République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université Abdelhamid Ibn Badis – Mostaganem
Faculté des Sciences et de la Technologie

THÈSE

Pour obtenir le diplôme de
DOCTORAT ES SCIENCES
En Électronique

Présentée et Soutenue Par

MOUSSA Mohammed

Intitulée

Conception d'un environnement d'apprentissage collaboratif interopérable dédié aux travaux pratiques distants

Le 08/12/2021 10h00
Salle des soutenances FST

JURY

Nom & Prénom	Grade & Établissement	Qualité
Mr Bekkouche Benaissa	Professeur - Université de Mostaganem	Président
Mr Babouri Abdeslam	Professeur - Université de Guelma	Examinateur
Mr Fezari Mohamed	Professeur - Université d'Annaba	Examinateur
Mr BENACHENHOU Abdelhalim	Professeur - Université de Mostaganem	Directeur de Thèse

Année 2021

الملخص

تعد المختبرات عن بعد حلاً للمؤسسات التي تقدم برامج تعليمية عبر الإنترنت وترغب في تقديم تدريب عملي لطلابها. في معظم الأحيان ، لا تمتلك بعض المؤسسات مختبرات مجهزة بشكل صحيح. لهذه الأسباب ، بدأت بعض الجامