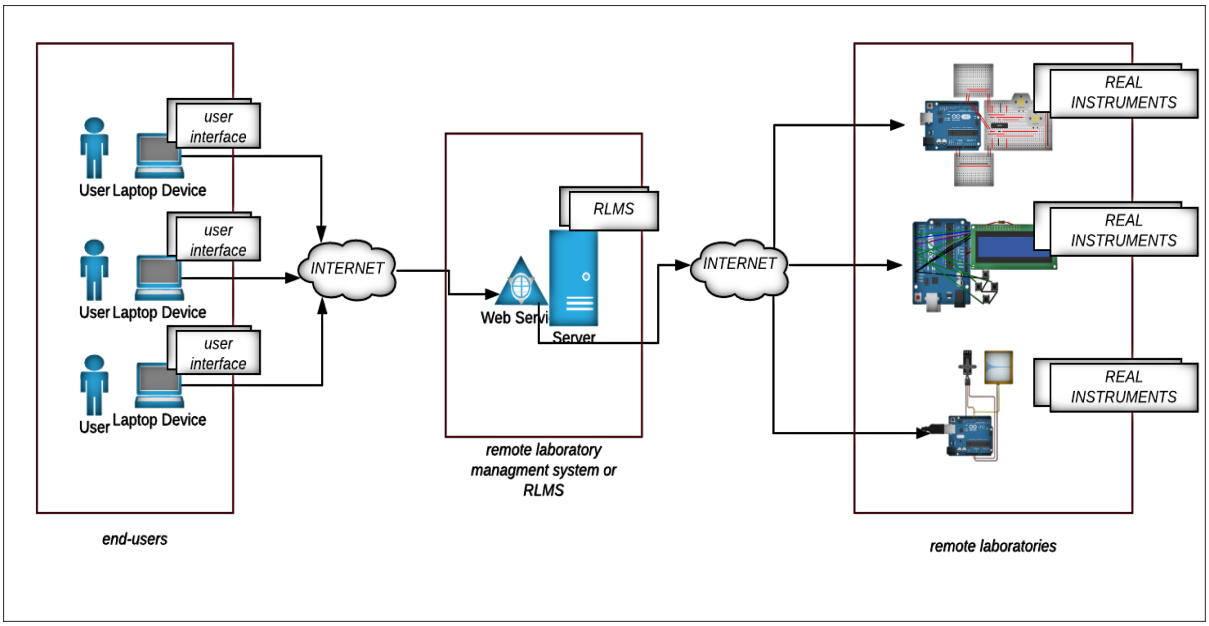


Figure 3 : Architecture générale de laboratoire distants (1).

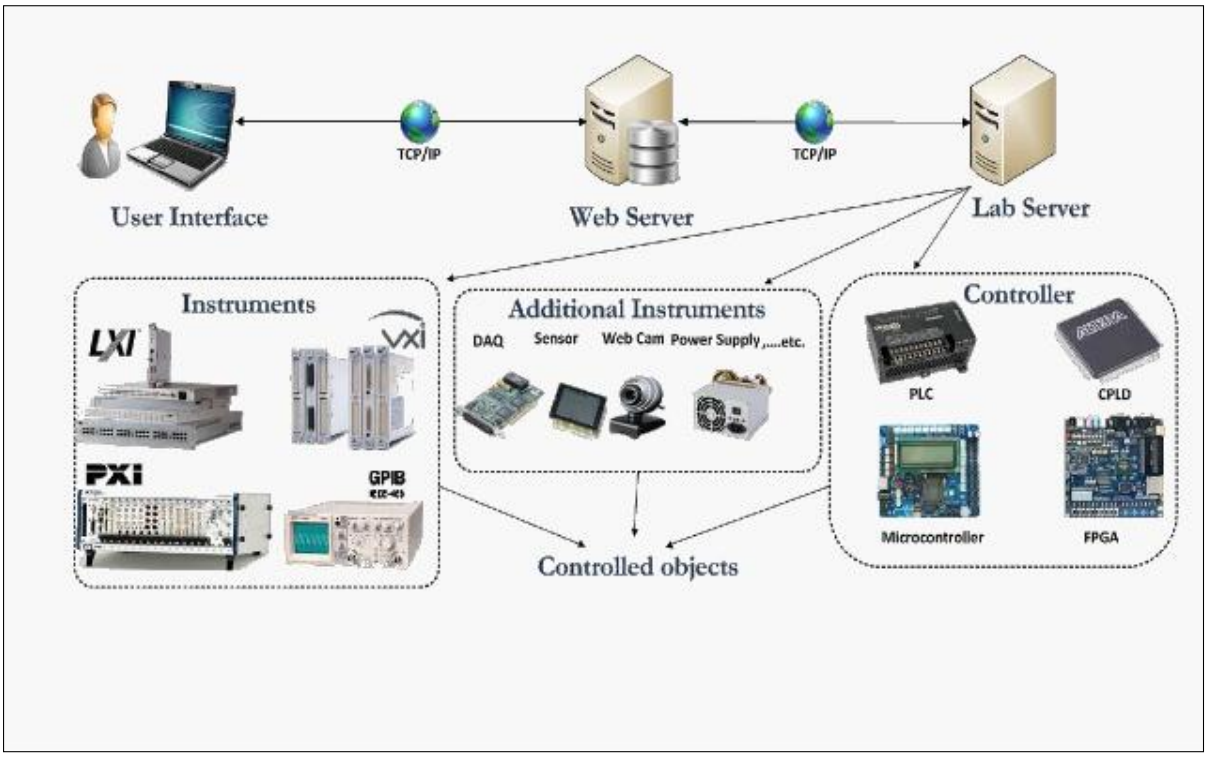


Figure 4: Architecture commune des laboratoires distants(2).

4.1. Les SGLD, une vue fonctionnelle

Dans ce qui suit, nous allons voir les principales fonctionnalités offertes par ces RLMS.

- **Gestion des laboratoires et d'expériences :** un SGLD doit offrir la possibilité de géré plusieurs laboratoires qui peuvent être géo-distribué _ c'est le cas généralement ou plusieurs labos sont maintenu par des différents université ou différentes facultés d'une façon collaborative[8]_, et aussi la possibilité de la gestion des défirent expériences dans ces laboratoires.
- **Gestion d'utilisateurs :** un SGLD doit aussi offrir la possibilité de géré l'utilisation et l'utilisateur veulent manipuler les laboratoires. Cette tâche peut inclus :
    - Inscription : l'inscription des utilisateurs sur le système afin d'exploiter ou utilisé ces laboratoires.
    - Authentification : si l'inscription d'utilisateurs est obligatoire, le SGLD doit posséder un moyen efficace pour authentifie les utilisateurs. Aujourd'hui il existe plusieurs solutions d'authentification des utilisateurs[9] .
    - Gestion des Rôles et des Autorisation : en général les différents utilisateurs ont des diffèrent rôles et des diffèrent autorisation sur les labos. Exemple, dans le cas un universités un utilisateur peut être un enseignant ou un étudiant ; les enseignants peuvent avoir des autorisations spéciales sur les laboratoires tel que l'ajout des nouveaux laboratoires, une tâche qui ne peut pas être réalisé par un étudiant.
    - Gestion des sessions **:** un rôle indispensable du SGLD est la gestion des sessions d'utilisateur sur les laboratoires, surtout dans le cas où il existe un nombre important des utilisateurs veulent y accédez à des nombres limités des laboratoires aux même temps ; dans ce cas le SGLD doit possède un moyen efficace qui le permet de gérer les sessions.
- **Autres :** un SGLD peut aussi inclus autres fonctionnalités complémentaires tel que : l'analyse d'utilisation des labos[10], l'analyse de la progression des étudiants, outil de comptabilité[11], outils d'intégration avec autres plate-forme[12] tel que Moodle[13] par exemple, etc.

Il faut noter qu'en général l'interface graphique (graphical user interface or GUI en anglais) du labo virtuel introduit aux utilisateurs finals ne fait pas partie du SGLD lui-même, mais du labo. Puisque dans la plupart des cas le SGLD est responsable seulement de la gestion des laboratoires et les utilisateurs de ces labos mais pas du fonctionnement du labo lui-même.

---

[8] Voir 3. Partage des laboratoires distants (page 15).

[9] Voir plus sur l'authentification sur : https://fr.wikipedia.org/wiki/Authentification.

[10]Voir : Orduña, Pablo, Aitor Almeida, Diego López-de-Ipiña and Javier García Zubía. "Learning Analytics on federated remote laboratories: Tips and techniques." 2014 IEEE Global Engineering Education Conference (EDUCON) (2014): 299-305.

[11] C'est le cas avec les sociétés qui offrent un accès non gratuit à ce genre des labos.

[12]Voir : Orduña, Pablo et al. "Generic integration of remote laboratories in public learning tools: Organizational and technical challenges." 2014 IEEE Frontiers in Education Conference (FIE) Proceedings (2014): 1-7.

[13]Moodle est une plate-forme d'apprentissage en ligne libre distribuée sous la Licence publique générale GNU écrite en PHP. Développée à partir de principes pédagogiques, elle permet de créer des communautés s'instruisant autour de contenus et d'activités. Voir plus sur : https://fr.wikipedia.org/wiki/Moodle.

Cette approche-là dans plus de flexibilité dans la conception et développement des labos personnalisés intégrables dans le système (SGLD).

4.2. Les SGLD, une vue conceptuelle (architectural)

D'une coté architectural _ ou conceptuelle si c'était vrai à dire_, n'importe quel système informatique[14] est implémenté (architecturé) d'une des deux manier suivant :

L'approche classique, c'est les applications ou systèmes monolithique, ou tous fonctionnalités (logiques) du système sons regroupe et implémenté comme une seule unité logique qui est l'application lui-même. (17)

L'approche moderne, c'est de décomposer le système en plusieurs unités ou sous-systèmes coopératives appelés aussi des micro-services[15]. En générale cette décomposition est par rapport aux différentes fonctionnalités du système ou chaque fonctionnalité est implémenté comme un service indépendant. (17)

[14] En disons « système informatique » nous voulons dire :la partie logique ou applicative du système mais non pas la partie matérielle.

[15]Voir plus sur les microservices sur : https://www.nginx.com/blog/introduction-to-microservices/.

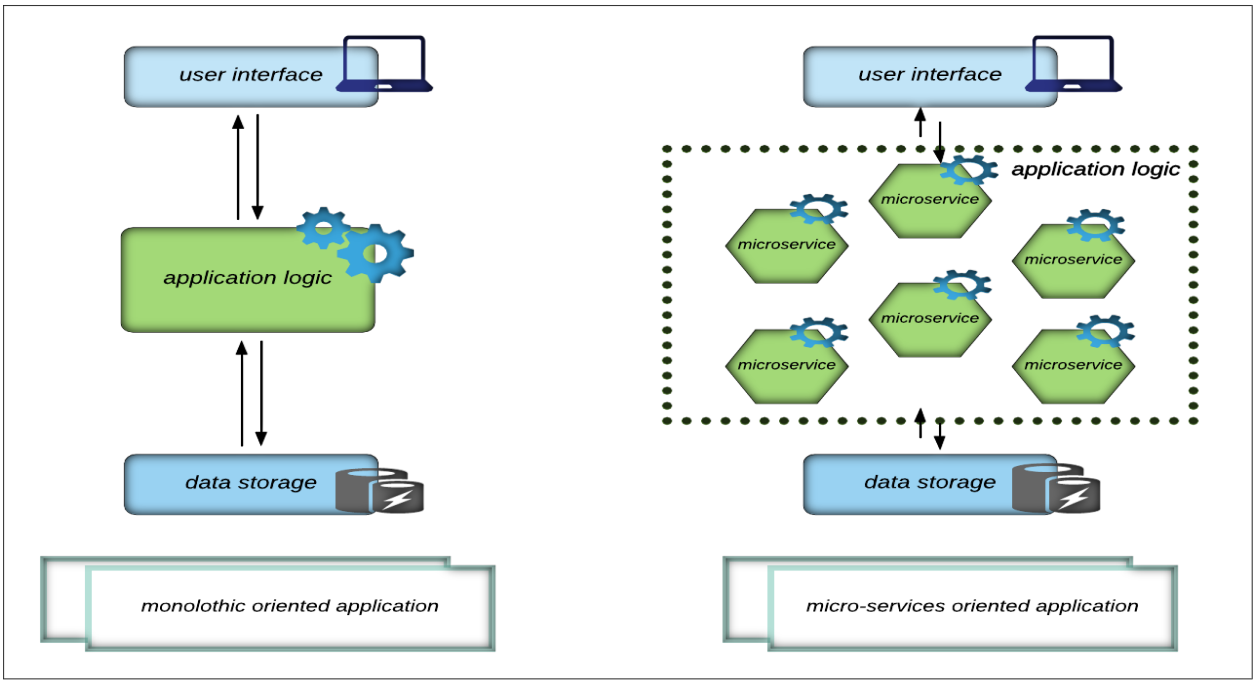


Figure 5 :Comparaison entre les deux approches de conception architecturale d'un système informatique.

4.3. L'implémentation des SGLD

Les systèmes de gestion des laboratoires distants sont généralement implémentés comme des portail web[16] (application web) offrent à l'utilisateur finale plus de flexibilité et plus de portabilités. Cette approche donne à ces systèmes plus flexibilité ou ils peuvent être consulté en ligne via un navigateur web indépendamment du système d'exploitation d'utilisateur (Windows, linux, MacOs, Android, …) et de l'appareil (micro-ordinateur, téléphone portable, Tablet, …).Ce portail qui compte sur des services ou applications du laboratoire distants lui-même (appeler en général Serveur Laboratoire, figure 5) afin de manipuler les laboratoires lors d'un session d'expérimentation (par exemple) offre aussi un outil de gestion centralisé des laboratoires aux personnel responsable de ce processus.

[16] Voir plus sur les portail web sur : https://fr.wikipedia.org/wiki/Portail_web.

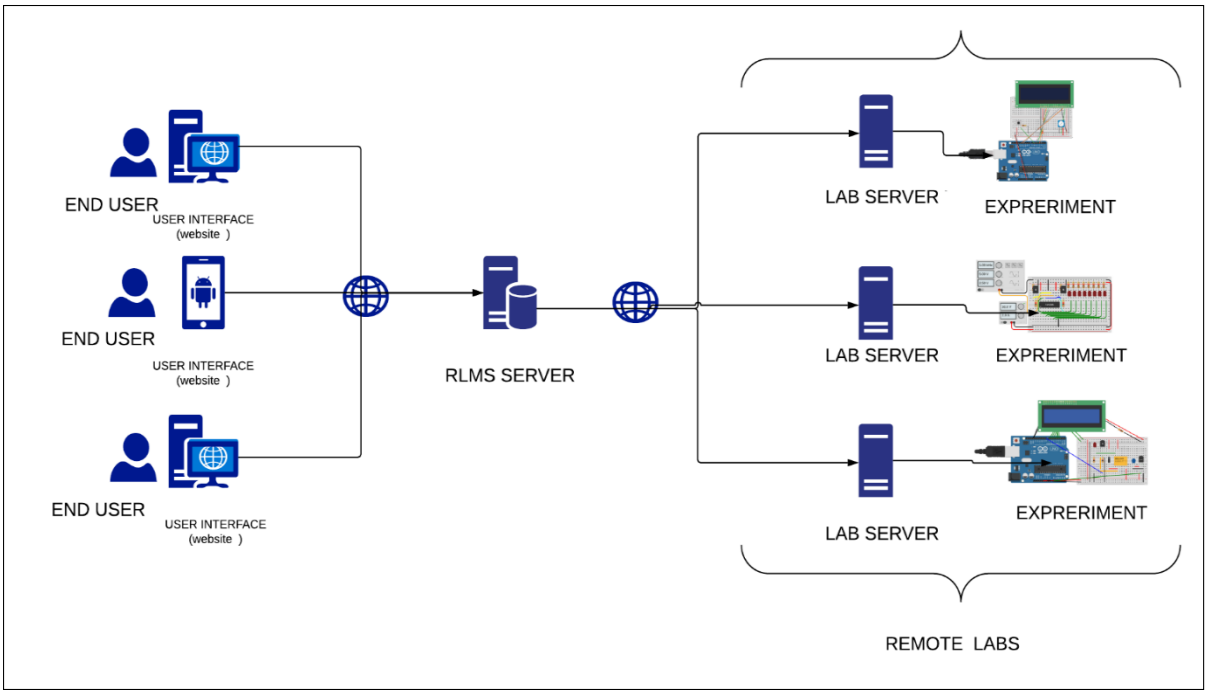


Figure 6: Schématisation d'un exemple d'implémentation d'une solution de gestion des laboratoires distants.

5. Partage des laboratoires distants

Comme nous avons déjà cité précédemment[17], l'un des principaux avantages des laboratoires distants est qu'ils peuvent être partagés avec un public plus large. Cela peut être fait en suivant l'une des trois approches (18) généralement configurable avec le RMLS:

1. Laboratoires à distance public : la première approche consiste à laisser le laboratoire accessible au public à quiconque le souhaite. Cela peut réduire les chances de fournir des analyses de l'apprentissage appropriées ou de prendre en charge des mécanismes de responsabilisation appropriés conduisant à un compromis entre accessibilité et fonctionnalités avancées.

2. Partagez les comptes entre les différents RLMS: si l'Université A souhaité utiliser les laboratoires de l'Université B, une personne de l'Université A fournira une liste de noms d'utilisateur à l'Université B et les étudiants iront dans cet établissement à l'aide les informations d'identification de l'Université B. Idéalement, certains L'authentification fédérée peut être utilisée pour éviter de fournir des informations d'identification dans différents domaines (tels que OAuth[18]), mais ce n'est généralement pas le cas.

[17] Voir chapitre 01 : 3.2. Les intérêts des laboratoires distants (page 09).

[18] Voir plus sur OAuth sur : https://en.wikipedia.org/wiki/OAuth.

3. Laboratoires fédérés: si un RLMS prend en charge la fédération, dans deux institutions différentes (par exemple, l'Université B et l'Université B), les étudiants de l'Université A s'installent et se rendent dans le RLMS de l'Université A et utilisent de manière transparente les laboratoires de l'Université B sous un approche d'établissement-à-établissement (l'Université B n'a donc pas besoin de connaître la liste des étudiants de l'Université A, ni de s'appuyer simplement sur un accord existant avec cette université), c'est généralement le meilleur scénario, et le plus avancé.

6. Exemples des solutions SGLD open sources existantes

6.1. Internet School Experimental System – iSES[19]

Le système expérimental d'école sur Internet iSES (Internet School Experimental System) est un outil complexe pour l'acquisition en temps réel et l'acquisition de données à distance, le traitement de données et le contrôle d'expériences et d'autres processus. Il s'agit d'un système ouvert constitué d'un matériel ISES de base avec le logiciel de contrôle ISESWIN et le logiciel iSES WEB Control Kit pour les laboratoires distants.

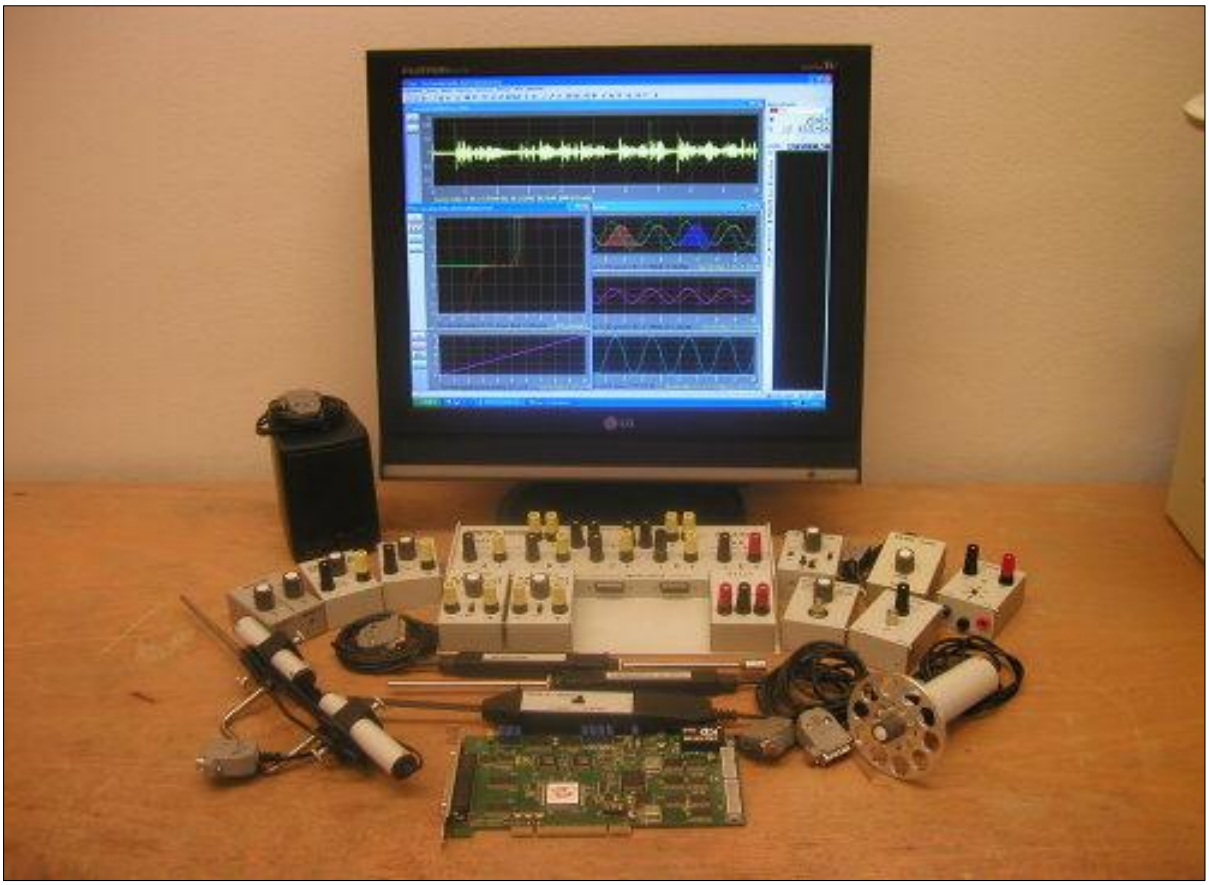


Figure 7:Système iSES, vue global

[19] Site officiel : http://www.ises.info/.

6.2. RemLabNet[20]

Le système de gestion de laboratoire à distance (RLMS), RemLabNet, destiné à l'intégration et à la gestion d'expériences à distance pour les nouveaux étudiants des niveaux universitaire et secondaire. Sa construction a été initiée à la fois par l'utilisation intensive et l'expertise du système ISES (Internet School Experimental System), par la réalisation d'expériences à distance et par l'absence d'un système similaire pour les écoles secondaires en Europe. RLMS est construit à l'aide de nouveaux composants, conçus à cet effet, tels que la gestion de l'espace Web, un entrepôt de données, la carte de communication du RLMS, etc. Le serveur de communication fournit également des services de connexion et de diagnostic, ainsi que des services destinés au confort de l'enseignant (tableau blanc, IP) téléphonie, inclusion de la simulation, gestion des tests et des réservations. Pour des raisons de sécurité, un accès optimal à toutes les expériences et une exploitation économique du nuage virtualisé seront utilisés.

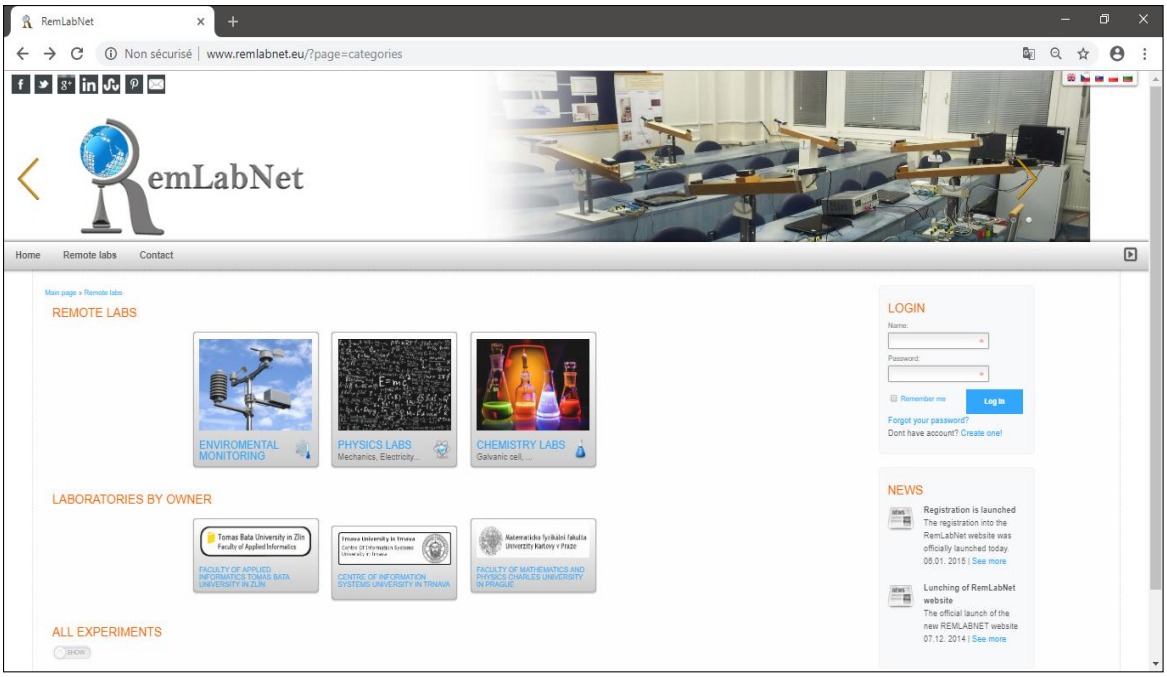


Figure 8:"RemLabNet", page w eb principal.

[20] Site officiel : http://www.remlabnet.eu

6.3. WebLab-Deusto[21]

WebLab-Deusto est une initiative de l'Université de Deusto visant à accroître l'apprentissage par l'expérience par l'utilisation et le développement de laboratoires distants. À cette fin, plusieurs laboratoires sont offerts gratuitement sur Internet, le logiciel sous-jacent est disponible sous licence Open Source et les équipements peuvent être dupliqués.

Figure 9:"WebLab-Deusto", page web principal.

7. Conclusion

Dans ce chapitre, nous avons donné une vue globale sur les généralités des laboratoires en générale et les laboratoires distants en ligne plus précisément, y compris les bienfaits de ces types des laboratoires et comment ils ont participé dans l'amélioration d'expérience éducative des étudiants. Nous avons aussi essayé 'élaborer une vue globale de l'architecture générale des laboratoires distants ou les systèmes de gestions des laboratoires distants, y compris les déférent fonctionnalités traitées par ces systèmes informatiques ainsi que les déférent aspect ces systèmes tels que l'aspect architectural ou conceptuel. etc.

[21] Site officiel : https://weblab.deusto.es

III. CHAPITRE 02

Conception et réalisation.

1. Introduction

Les chapitres antécédents nous ont permis de donner une vue globale sur le domaine des laboratoires distants en général, et les systèmes des gestions de ces laboratoires plus précisément. Dans ce chapitre, nous allons essayer d'élaborer et concevoir notre propre solution de gestion des laboratoires distants en se basant sur des solutions open source existants.

Le résultat final sera un système de gestion de laboratoires open source complet. fronté par une interface Web (application Web) et soutenu par une solution multi-unités pour la gestions du logique applicative (gestion des laboratoires)

2. La Conception de la solution

L'objectif essentiel de ce travail se résume à «la conception et la réalisation d'un système (application) web pour la gestion des laboratoires distants », un système fiable et maniable afin de faciliter les tâches reliées à ce problématique[22] (gestion des laboratoires).

Dans ce qui suit nous allons décrire les contraintes auxquelles est soumis le système pour sa réalisation et son bon fonctionnement. Ces contraintes seront décrites par la suite comme étant des : contraintes fonctionnels ou contraintes non fonctionnels (techniques).

Dans ce qui suit nous allons décrire les acteurs principaux du système, les contraintes auxquelles est soumis le système pour sa réalisation et son bon fonctionnement ainsi que les besoins fonctionnels du système.

2.1. Identification des acteurs

Les acteurs principaux que nous avons réussi à identifier sont :

- **Administrateur système :** c'est l'acteur qui a pour rôle principale d'administrer le système, il est chargé des tâches relié configuration tel que par exemple : l'intégration des nouveaux laboratoires au système. Il est à noter que cet acteur possède un contrôle total sur le système, mais on peut aussi dériver autres acteurs de celui-là avec des autorisations (contrôles) partial. Exemple : un agent avec des autorisations de gestion des certains types laboratoires seulement.

[22] Voir chapitre 02 : 1.1Les SGLD, une vue fonctionnelle

- **Client :** c’est l’utilisateur final des laboratoires.

2.2. Contraintes fonctionnelles

Dans cette section, on va détailler les contraintes fonctionnelles majeures à respecter lors de la conception de la solution.

- La première chose, la solution doit assurer toutes les fonctionnalités basiques de tels systèmes qui peuvent être résumé dans les deux principaux points suivants :
    1. Gestions des laboratoires.
    2. Gestions des utilisateurs des laboratoires.
- Autre contrainte, l’intégration des nouveaux labos doits être assuré. On parle d’une intégration simple car ce genre des solutions est plutôt une solution très évolutive (scalable) ou des nouveaux laboratoires peuvent être ajoutés quotidiennement.
- Autre contrainte à prendre en compte c’est la scalabilité est un facteur très important qu’il faut prendre en considération lors de la conception de cette solution. Comme nous l'avons déjà mentionné le point précédent, ce genre de systèmes est un système très évolutif. Donc il doit être conçu d’une manière optimisée à être évolué. Et ça c’est un des réseaux pour lesquelles nous avons opté l’architecture micro-services qui est très optimisé pour les systèmes évolutifs.
- Autrement, l’expérience utilisateur elle doit être optimale et très facile à maitriser. On parle de « User Experience (UX) » car notre système est plutôt complexe.

2.3. Contraintes techniques

Dans cette section, nous allons parler des majeures contraintes techniques (non fonctionnels) qu’on a prises en compte pour développer la solution.

- Dans un premier temps, la solution proposée doit pour assurer le maximum de la disponibilité et de portabilité aux utilisateurs c’est pour cela nous avons opté une solution web (portail web).
- La deuxième contrainte, c’est l’interface de l’application qui doit être simple et ergonomique.
- Autre contrainte, ce genre des systèmes est plutôt complexe à mettre en pratique, c’est pour cela qu’il faut assurer qu’il est conçu d’une manière maintenable. Pour assurer ça nous a opté l’architecture micro-services lors de la conception de la solution.
    - Autres aspects très importants, la solution proposée doit assurer que les laboratoires intégrés (gérées) peuvent être développés séparément de la solution de gestions qui va donner plus de flexibilité aux processus de développement des laboratoires.
    - Autrement, la sécurité qu'il faut prendre en charge lors de la conception de la solution.

2.4. Besoins fonctionnels

Dans cette section, nous allons détailler les principaux besoins fonctionnels à respecter lors de la conception de notre solution.

- Gestion des laboratoires :
  - La solution permet d'ajouter, supprimer, modifier des laboratoires.
- Gestion des utilisateurs des laboratoires :
  - La solution permet d'inscrire, modifier, supprimer des nouveaux profiles des utilisateurs.
- Gestion des catégories et instances des laboratoires :
  - La solution permet de regrouper les laboratoires sous des catégories.
  - La solution permet la gestion des copiés les laboratoires (dans le cas où il existe plusieurs instances du même labo).
- Gestion des réservations des laboratoires
  - La solution permet l'utilisateur a explorer, consulter et réserver des sessions sur les laboratoires gères.

2.5. Modélisation des besoins fonctionnels

La recherche ciblée des besoins fonctionnels, est considérée nécessaire avant d'entamer la conception afin d'obtenir une vue globale sur les exigences de notre système.

2.5.1. Etude des cas d'utilisations

Le diagramme de « use case UML[23] (19) (Unified Modeling Language) » est utilisé pour la modélisation des différents cas d'utilisations de notre système.

Avant d'entamer les diagrammes, intéressons-nous au terme cas d'utilisations, Un cas d'utilisation est une unité cohérente représentant une fonctionnalité visible de l'extérieur. Il réalise un service de bout en bout, avec un déclenchement, un déroulement et une fin, pour l'acteur qui l'initie. Un cas d'utilisation modélise donc un service rendu par le système, sans imposer le mode de réalisation de ce service. (20)

Diagramme de cas d'utilisations. La figure suivante représente le diagramme de cas d'utilisation des principales tâches sur notre système.

[23] UML est un langage formel et normalisé en termes de modélisation objet. Son indépendance par rapport aux langages de programmation, aux domaines de l'application et aux processus, son caractère polyvalent et sa souplesse ont fait lui un langage universel.

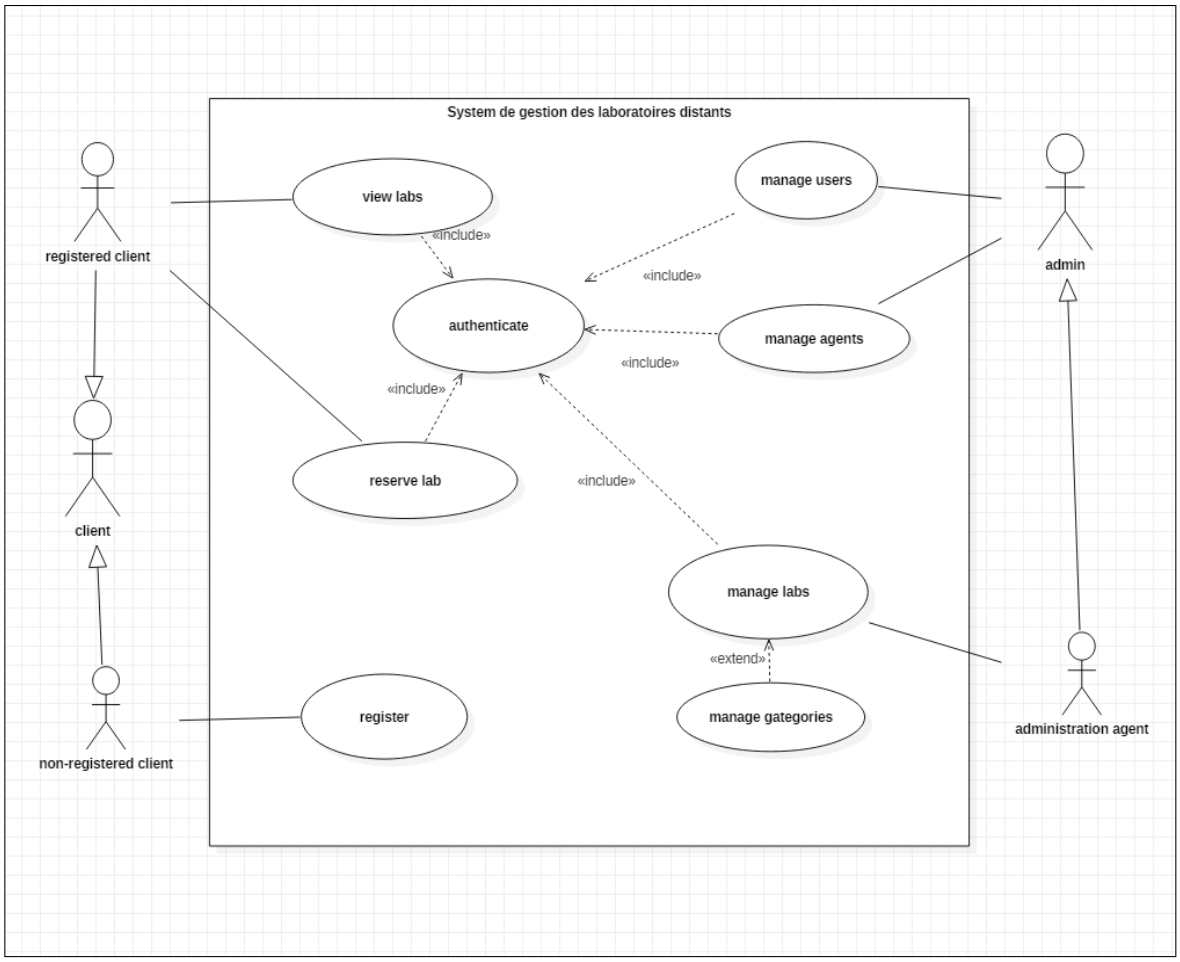


Figure 10: diagramme de cas d'utilisation.

- L’administrateur a comme rôle principale de gérer toutes les taches de la plate-forme ; gestion des laboratoire, agents administratifs, utilisateurs.
- L'administrateur attribue les laboratoires déjà créer a des agents administratifs.
- L’administrateur peut ajouter (intégrer), supprimer ou modifier des laboratoires.
- L’administrateur peut ajouter, supprimer ou modifier des profils d’agents administratifs.
- L’administrateur peut ajouter, supprimer ou modifier des profils d’utilisateurs.
- L’agent administratif peur consulter, modifier des laboratoires qu’ils ont lui a été attribué par l’administrateurs.
- L’utilisateur peut inscrire, modifier ou supprimer son profile.
- L’utilisateur peut consulte la liste des laboratoires.
- L’utilisateur peut réserver une session sur un laboratoire.

3. Présentation de la solution proposée

Après avoir donné une vue globale sur les principales lignes à respecter lors la conception de notre solution, nous allons présenter la solution finale que nous proposons. Mais avants de

la présenté nous allons d'abord discuter les choix techniques (tel l'intégration que projets déjà existants) architecturaux et nous avons opté.

Lors de la phase de conception de notre solution nous avons conclu que la problématique globale lié à notre thématique (gestion des laboratoires distants) peut être résumé sous les deux sous problèmes (taches) :

1. Gestion des laboratoires
2. Gestion des utilisateurs des laboratoires

Pour la premier (Gestion des laboratoires), et vue que la plupart des laboratoires distants sont basés (implémenté) sur des objets connecté (IoTs) ; nous avons décidé d'utiliser une solution existante qui peut servir notre objectif. Pour cela et après avoir étudier plusieurs solution open source existants nous avons choisi « **ThingsBoard**[24] **»**, une plate-forme open source pour la gestion des IoTs.

Pour la deuxième (Gestion des utilisateurs des laboratoires), nous avons développé notre propre solution. Une plate-forme (portail) web avec tout un système de gestion d'utilisateurs basique (inscription, modification profile, …) et avec la possibilité d'exposé les laboratoires du premier plate-forme (Thingsboard). Nous avons la donnée nom « **LabsUi** ».

La relation entre les deux plateformes a été rendu possible grâce à une troisième composant intermédiaire que nous avons développé à servir comme une API REST[25] intermédiaire entre les deux. Nous avons le donnée nom « **Resty** ». Resty gère non pas la relations entre les deux premiers composent seulement, mais aussi la relation entre le portail web (LabsUi) et les laboratoires distants dans certains cas tel que la réservation des sessions sur un labos.

Les relations (interactions) entre les différents composent possible est mis possible via des interface REST[26] ou chaque composant possède un API qui le permet à offrir ces services aux autres composants.

Ces API sans sécurisés grâce à l'utilisation d'authentification via des jetons JWT[27] (Json Web Token).

Il est à noter que la solution que proposons ne vise que l'aspect "gestion et publication des laboratoires distants" du problématique "**Laboratoires distants"**. Cependant qu'ils y autres aspects qui doivent être géré par les laboratoires elles-mêmes[28] ; tel que la gestion des sessions

[24] https://thingsboard.io.

[25] Representational State Transfer est un style d'architecture logicielle définissant un ensemble de contraintes à utiliser pour créer des services web. Plus d'infos sur : https://fr.wikipedia.org/wiki/Representational_state_transfer.

[26] Discuté en : 3.5 Interactions entre les composants système

[27] Plus sur ce notion sur : https://fr.wikipedia.org/wiki/JSON_Web_Token.

[28] Discuter en plus de détails dans : 3.2 Développement laboratoires.

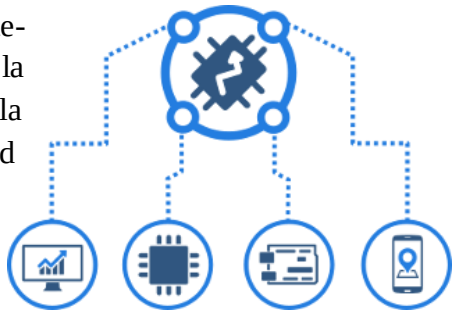
sur un laboratoires qu'il doit être géré par le labo lui-même ou dans tels cas où notre système est responsable seulement de la réservation du ces sessions sur le labo, mais les sessions elles-mêmes sont généré et géré par le laboratoire lui-même.

3.1. Gestions des laboratoires (ThingsBoard)

Comme nous avons déjà cité, pour la gestion des laboratoires nous avons choisi la plate-forme open source ThingsBoard. Par la suite nous allons donner une vue globale sur cette plate-forme, pourquoi nous l'avons choisi et comment nous l'avons intégré dans notre projet.

3.1.1. ThingsBoard

Selon la définition officiel, ThingsBoard est une plate-forme IoT open source pour la collecte, le traitement, la visualisation et la gestion de périphériques. Il permet la connectivité des périphériques via les protocoles IoT standard : MQTT[29], CoAP[30] et HTTP[31] et prend en charge les déploiements sur le cloud et sur site. ThingsBoard associe l'évolutivité, la tolérance aux pannes et la performance.



3.1.2. ThingsBoard principaux Caractéristiques et fonctionnalités

- **Provisionner et gérer des périphériques :** ThingsBoard vous donne la possibilité de provisionner, surveiller et contrôler vos entités IoT de manière sécurisée à l'aide de riches APIs côté serveur. Définissez les relations entre vos appareils, vos clients ou toute autre entité.
- **Collecter et visualiser les données :** vous donne la possibilité de collecter et stocker les données des périphériques de manière évolutive et tolérante aux pannes. Visualiser les données avec des widgets intégrés ou personnalisés et des tableaux de bord flexibles. Partagez des tableaux de bord avec les clients de la plateforme.
- **Traiter et réagir :** Grâce à son Moteur de règles (Rule Engine)[32], ThingsBoard vous donne la possibilité de Définir les chaînes de règles de traitement de données. Transformer et normaliser les données des appareils.

[29] Message Queuing Telemetry Transport est un protocole de messagerie publish-subscribe basé sur le protocole TCP/IP, Plus d'infos sur : https://fr.wikipedia.org/wiki/MQTT

[30] Constrained Application Protocol) est un protocole de transfert Web optimisé pour les périphériques et réseaux contraints, Plus d'infos sur :

[31] Hypertext Transfer Protocol est un protocole de communication client-serveur développé pour le World Wide Web, Plus d'infos sur : https://fr.wikipedia.org/wiki/Hypertext_Transfer_Protocol

[32] Plus sur le Moteur de règles sous ThingsBoard sur : https://thingsboard.io/docs/user-guide/rule-engine-2-0/overview/

Générer des alarmes sur les événements de télémétrie[33] entrants, les mises à jour d'attributs[34], l'inactivité des périphériques et les actions de l'utilisateur.

- **Gestion des alarmes :** ThingsBoard Offre la possibilité de créer et de gérer des alarmes liées à vos entités : appareils, actifs, clients, etc. Permet la surveillance des alarmes en temps réel et la propagation d'alarmes vers la hiérarchie des entités associées. Relever les alarmes lors d'événements de déconnexion ou d'inactivité de l'appareil.
- **API administrative :** ThingsBoard Offre un API administrative REST qu'il peut être utilisé pour faire tout ce qui est possible via l'interface utilisateur graphique de manière programmatique, ce qui est un point très important pour l'intégrations de ThingsBoard dans d'autres projets plus importants comme notre cas.
- **Microservices ou monolithiques :** ThingsBoard prend en charge le déploiement monolithique pour les environnements de petite taille, ainsi que la possibilité de passer à des déploiement microservices pour une haute disponibilité et une évolutivité horizontale.
- **Installations multiples :** ThingsBoard prend en charge des installations multi-locataires prêtes à l'emploi. Un seul locataire peut avoir plusieurs administrateurs de locataires et des millions de périphériques et de clients.
- **Evolutivité horizontale :** Si la quantité de requêtes côté serveur et de périphériques pris en charge augmente de manière linéaire, des nouveaux serveurs Thingsboard peuvent être ajoutés en mode clustering. Pas de temps d'arrêt, de redémarrage du serveur ou d'erreurs d'application.
- **Tolérance aux pannes :** Tous les serveurs ThingsBoard sont identiques. Pas de maître-ouvriers relation ni de réserve chaude (Hot standby)[35]. La défaillance d'un nœud est automatiquement détectée. Les nœuds défaillants peuvent être remplacés sans temps d'arrêt. Les données persistantes sont automatiquement répliquées.
- **Sécurité :** ThingsBoard prend en charge le Sécurité de la couche de transport[36] ou Transport Layer Security (TLS) pour les protocoles MQTT et HTTP (s). Prend en charge l'authentification et la gestion des informations d'identification de l'appareil.
- **Open-source :** ThingsBoard est sous licence Apache 2.0[37], vous pouvez donc l'utiliser gratuitement dans vos produits commerciaux.

[33] Plus d'information sur cette notion sure : https://en.wikipedia.org/wiki/Telemetry.

[34] Pans ThingsBoard les attributs sont des données statiques ou semi-statiques peuvent être liées aux périphériques, exemple : version du logiciel installé sur le périphérique, identifiant personnalisé, etc.

[35] Plus d'information sur cette notion sure : https://www.techopedia.com/definition/1024/hot-standby.

[36] Plus d'information sur cette notion sure : https://fr.wikipedia.org/wiki/Transport_Layer_Security.

[37]Plus d'information : https://fr.wikipedia.org/wiki/Licence_Apache.

3.1.3. Installation

ThingsBoard peut être installé et exécuté sur divers systèmes d'exploitation (Windows, Linux, Mac, ...) et sur l'environnement (système, machines virtuelles, Docker[38], ...). Le processus d'installation du serveur ThingsBoard est une opération très simple.

Suivant la documentation officielle[39], nous avons pu l'installer sur un système client Ubuntu sous Docker, en suivant les étapes suivantes :

1. Installation Docker :

```
1: sudo apt-get update
2: sudo apt-get install docker-ce docker-ce-cli containerd.io
```

2. Installation ThingsBoard

```
3: docker run -it -p 9090:9090 -p 1883:1883 -p 5683:5683/udp -v ~/.mytb-data:/data
-v ~/.mytb-logs:/var/log/thingsboard --name mytb --restart always thingsboard/tb-
cassandra
```

Où :

- docker run - lance ce conteneur
- -it - attache une session de terminal avec la sortie actuelle du processus ThingsBoard
- -p 9090: 9090 - connecte le port local 9090 au port HTTP interne exposé 9090
- -p 1883: 1883 - connecte le port local 1883 au port MQTT interne exposé 1883
- -p 5683: 5683 - connecte le port local 5683 au port COAP interne exposé 5683
- -v ~ /.mytb-data/data - monte le répertoire de l'hôte ~/.mytb-data dans le répertoire de données ThingsBoard DataBase
- -v ~/.mytb-logs: / var / log / thingsboard - monte le répertoire de l'hôte ~/.mytb-logs dans le répertoire des journaux ThingsBoard.
- --name mytb - nom local convivial de cette machine.
- --restart always - démarre automatiquement ThingsBoard en cas de redémarrage du système et redémarre en cas d'échec.
- Thingsboard/tb-cassandra - image de menu fixe, peut également être Thingsboard /tb-postgres ou Thingsboard/tb.

[38] Docker est un logiciel libre permettant facilement de lancer des applications dans des conteneurs logiciels. Plus d'info sur le site officiel : https://www.docker.com.

[39] Sur : https://thingsboard.io/docs/installation.

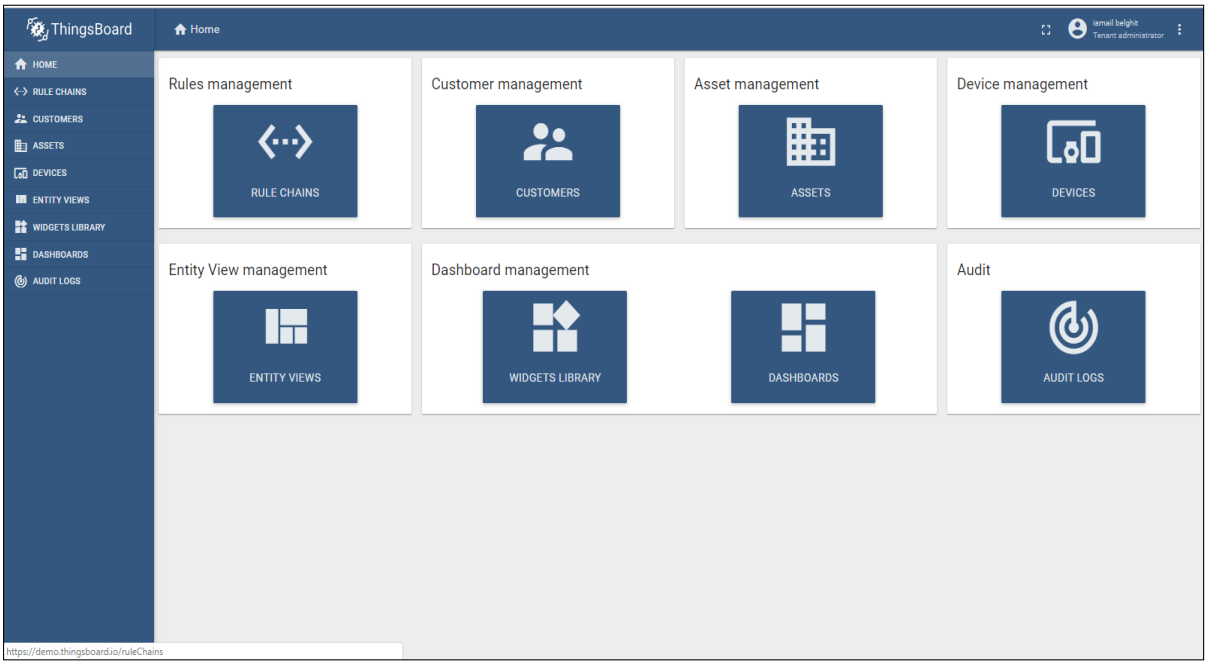


Figure 11 : Tableau de bord principale sous ThingsBoard.

3.1.4. Notions fondamentales sous ThingsBoard

Comme nous l'avons déjà mentionné, nous avons décidé d'opter pour la plateforme ThingsBoard pour la gestion des laboratoires dans notre solution proposée. Par la suite nous allons citer les principales notions sous cette plateforme.

- **Entités :** Sous ThingsBoard vous pouvez provisionner et gérer plusieurs types d'entités et leurs relations dans votre application IoT. Les entités prises en charge sont :
  - *Locataires (Tenants)* : vous pouvez traiter le locataire en tant qu'entité commerciale distincte : personne ou organisation possédante ou produisant des périphériques et des actifs ; Le locataire peut avoir plusieurs utilisateurs d'administrateur locataire et des millions de clients.
  - *Clients (Customers)* : le client est également une entité commerciale distincte : personne ou organisation qui achète ou utilise des appareils locataires et / ou des assets (Anglais) ; Le client peut avoir plusieurs utilisateurs et des millions de périphériques et / ou d'assets ;
  - *Utilisateurs (Users)* : les utilisateurs peuvent parcourir des tableaux de bord et gérer des entités.

- *Périphériques (Devices)* : entités IoT de base pouvant générer des données de télémétrie et gérer des commandes RPC (Remote Procedure Call)[40]. Par exemple, des capteurs, des actionneurs, des commutateurs, ou dans notre cas des laboratoires.
- *Assets :* entités abstraites IoT pouvant être liées à d'autres périphériques et assets. Par exemple usine, champ, véhicule.
- *Alarmes (Alarms)* : événements qui identifient des problèmes avec vos assets, appareils ou autres entités.
- *Tableaux de bord (Dashboard)* : Visualisation de vos données IoT et possibilité de contrôler des périphériques particuliers via des interface graphique utilisateur.
- *Nœud de règle (Rule Node)* : unités de traitement des messages entrants, des événements de cycle de vie d'entité, etc.
- *Chaîne de règles (Rule Chain)* : unité logique des nœuds de règle associés.

Chaque entité supporte :

- **Attributs (Attributes)** : Paires clé-valeur statiques ou semi-statiques associées à des entités. Par exemple, numéro de série, modèle, version. ThingsBoard offre la possibilité d'attribuer des attributs personnalisés à vos entités et de gérer ces attributs. Les attributs sont séparés en trois groupes principaux :
  - *Attributs côté serveur* : les attributs sont rapportés et gérés par l'application côté serveur (ThingsBoard). Non visible pour l'application de l'appareil (Périphérique). Certaines données secrètes peuvent être utilisées par les ThingsBoard, mais ne doivent pas être disponibles pour le périphérique. Toute entité ThingsBoard prend en charge les attributs côté serveur : périphérique, ressource, client, locataire, règles, etc.

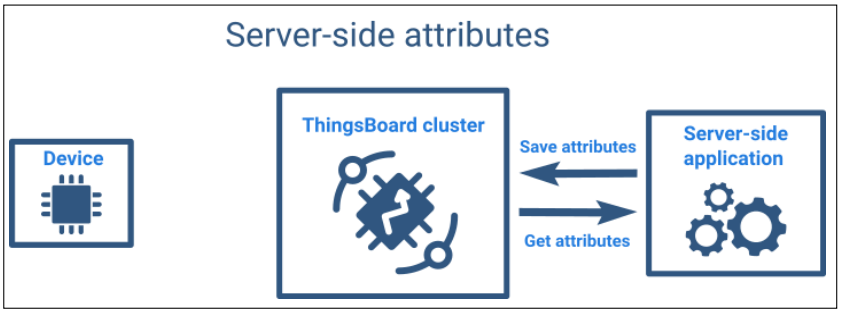


Figure 12: Représentation shematiques des Attributs côté serveur

[40] Pus sur cette notion sure : https://fr.wikipedia.org/wiki/Remote_procedure_call.

- *Attributs côté client* : les attributs sont signalés et gérés par l'application du périphérique. Par exemple, version actuelle du logiciel / du micrologiciel, spécification du matériel, etc.

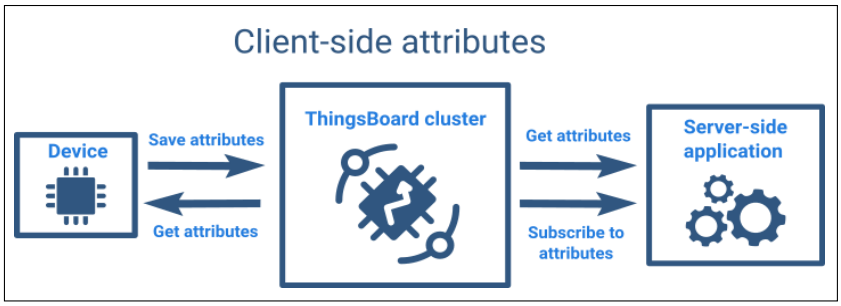


Figure 13: Représentation shematiques des Attributs côté client.

- *Attributs partagés* : les attributs sont signalés et gérés par l'application côté serveur. Visible pour l'application de l'appareil. Par exemple, abonnement client, version du logiciel (firmware) cible.

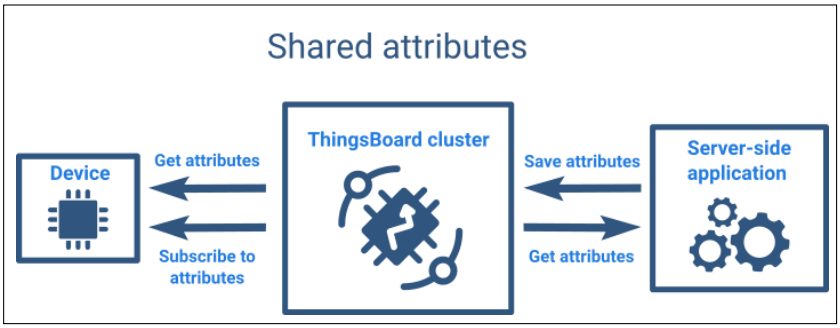


Figure 14: Représentation shematiques des Attributs partagés

- **Données de télémétrie (Telemetry data)** : Points de données de série chronologique (Timeseries) disponibles pour le stockage, l'interrogation et la visualisation. Par exemple, la température, l'humidité, le niveau de la batterie.
- **Relations** : Connexions dirigées vers d'autres entités. Par exemple, contient, gère, possède, produit.
- **Vues d'entité (Entity Views)** : Comme pour les vues de base de données SQL, qui limitent le degré d'exposition des tables sous-jacentes au monde extérieur, les vues d'entités ThingsBoard limitent le degré d'exposition de la télémétrie du périphérique ou de l'assets et des attributs aux clients.

De plus, les périphériques et les assets ont également **un type**. Cela permet de les distinguer et de traiter les données d'une manière différente.

3.1.5. Manipulations basiques sous TingsBoard

Nous allons voire par la suite les basiques opérations peuvent être exécuté sous ThingsBoard.

Il est à noter que nous avons baser seulement sur les opérations qui vont nous servir lors de l'intégration de TingsBoard dans notre solution.

- **Provisionnement des appareils/assets :** sous ThingsBoard (TB) provisionnement des nouveaux appareils il suffit de :
  1. Création d'un nouvel appareil (device) : Pour créer un appareil, cliquez d'abord sur DEVICES, puis sur le signe plus. (Voir figure 12)

Pour les assets l'exercice est très similaire, il suffit juste d'exécuter les mêmes étapes dans sous l'onglet ASSETS.

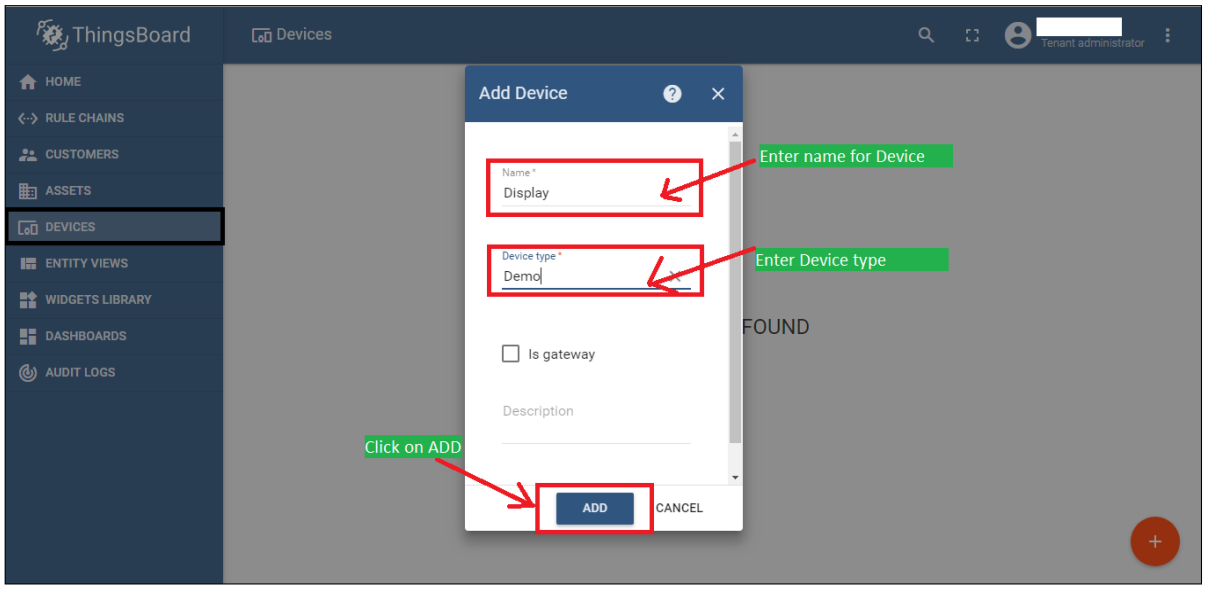


Figure 15:Création d'un nouvel appareil sous ThingsBoard .

2. Connection de l'appareil à TingsBoard : connecter l'appareil à TingsBoard en utilisant le jeton généré lors de la création[41] de l'appareil. ThingsBoard prend en charge les protocoles suivants pour la connectivité des périphériques[42] :
    - MQTT
    - CoAP
    - HTTP

Pour notre cas exemple, nous supposons que le périphérique est déjà créé (suivant l'étape 1) dans un serveur ThingsBoard installe est accessible via l'adresse : *http://localhost:8080*, nous peuvent donc tester la connexion via le protocole HTTP en utilisons l'outil cURL[43] pour simuler l'envoi des **Attributs côté client**.

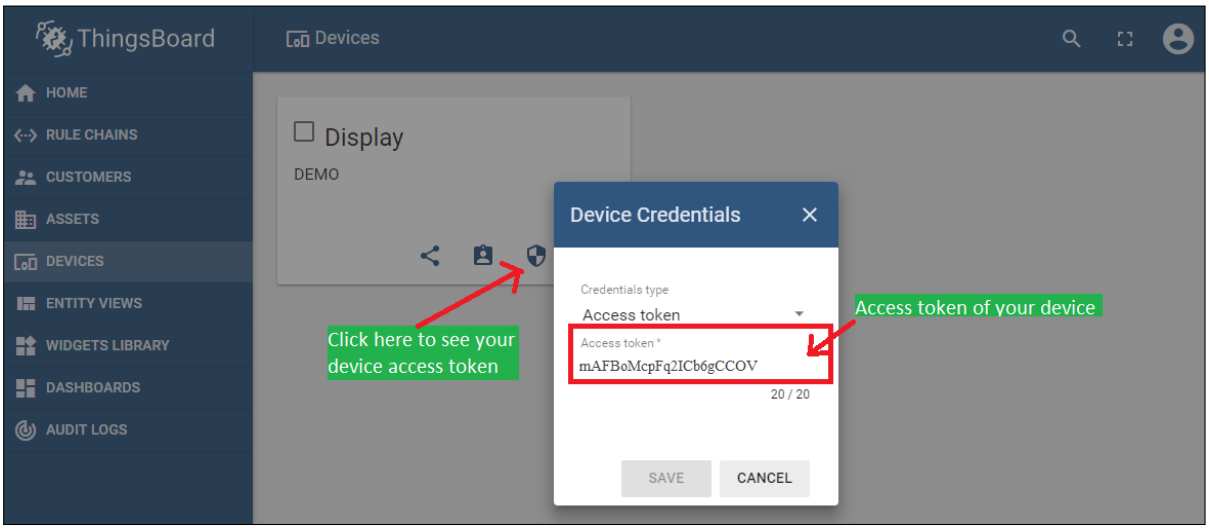


Figure 16:Récupération du jeton de connexion d'un prepherique sous ThingsBoard.

[41] Noter que chaque fois un nouvel appareil est créé. ThingsBoard générera automatiquement un jeton utilisé pour authentifier la connexion de l'appareil réel au serveur ThingsBoard. Voir plus sur le provisionnement des appareils sous ThingsBoard sur : https://thingsboard.io/docs/getting-started-guides/connectivity/

[42] Plus d'information sur la connectivité des périphériques sous ThingsBoard sur : https://thingsboard.io/docs/reference/protocols/

[43] cURL (ou CURL) est un outil qui permet d'envoyer des commandes HTTP sous la ligne de commande, plus d'infos sur cette outil sur : https://curl.haxx.se/

La command sera:

```
#new-attributes-values.json is a JSON data file with data you want to publish:
{"attribute1":"value1", "attribute2": true, "attribute3":42.0, "attribute4":73}

1: Curl -v -X POST -d @new-attributes-values.json
http://localhost:8080/api/v1/mAFBoMcpFq2ICb6gCCOV /attributes --header "Content-
Type:application/json"
```

- **Manipulation des attributs des appareils :** concernant les attributs il y a trois manipulations majeures :
    1. *Publier/mise-à-jour d'attribut sur le serveur* : les attributs coté clients peuvent seulement Publier/mise-à-jour par le client lui-même, sinon tout autre type est créé/mis-à-jour directement sur TingsBoard,

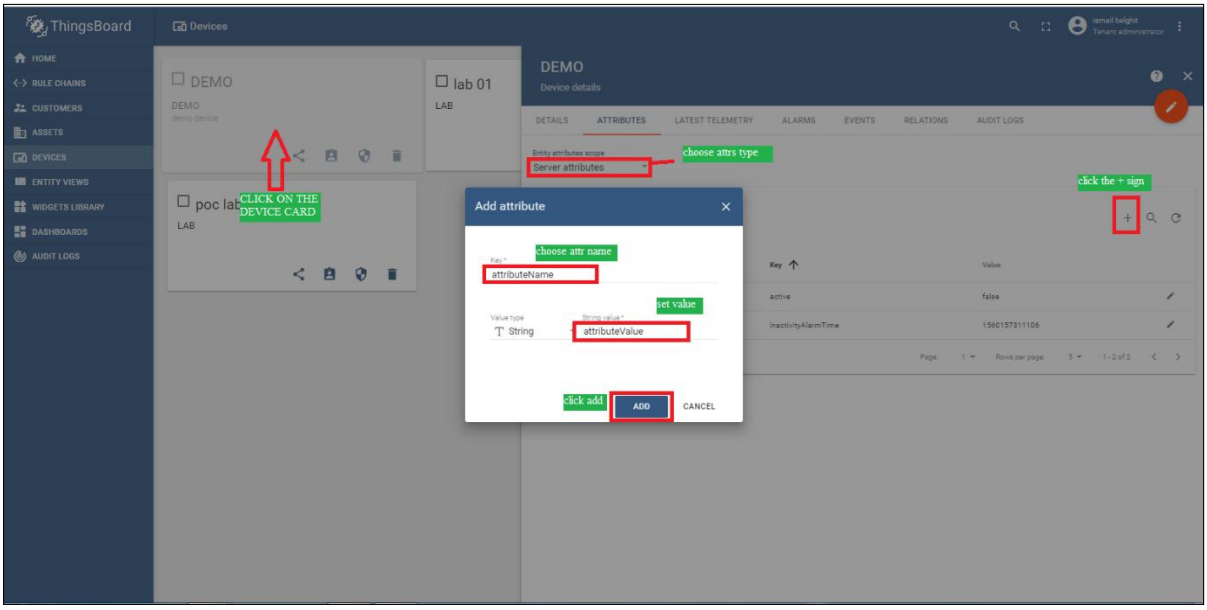


Figure 17:Manipulation des attributs sous ThingsBoard.

Pour un exemple sur la manipulation des attributs coté client consulté la partie 2 du point précédent (Provisionnement des appareils).

```
1: Curl -v -X GET
http://localhost:8080/api/v1/$ACCESS_TOKEN/attributes?clientKeys=attribute1,attrib
ute2&sharedKeys=shared1,shared2
```

2. *Demander des valeurs d'attribut au serveur* : Afin de demander des attributs de périphérique côté client ou partagés au nœud de serveur ThingsBoard, il suffit d'envoyer la demande GET à l'URL suivante *:*

Où :

- *$ACCESS_TOKEN* : le jeton de connexion de l'appareil.
- attribute1, attribute2 : des attributs coté client a demandé.
- shared1, shared2: des attributs partage a demandé.

```
# Send subscribe attributes request with 20 seconds timeout

1: curl -v -X GET
http://localhost:8080/api/v1/$ACCESS_TOKEN/attributes/updates?timeout=20000
```

3. *Abonner au mises à jour des attributs* : Concerne seulement les attributs partagés, Pour vous abonner aux modifications d'attributs de périphériques partagés, envoyez une demande GET avec le paramètre de demande facultatif « timeout » à l'URL suivante :

```
#content of the file telemetry-data-as-object.json : {"key1":"value1",
"key2":"value2"}

1: curl -v -X POST -d @telemetry-data-as-object.json
http://localhost:8080/api/v1/$ACCESS_TOKEN/telemetry --header "Content-Type:
application/json"
```

- **Publier des Données de télémétrie** : Pour publier des données de télémétrie sur le nœud du serveur ThingsBoard, envoyez la demande POST à l'URL suivante :

3.1.6. ThingsBoard pour la gestion des laboratoires

Après avoir vu le concept fondamental principal sous ThingsBoard. Nous allons maintenant discuter comment nous avons l'intégré dans notre projet pour la partie "Gestion des labos".

Notre intégration de ThingsBoard dans la solution proposée ne dépend sur aucune modification coté ThingsBoard mais seulement sur des règles (ou bien supposition) qui doivent être respecté. Nous allons les cités par la suite.

La première partie concerne les laboratoires ; il faut noter que ces règles contrôlent non seulement la définition des laboratoires sous ThingsBoard mais, mais aussi une partie de la conception des laboratoires lui-même.

- ✓ **R1** : Sous ThingsBoard, chaque laboratoire est représenté comme un périphérique (Device) ThingsBoard.
- ✓ **R2** : Tout périphérique représentant un laboratoire doit être du type[44] "**LAB**".
- ✓ **R3** : Chaque laboratoire est identifié par son **ID**[45] ThingsBoard
- ✓ **R4** : Chaque laboratoire a un type de laboratoire qui est défini par un attribut côté client nommé **"labTypeId**" qui doit être publie par le labo lui-même.
- ✓ **R5** : Des instances du même type de laboratoire ont le même **"labTypeId**".
- ✓ **R6** : Chaque laboratoire doit publier son adresse (web) distant comme un attribut côté client nommé **"URL**".
- ✓ **R7** : Chaque laboratoire doit publier un jeton (JWT[46]) de demande des session labo comme un attribut côté client nommé **"sessionCreationToken**"[47].
- ✓ **R8** : Chaque laboratoire doit maintenir son état d'activité (active ou non) sous ThingsBoard.
- ✓ **R9** : A fin atteindre l'objectif du **R6**, un laboratoire doit envoyer des mis-a-jour d'attribut côté client nommé "**status**"[48] périodiquement chaque **1min**.

La deuxième partie concerne la représentation des unités administratifs tel différents fournisseurs laboratoires par exemple, des agents administratifs :

- ✓ **R10 :** Un fournisseur (université par exemple) est représenté comme un Client (Customer) ThingsBoard.
- ✓ **R11 :** Les unités administratifs des fournisseur (par exemple des départements d'une université) sont représenté comme des assets ThingsBoard.
- ✓ **R12 :** Un agent administratif est représenté comme un utilisateur (User) ThingsBoard.

[44] Sous ThingsBoard chaque périphérique a un type, défini lors de sa création, est là nous proposent créations et réservations d'un nouveau type sous le nom "LAB" pour l'utilisation _est l'identifications_ des périphériques laboratoires (utilisée plus-tard par RESTY).

[45] Génère automatiquement par ThingsBoard lors de la création des périphériques.

[46] JSON Web Token (JWT) est un standard ouvert défini dans la RFC 7519. Il permet l'échange sécurisé de jetons entre plusieurs parties. Utilisés souvent comme une méthode d'authentification dans les applications web moderne. Plus d'infos sur cette notion sure : https://jwt.io/introduction.

[47] L'utilisation de ce jeton est discutée dans conception des laboratoires dans : 3.4 Développement laboratoires.

[48] Nous utilisons ces attributs avec la Chaîne des règles ThingsBoard (ThingsBoard Rule Engine) afin de contrôler l'état du laboratoires (active ou non) dans notre système.

Les relations entre ces unités sont représentées sous ThingsBoard en utilisons les relations Thingsboard.

Il est à noter que la plus parts des règles citées précédemment sont des propositions plus que des règles.

Il est à noter aussi que là nous avons cité seulement les règles qui qui gèrent les manipulations sous Thingsboard el les interactions entre les laboratoires et le serveur ThingsBoard. Autres règles qui gèrent autres aspects seront cité plus-tard.

Par la suite nous présentons les configurations/exemples manipulations sous l'enivrement ThingsBoard suivant les règles citées précédemment.

1. Configuration du moteur des règles (Rule engine) [49] : la première chose que nous avons faite c'était la configuration du moteur des règles en :

[49] Pour plus d'infos sur le moteur des règles sous ThingsBoard consulté la documentation officielle sur : https://thingsboard.io/docs/user-guide/rule-engine-2-0/re-getting-started/

1.1. Modifiant la chaîne principale (root) en ajoutant une règle qui contrôle les mis-a-jours d'états d'activités des laboratoires **(R9)**:

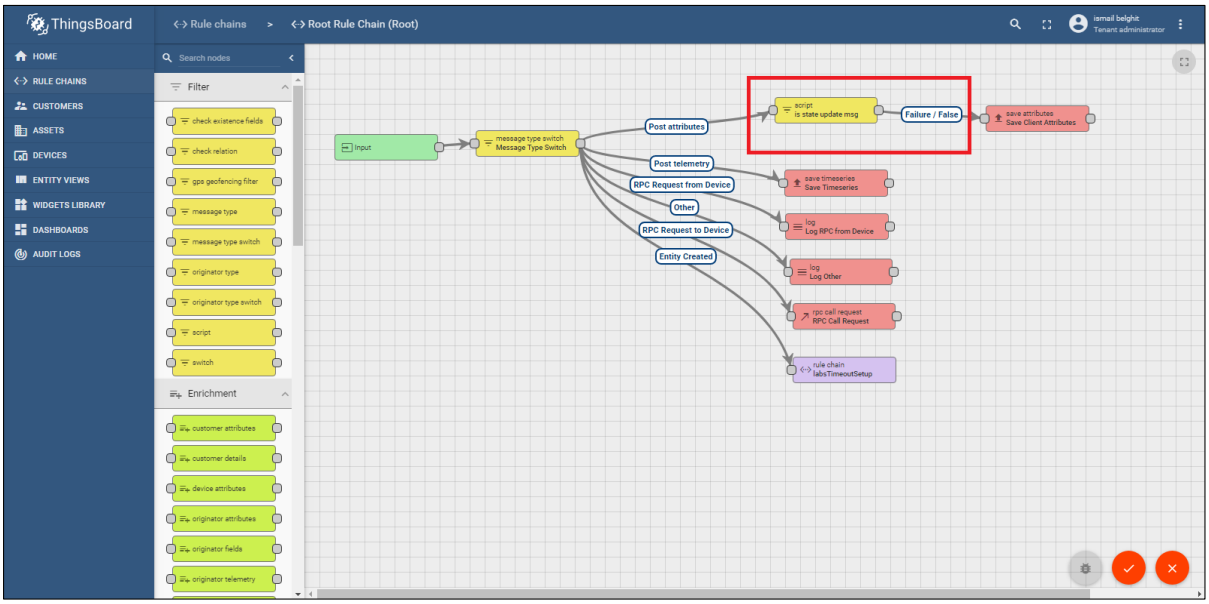


Figure 18: Modification de la chaîne principale (root).

1.2. Création d'une nouvelle chaîne pour la gestion des états d'activités des laboratoires (**R9**) :

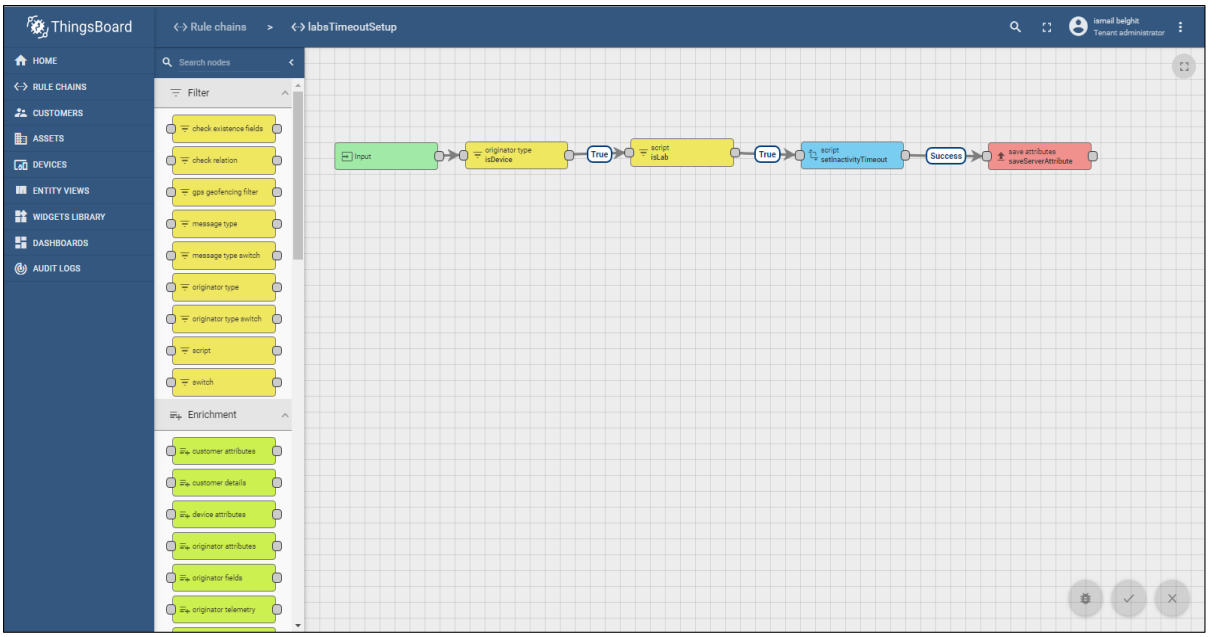


Figure 19: La nouvelle chaîne pour la gestion des états d'activités des labos.

2. Créations des nouveaux fournisseurs :

2.1. Senario : nous allons réaliser le scénario d'exemple suivant (figure 18) :

- ❖ Deux universités (fournisseurs) : univ A et univ B.
- ❖ Univ A avec deux départements : dép 1 et dép2.

- Univ B avec un département : dép 3.
- Dép 1 avec deux labos : lab 1 et lab 2.
- Dép 2 avec un labo : lab 3.
- Dép 3 avec deux labos : lab 5 et lab 5.

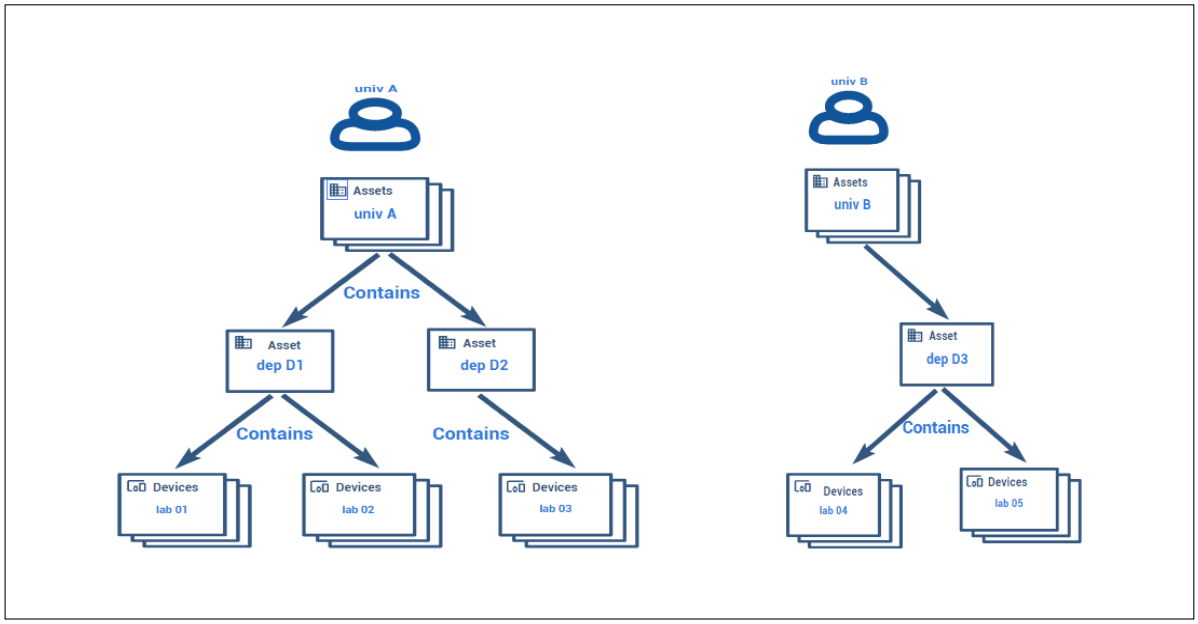


Figure 20: Scénario d'une configurations ThingsBoard 1.

Note du figure 20 que nous proposons la création d'un asset _ global si c'était juste à dire_ qui group tous les autres assets du même fournisseur afin de faciliter la gestion.

2.1 Manipulation :

- Création des fournisseurs (Costumers) : pour cela suivre les étapes suivantes :

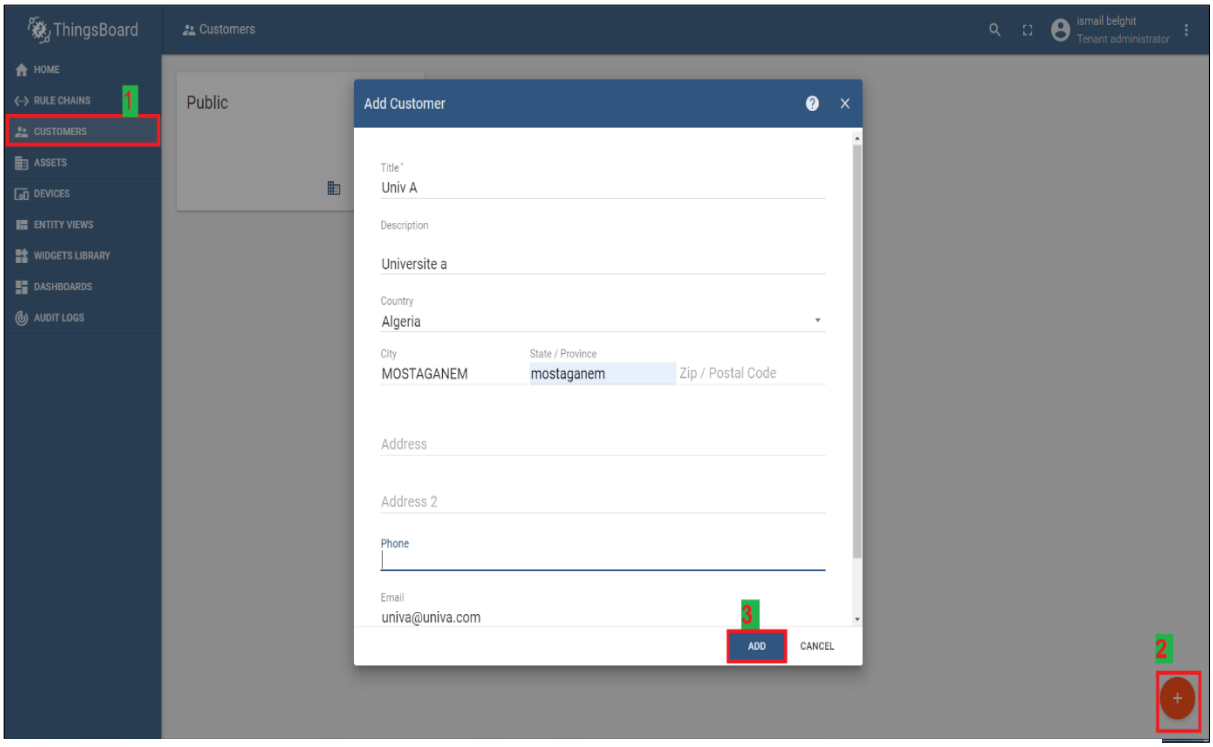


Figure 21: Création des fournisseurs (Costumers) 1.

- Création des départements (assets) :

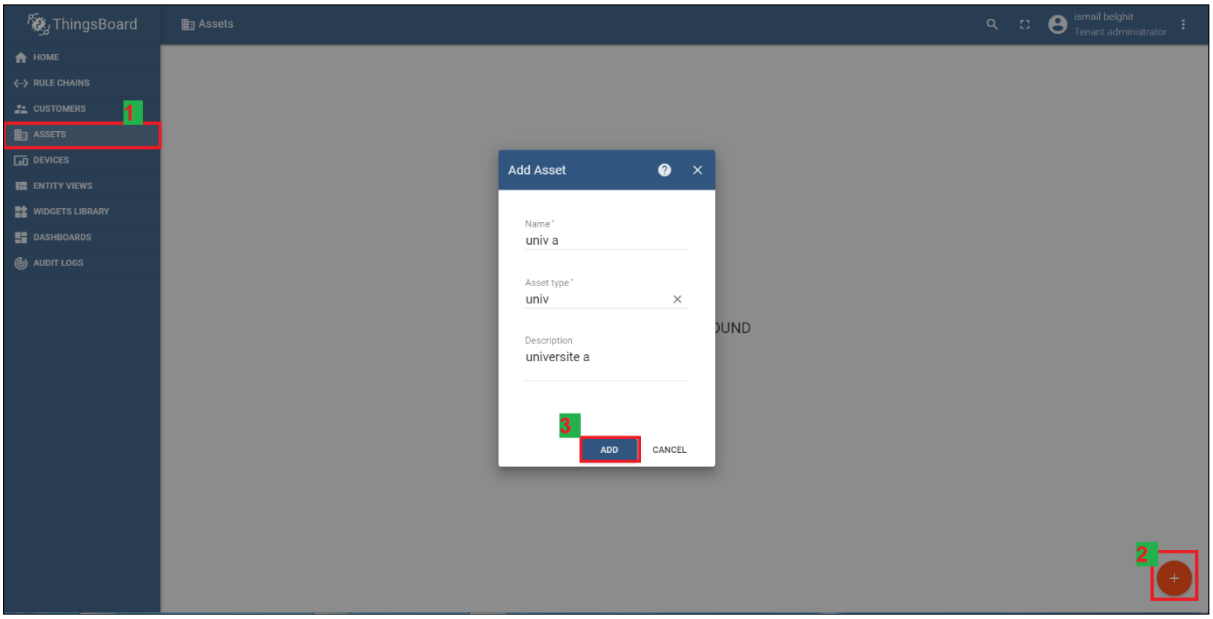


Figure 22: Création des départements (assets) 1.

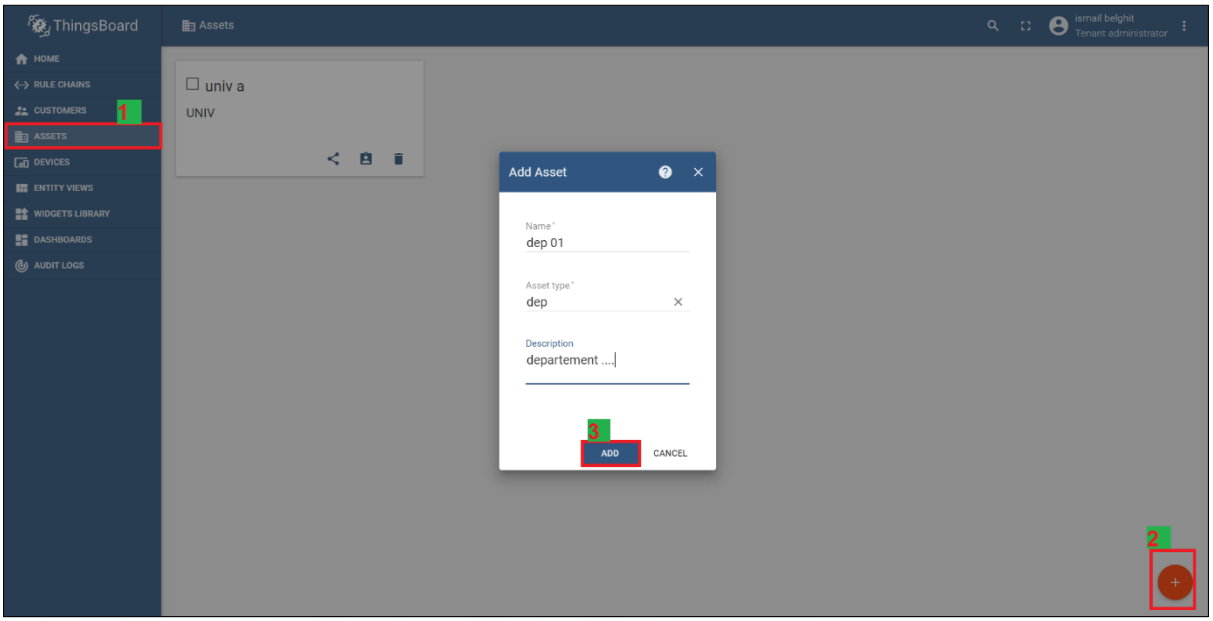


Figure 23: Création des départements (assets) 2.

Maintenant on attribuer les universités aux clients (Customers)

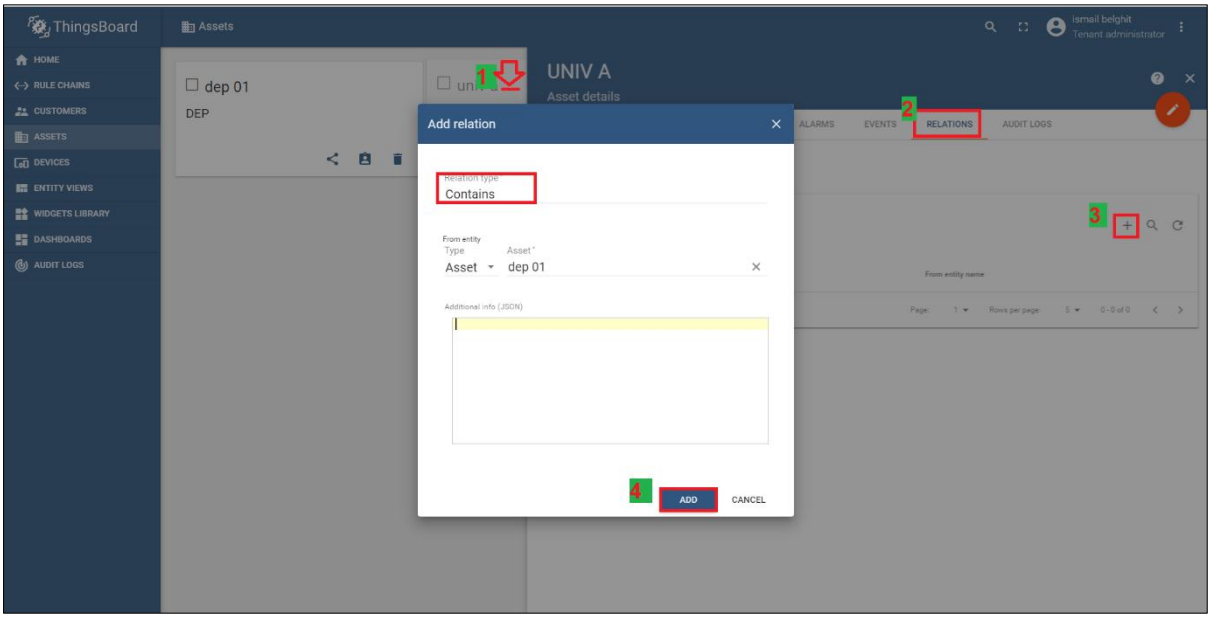


Figure 24: Attribution les universités aux clients.

Noter que par défaut ThingsBoard ne prend pas en charge l'attribution sous entités d'une manière automatique, donc l'attribution des départements et les labos au client doit être réalisé manuellement, pour cela il suffit juste de répéter l'étape précèdent avec les départements et les labos.

- Création des labos (devices) :

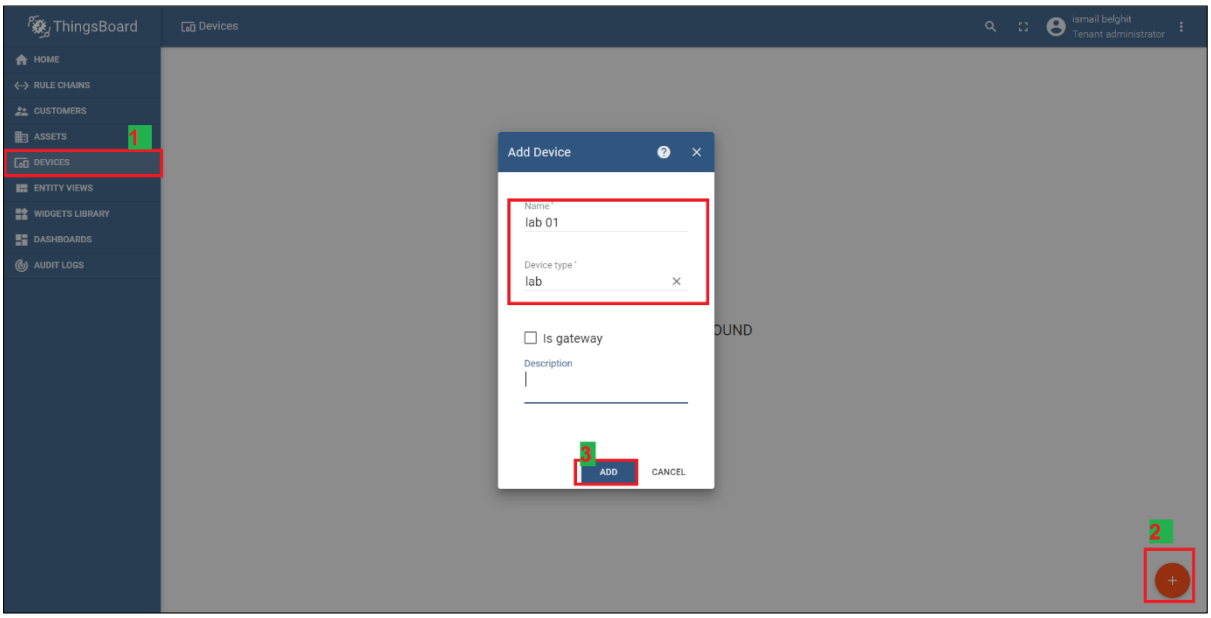


Figure 25: Création des labos (devices) 1.

De la même façon :

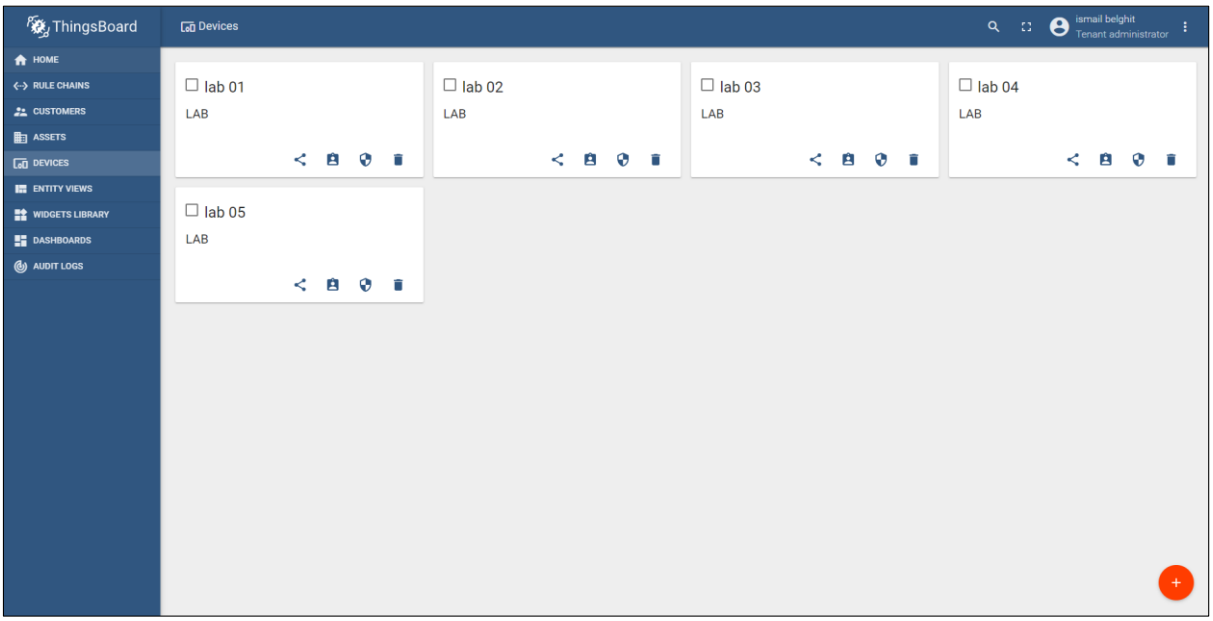


Figure 26 : Création des labos (devices) 2.

Relation labo-département :

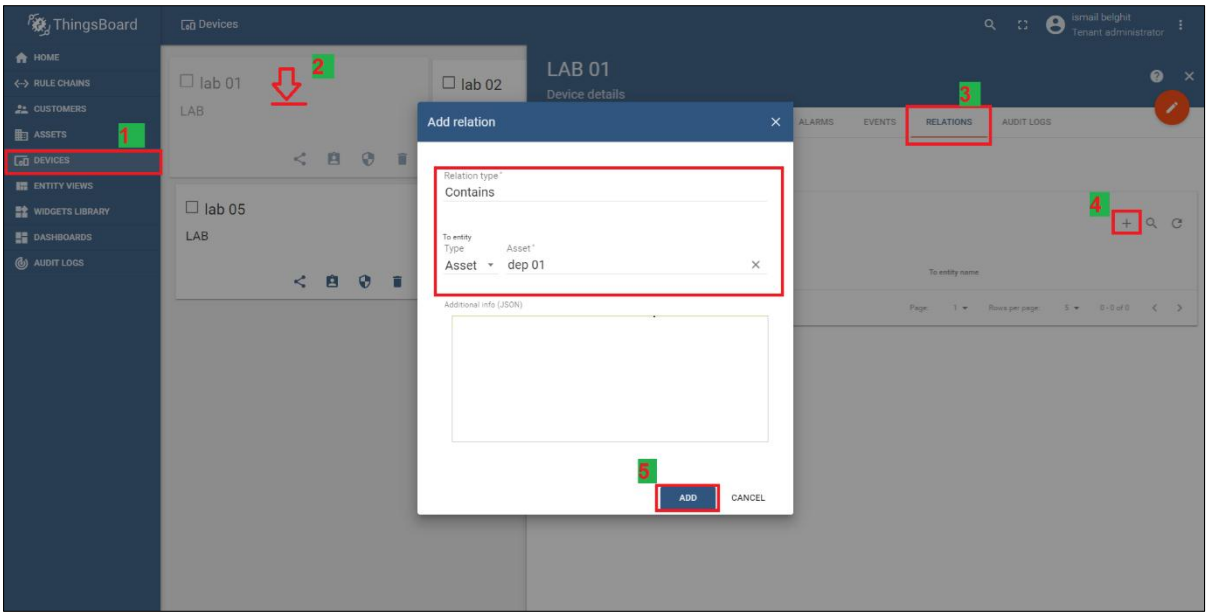


Figure 27 : Relation labo-département

3.2 Gestions des utilisateurs des laboratoires (Portail Web : LabsUi)

Comme déjà mentionné, partie gestion des utilisateurs des laboratoires ; Nous avons développé notre propre plate-forme (web) qu'il s'agit d'un portail web avec un système de gestion d'utilisateurs basique (inscription, connexion, ...) qui est capable _ Par l'intermédiaire du **Resty**[50] _ d'exposés les laboratoires intégrés (gérés) dans notre système (intégrés dans ThingsBoard) aux ces utilisateurs.

Par la suite nous allons essayer de donner une vue globale sur cette plate-forme, comment nous l'avons développé, les technologies utilisées, et une présentation du résultat final.

3.2.1 LabsUi

LabsUi est un portail web avec un système de gestion d'utilisateurs basique qui offre à nos utilisateurs final la possibilité d'inscription et la création des nouveaux profiles afin d'explorer nos laboratoires ainsi que la possibilité de réserver des sessions sur ces labos.

3.2.2 Technologies utilisées

LabsUi est une application web développé principalement avec des technologies web modernes ; comme tout application web notre application peut être décomposée en deux parties complémentaires ; Partie frontal (font-end) représenté par l'interface graphique de la plateforme, et partie coté serveur (back-end) représente la logique applicative de l'application qui est dans notre cas API REST.

Partie frontal (font-end)

Cette partie représente l'interface graphique de l'application développé principalement en **HTML**[51], **CSS**[52], **JavaScript**[53] , en utilisant les bibliothèques/frameworks (kit logiciel) suivantes :

- ***VueJs***[54] : tel que défini par ses créateurs, Vue est un framework évolutif pour la construction d'interfaces utilisateur.

[50] Voir page 25 :"3. Présentation de la solution proposée".

[51] HyperText Markup Language, est le langage de balisage conçu pour représenter les pages web ; plus d'infos sur : https://fr.wikipedia.org/wiki/Hypertext_Markup_Language.

[52] Cascading Style Sheets, un langage informatique qui décrit la présentation des documents HTML ; plus d'infos sur : https://fr.wikipedia.org/wiki/Feuilles_de_style_en_cascade.

[53] JavaScript, est un langage de programmation de scripts principalement employé dans les pages web interactives mais aussi pour les serveurs ; plus d'infos sur : https://fr.wikipedia.org/wiki/JavaScript.

[54] Site officiel : https://vuejs.org.

- ***Bootstrap***[55] : un kit CSS créé par les développeurs de *Twitter*, est devenu en peu de temps le framework CSS de référence dans la communauté de développement web.
- ***jQuery***[56] : Est une bibliothèque JavaScript libre et multiplateforme créée pour faciliter l'écriture de scripts côté client dans le code HTML des pages web.
- ***Axios*** : Est une bibliothèque client HTTP pour JavaScript qui peut être utilisée dans les applications web frontal (front-end) ou coté serveur.
- ***Lodash***[57] : Est une bibliothèque JavaScript réunissant des fonctions bien pratiques pour manipuler des données, là où peuvent manquer des instructions natives pour: des tableaux, objets, chaînes de texte, etc.

Coté serveur (back-end)

Représente la logique applicative de l'application qui est expose à l'autre partie comme un API REST, développe sous **NodeJs** en utilisant les bibliothèques/frameworks suivantes :

- ***NodeJs***[58] : Est une plateforme logicielle libre open source en JavaScript orientée vers les applications coté serveur.
- ***SailsJs***[59] : Est un framework pour la création des applications web sous NodeJs.
- ***Restify-clients***[60] : Est une bibliothèque client HTTP pour JavaScript qui peut être utilisée dans les applications web frontal (front-end) ou coté serveur.
- ***Jsonwebtoken***: Implémentation des standards JWT en JavaScript.
- ***Axios***.
- ***Lodash***.

3.2.3 Présentation LabsUi

Par suite nous allons présenter quelque captures écran réel l'application final (LabsUi).

[55] Site officiel : https://getbootstrap.com.
[56] Site officiel : https://jquery.com.
[57] Site officiel : https://lodash.com.
[58] Site officiel : https://nodejs.org.
[59] Site officiel : https://sailsjs.com.
[60] Site officiel : restify.com.

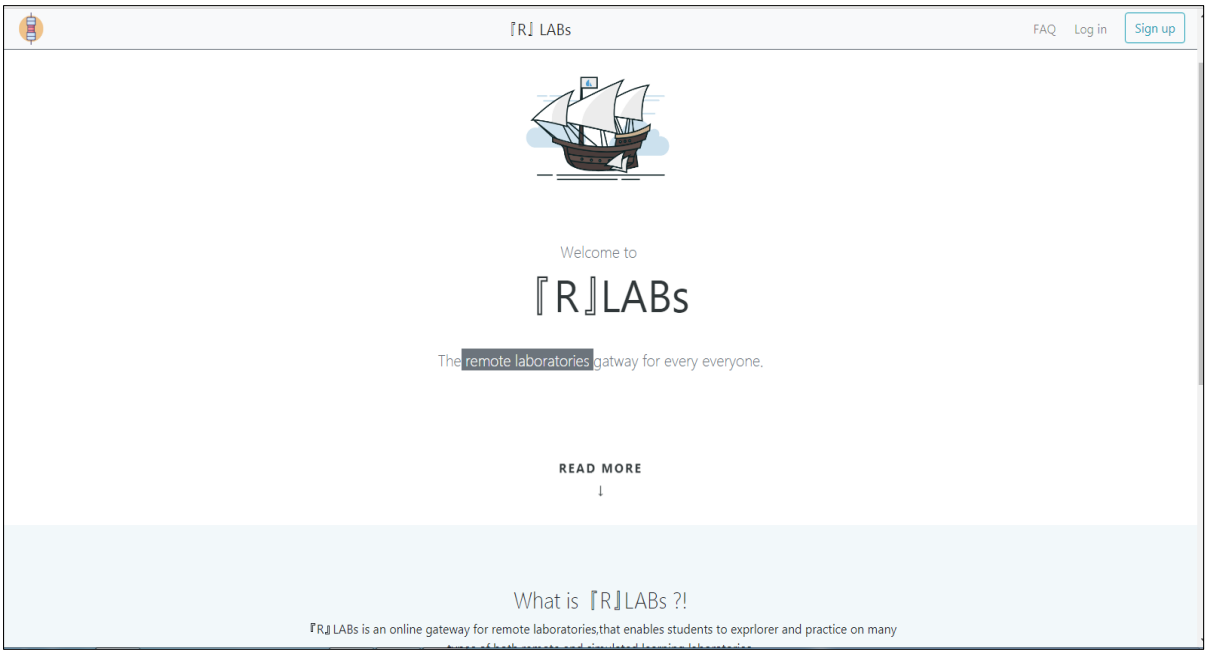


Figure 29 : LabsUi, page principale.

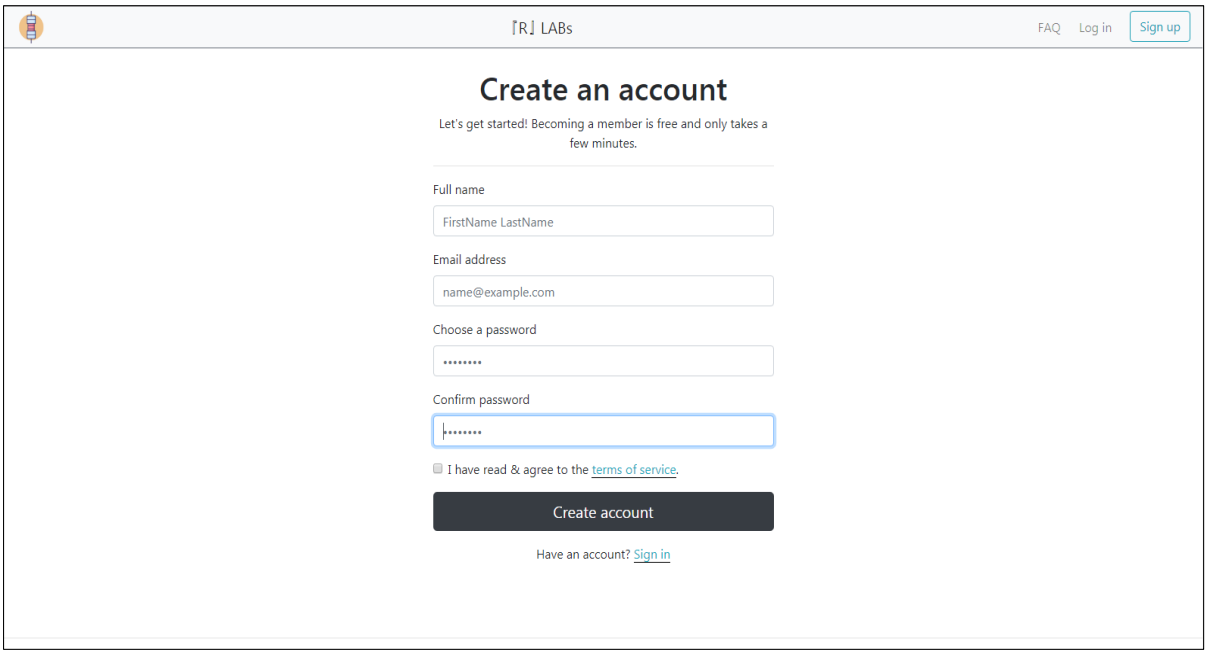


Figure 28 : LabsUi, page d'inscription.

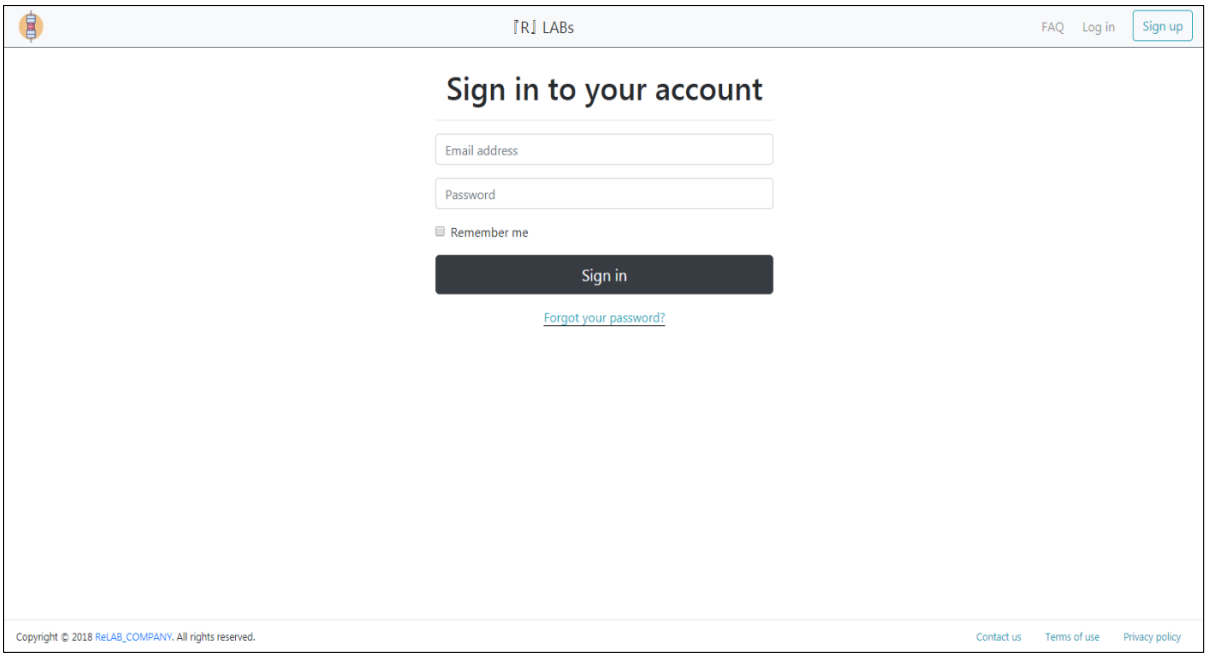


Figure 30: LabsUi, page de conexion.

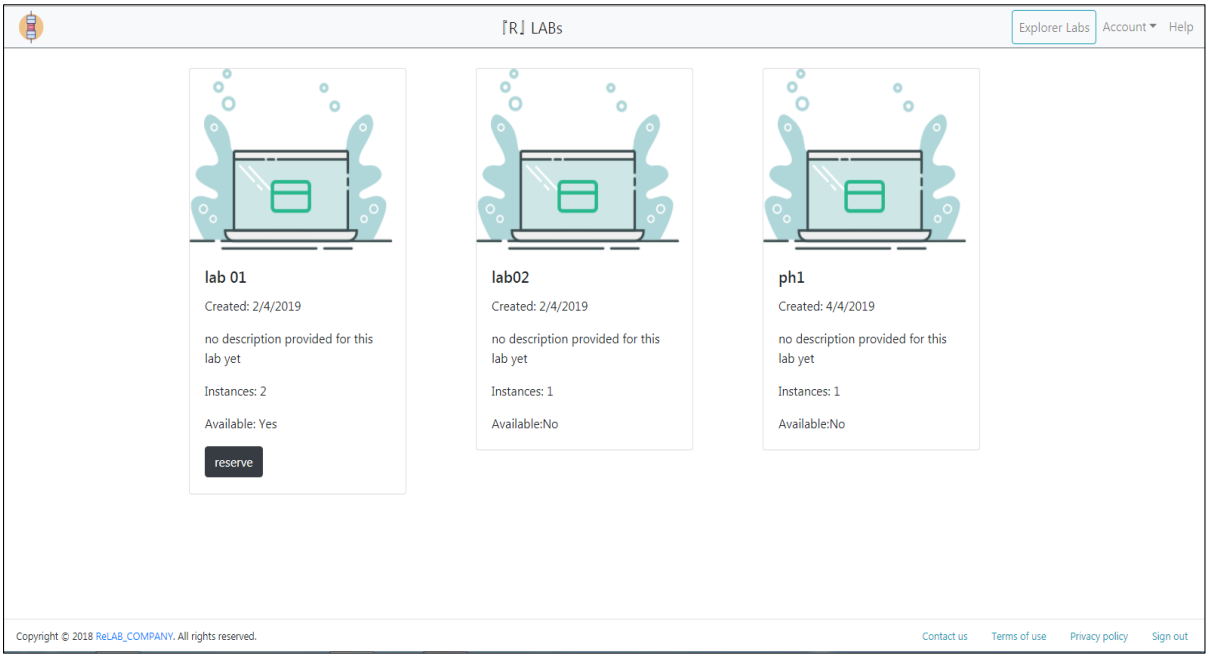


Figure 31: LabsUi, page des laboratoires

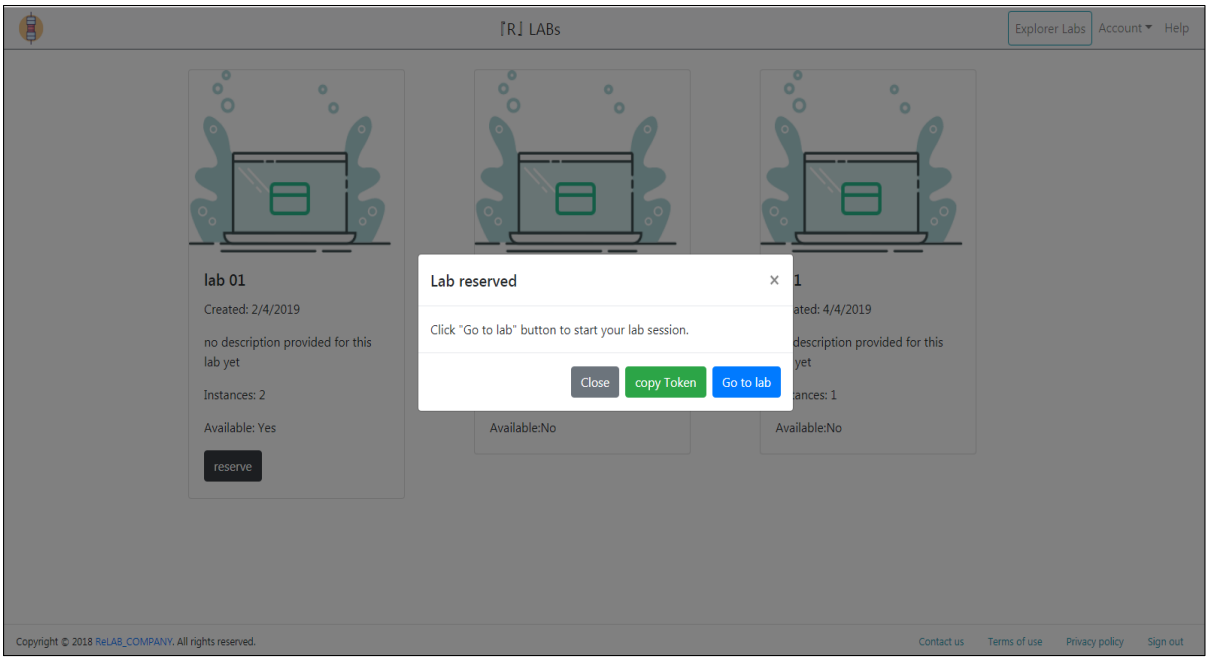


Figure 32: LabsUi, reservation d'un laboratoire.

2.3. API intermédiaire (Resty)

Comme nous avons déjà cité, la coordination entre notre portail web (LabsUi) et tout autre composant dans notre système (ThingsBoard el les laboratoires) est géré par un API web intermédiaire (Resty).

Par la suite nous allons essayer de donner une vue globale sur cette API.

2.3.1. Resty

Resty est un API REST web développé en **JavaScript** sous **NodeJs**. Le principale rôle cette API web est de servir comme intermédiaire entre les autres composantes de notre système. Resty gère la relation (interactions) entre les deux parties principales dans notre système : gestion des laboratoires ou **ThingsBoard** et le portail web (**LabsUi)**, Ainsi que le portail web et les laboratoires dans certains cas.

L'idée avec **Resty** c'était juste de créer une API middleware (intermédiaire) qui simplifiait les interactions avec les fonctionnalités proposées déjà par Thingsboard et les laboratoires. Ainsi, Resty n'offre pas de véritables fonctionnalités réellement, mais c'est juste une couche d'abstraction qui simplifier les fonctionnalités déjà disponibles via ThingsBoard et les laboratoires. Cette couche d'abstraction dépend donc l'API d'administration ThingsBoard et des API offertes par les API des laboratoires[61].

[61] Discuté en plus de détails en : 3.4 Développement laboratoires.

2.3.2. Utilités

Les principaux tâches (utilités) de Resty prouvent êtres résumées sous les deux points suivants :

1. *Exposer les laboratoires inscrit sous ThingsBoard au portails web* : Notre portail dépend sur **Resty** pour récupérer les informations nécessaires sur les laboratoires inscrits ThingsBoard. Dans ce cas, Resty lui-même dépend sur l'API d'administration ThingsBoard pour afin de récupérer ces informations.
2. *Réservations des sessions laboratoires :* Dans le cas d'une réservation des le portail web aussi dépend sur **Resty**. Dans ce cas, Resty utilise les API des laboratoires pour compléter la réservation.

2.3.3. Technologies utilisées

Resty est une application web développé principalement avec des technologies web modernes sous **NodeJs** ; en utilisant les bibliothèques/frameworks (kit logiciel) suivantes :

- ***NodeJs***[62] : Est une plateforme logicielle libre open source en JavaScript orientée vers les applications coté serveur.
- ***Restify***[63] : tel que défini par ses créateurs, Restify est une framework de développement services Web Node.js optimisée pour la construction de services Web REST sémantiquement corrects, prêts à être utilisés en production à grande échelle. Restify optimise l'introspection et les performances et est utilisé dans certains des plus grands déploiements de Node.js sur Terre.
- ***Restify-clients***[64] : Est une bibliothèque client HTTP pour JavaScript qui peut être utilisée dans les applications web frontal (front-end) ou coté serveur.
- ***Axios***[65] : Est une bibliothèque client HTTP pour JavaScript qui peut être utilisée dans les applications web frontal (front-end) ou coté serveur.
- ***Lodash***[66] : Est une bibliothèque JavaScript réunissant des fonctions bien pratiques pour manipuler des données, là où peuvent manquer des instructions natives pour : des tableaux, objets, chaînes de texte, etc.

[62] Site officiel : https://nodejs.org.

[63] Site officiel : http://restify.com.

[64] Site officiel : restify.com.

[65] Site officiel : https://github.com/axios/axios.

[66] Site officiel : https://lodash.com.

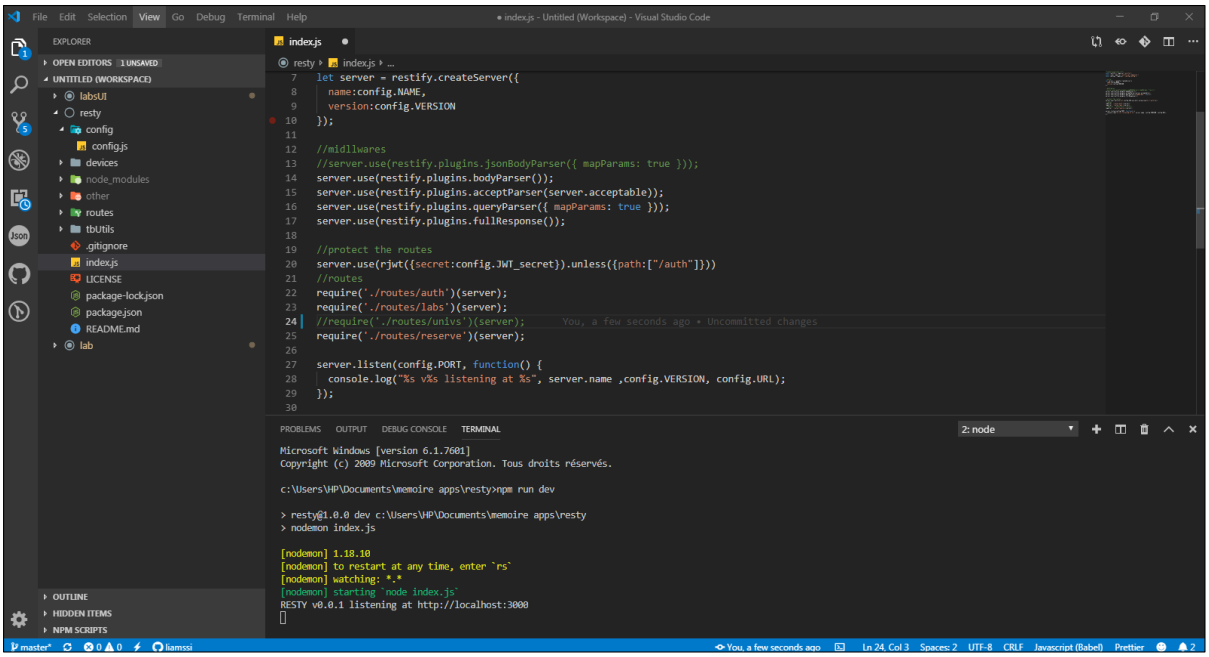


Figure 33: Resty , environnement de développement.

3. Développement laboratoires

Comme nous l'avons déjà cité, Notre solution ne vise que l'aspect "gestions et publications des laboratoires distants" du problématique "**Laboratoires distants**". Cependant qu'ils y autres aspects qui doivent être géré par les laboratoires.

Notre solution proposée ne pose aucunes restrictions sur la méthodologie de développement des laboratoires ou les technologies utilisées ; Cependant, qu'ils Il existe certaines attentes qu'ils doivent être satisfaites par ces laboratoires ; Pour cela nous avons défini quelques règles (ou besoins à satisfaire) à suivre lors de développement de ces labos afin de garantir une intégration correcte dans notre système. Certains de ces règles contrôlent (gèrent) la relation entre les labos et le serveur **ThingsBoard** (nous avons déjà les citer dans "3.1.5 ThingsBoard pour la gestion des laboratoires "[67]), les autres gèrent la relation entre les labos et les serveur **Resty** ; nous allons les cités par la suit à :

- ✓ **R1** : Chaque laboratoire doit offrir la possibilité générations des jetons sessions (session token) et la création des sessions a base de ces jetons.[68]
- ✓ **R2** : Chaque laboratoire doit offrir les point API suivants :

[67] Voir du **R4** au **R9** du : 3.1.5ThingsBoard pour la gestion des laboratoires.

[68] Resty dépend sur cette contraint à l'aide de **R2**, pour la réservation des sessions.

Chemin	Methods	Headers	Paramètres	Commentaire
/sessiontoken	*Post*	*Authaurisation : $sessinCreationToken*[69]		Le retour de cette point API et un jeton de session (session Token). Utilisé par Resty pour réserver des jetons de sessions pour les clients.
/session	GET		Token=$sessionToken	Utilisé pour la création des sessions sur le lab. $sessionToken est un jeton de session généré par le labo.

Suivant les contraints de conceptions des laboratoires cité précédemment (R4 au R11) nous avons développé un petit application web qui va servir comme une preuve de concept pour notre solution. Dans cette application nous basons seulement sur les aspects que nous intéresses qui la partie gestion de labo.

Comme nous l'avons cité notre application labo n'est pas un laboratoire réel complète mais juste la partie gestion labo ; les fonctionnalités inclus sont seulement ceux qui sont nécessaire pour **l'intégration** avec notre solution proposée.

Le but d'avec cette application est de servir comme une preuve de concept pour notre solution et aussi de fournit colonne vertébrale (backbone) pour _ faciliter _ le développement des nouveaux laboratoires intégrables avec notre solution pour ceux développeurs laboratoires qui veulent se concentrer sur le développement des laboratoires elles-mêmes mais non pas sur les fonctionnalités secondaires.

3.1. Technologies utilisées

Cette application est une application web développé principalement avec des technologies web modernes sous **NodeJs** avec une simple interface en **HTML**, **CSS** et **JavaScript** ; en utilisant les bibliothèques/frameworks (kit logiciel) suivantes :

- ✓ ***NodeJs*** : Est une plateforme logicielle libre open source en JavaScript orientée vers les applications coté serveur.
- ✓ ***ExpressJs*** : Tel que défini par ses créateurs, ExpressJs est une infrastructure web minimaliste, souple et rapide pour Node.js.

[69] Voir **R7** de : 3.1.5ThingsBoard pour la gestion des laboratoires.

- ***Bootstrap*** : un kit CSS créé par les développeurs de *Twitter*, est devenu en peu de temps le framework CSS de référence dans la communauté de développement web.
- ***jQuery*** : Est une bibliothèque JavaScript libre et multiplateforme créée pour faciliter l'écriture de scripts côté client dans le code HTML des pages web.

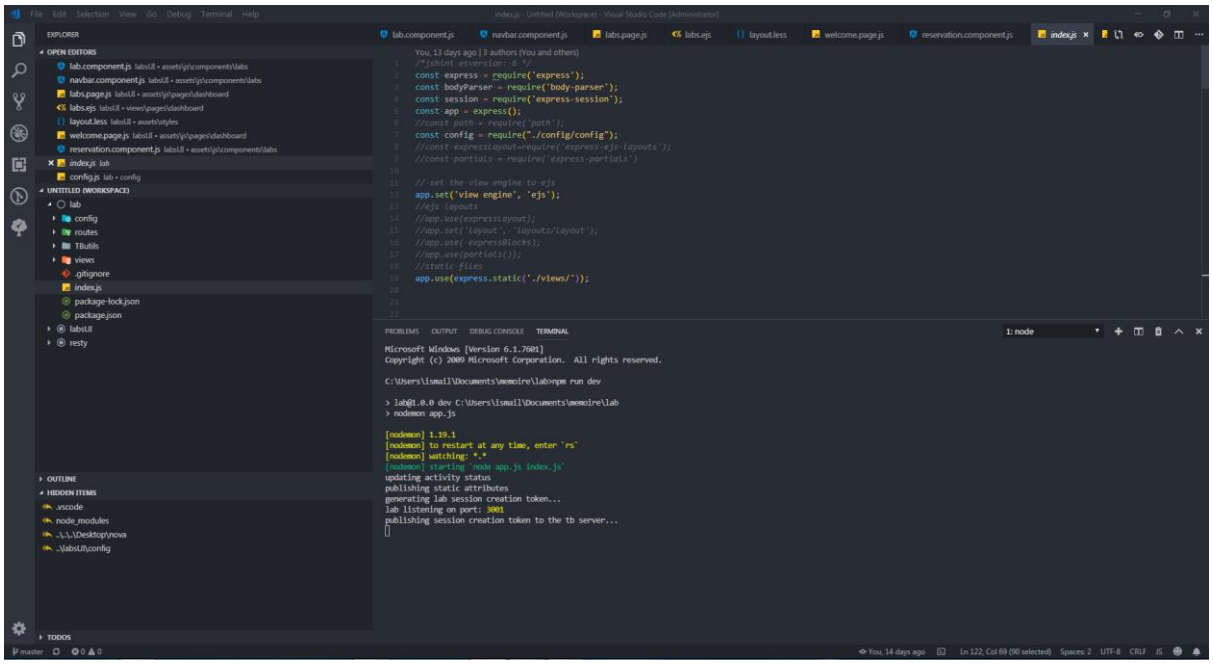


Figure 34: Laboratoire exemple , environnement de développement.

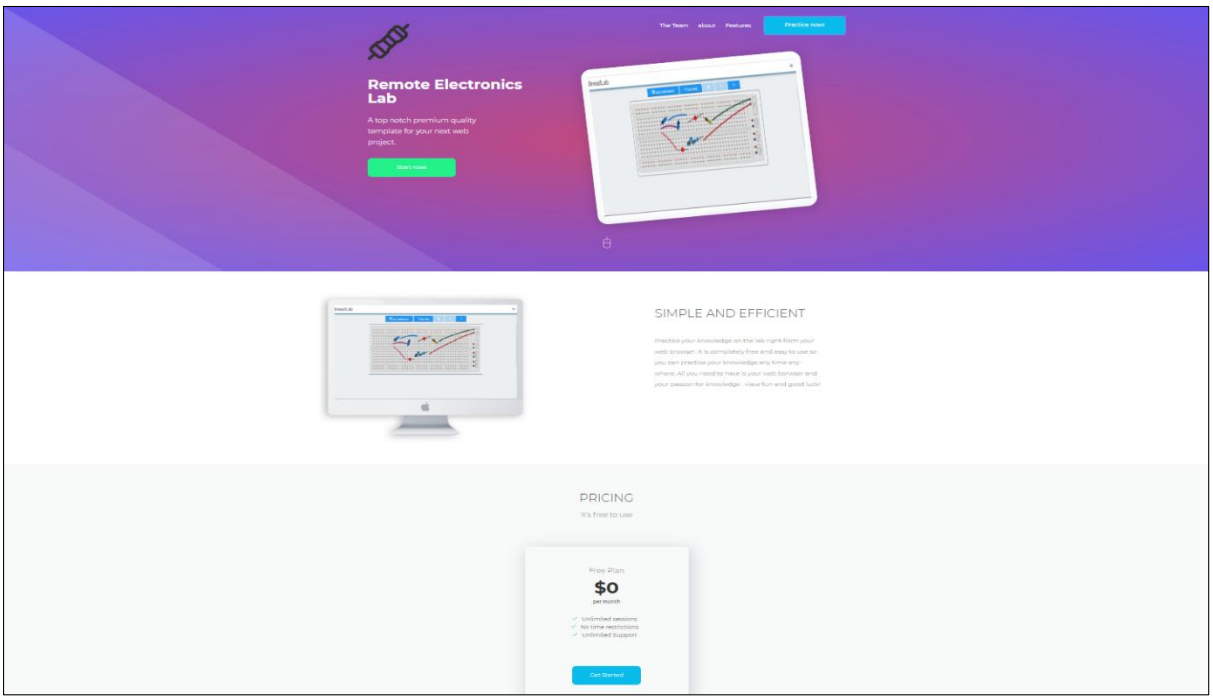


Figure 35: Laboratoire exemple ,page principale.

4. Interactions entre les composants système

Après avoir présenté tous les composants de notre système, nous allons maintenant discuter les interactions possibles entre ces composants.

Les deux Scénarios possible d'interaction entres les composants de notre système sont :

1. Lors de l'demande de liste des laboratoires au niveaux du portail (pour l'affichage)

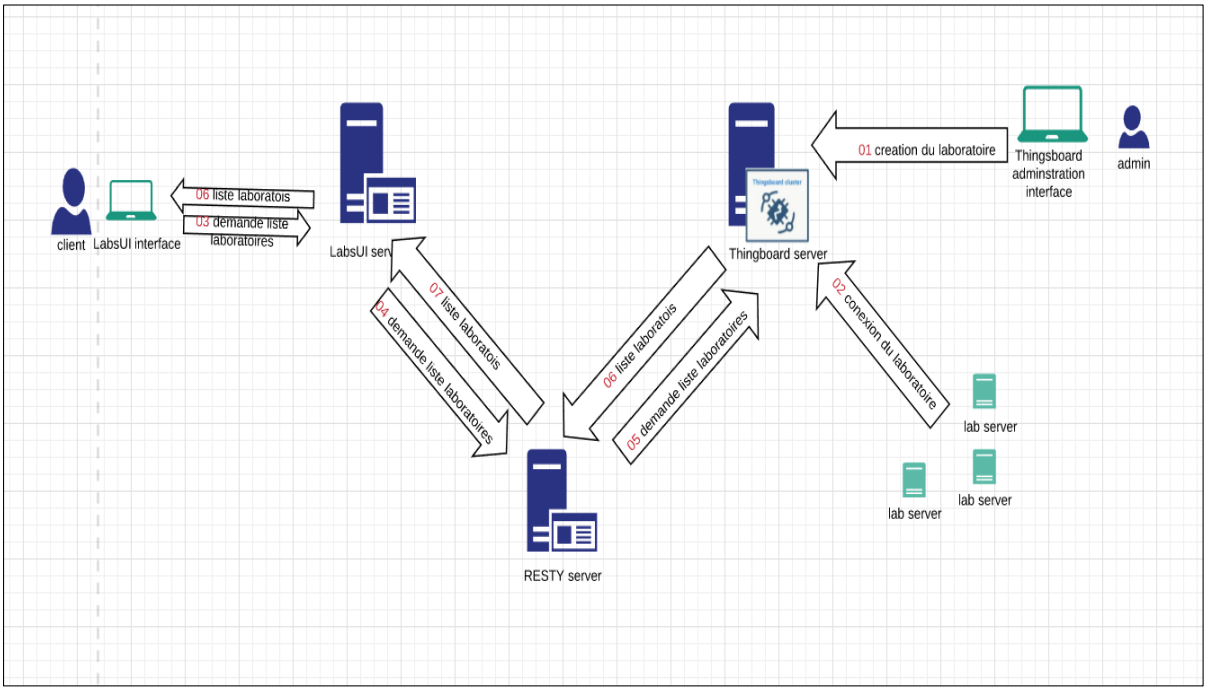


Figure 36: Les interactions entre les composants système lors d'affichage au niveaux portail web.

Noter que les étapes 05 et 06 peuvent être exécuté avant l'étape 04 afin d'optimisé le temp de réponse.

2. Lors d'une demande de réservation d'un laboratoire :

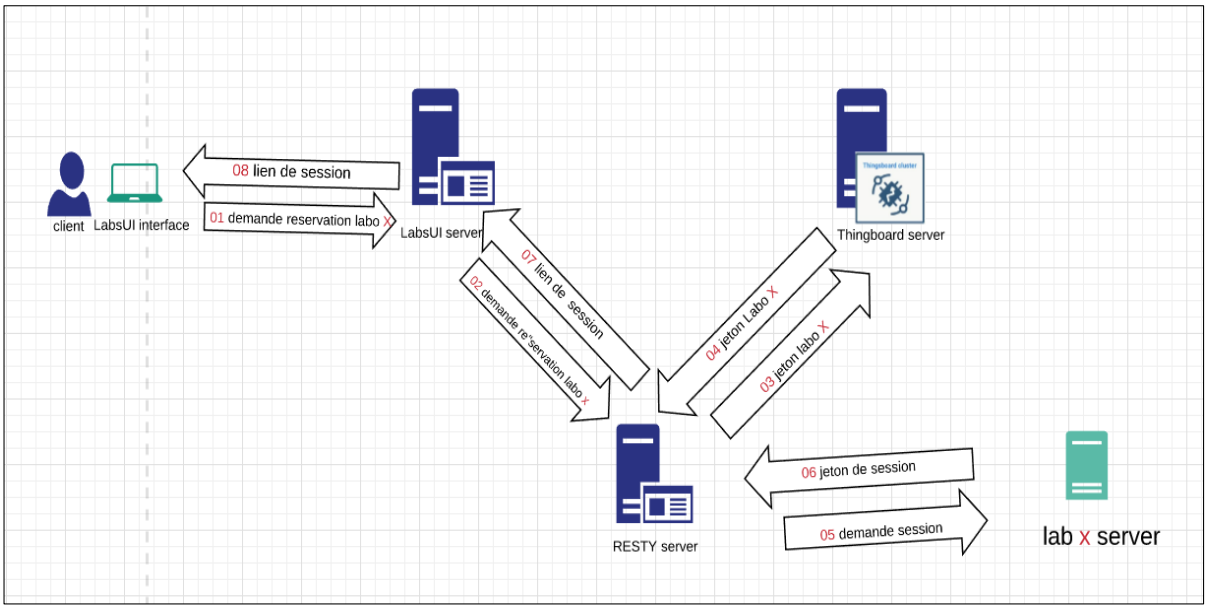


Figure 37 : processuse d'une reservation laboratoire.

Finalement le lien obtenu d'étape 08 est utilisé pour la création une nouvelle session au niveau de labo X.

5. Conclusion

Dans ce chapitre nous avons présenté une solution complète pour la gestion des laboratoires en ligne. La solution que nous avons proposée et se base sur l'intégration des solution logiciel open source déjà existants (ThingsBoard) avec autres applicatifs web que nous avons développés.

2. CONCLUSION GENERALE

Le travail réalisé est pour nous une très bonne expérience personnelle et pédagogique afin de suivre le rythme de la recherche scientifique dans le domaine d'informatique et notamment le développement des laboratoires à distance. En effet, nous avons pu réaliser un projet d'actualité qui représente une solution pour un des problèmes rencontrés dans ce domaine :la gestions et publications des laboratoires distants.

L'avantage de notre projet est qu'il est basé sur des outils open source et efficace avec un coût de développement très faible par rapport aux autres travaux tout en offrant une implémentation très facile avec une architecture innovante.

En guise de perspectives, un complément de ce travail peut être réalisé en développant d'autres fonctionnalités complémentaires tel que l'amélioration d'expérience utilisateur ou la collecte des données d'usages des laboratoires et du plateforme afin de les améliorer.

REFERRENCES

1. Alves, Gustavo Ribeiro et al. “Spreading remote lab usage a system — A community — A Federation.” 2016 2nd International Conference of the Portuguese Society for Engineering Education (CISPEE) (2016): 1-7.

2. Gomes, Luís and Javier Garcia-Zubia. “Advances on Remote Laboratories and e-Learning Experiences.” (2007).

3. Molly, H., Shory, Anna Lawrence and CrowlDepartment. “Running Control Engineering Experiments Over the InternetCarisa Bohus.” (1995).

4. Distance learning applied to control engineering laboratories. Education, IEEE Transactions on, 39(3):320– 326.Aktan, B., Bohus, C., Crowl, L., and Shor, M. (1996).

5. J. Henry " Running laboratory experiments via the world wide web" In ASEE Annual Conference. (1996).

6. Orduña, Pablo, Luis Rodriguez-Gil, Javier Garcia-Zubia, Olga Dziabenko, Ignacio Angulo, U. Miguel Hernández and Esteban Azcuenaga. “Classifying online laboratories: Reality, simulation, user perception and potential overlaps.” . 13th International Conference on Remote Engineering and Virtual Instrumentation (REV) (2016): 224-230.

7. .M. H. Shor, C. Bohus, and B. Atkan.“Second best to being there: An historical perspective,” Using Remote Labs in Education, p. 27, 2011.

8. Ma, Jing and Jeffrey V. Nickerson. “Hands-on, simulated, and remote laboratories: A comparative literature review.” ACM Comput. Surv. 38 (2006): 7.

9. Pablo Orduna, "Transitive and scalable federation model for remote laboratories",Thèse de doctorat, Université de Bilbao, Spain, 2013.

10. Jong, Ton A W de, Marcia C. Linn and Zacharias C. Zacharia. “Physical and virtual laboratories in science and engineering education.” Science 340 6130 (2013): 305-8 .

11. J. Froyd, P. Wankat, and K. Smith, “Five major shifts in 100 years of engineering education,” Proceedings of the IEEE, vol. 100, no. 13, pp. 1344–1360, 2012.

12. G. Canfora, P. Daponte, and S. Rapuano,“Remotely Accessible Laboratory for Electronic Measurement Teaching”,Computer Standards and Interfaces, vol. 26, no. 6, pp. 489-499, 2004.

13. W.J. Hutzel, Int J. Eng. Education“A Remotely Accessed HVAC Laboratory for Distance Education”, vol. 18, no. 6, pp. 711-716, 2002.

14. Cooper, Martyn, Alexis A. Donnelly and José M. M. Ferreira. “Remote controlled experiments for teaching over the internet: a comparision of approaches developed in the pearl project.” ASCILITE (2002).

15. Sonnenwald, Diane H., Mary C. Whitton and Kelly Maglaughlin. “Evaluating a scientific collaboratory: Results of a controlled experiment.” ACM Trans. Comput.-Hum. Interact. 10 (2003): 150-176.

16. Orduña, Pablo, Aitor Almeida, Diego López-de-Ipiña and Javier García Zubía. “Learning Analytics on federated remote laboratories: Tips and techniques.” 2014 IEEE Global Engineering Education Conference (EDUCON) (2014): 299-305.

17. L’article web [En line]: https://www.nginx.com/blog/introduction-to-microservices/.

18. Pablo Orduna, Luis Rodriguez-Gil, Javier Garcia-Zubia, Ignacio Angulo,Unai Hernandez, and Esteban Azcuenaga. *"Increasing the value of Remote Laboratory federations through an open sharing platform: LabsLand"*,. DeustoTech - Deusto Institute of Technology, Faculty of Engineering, University of Deustoand and LabsLand, Bilbao, Spain.

19. audibert, Laurent. *developpez.com.* [En line]. https://laurentaudibert.developpez.com/Cours-UML (2013).

20. *wikipedia.* [En line] https://fr.wikipedia.org/wiki/UML_(informatique).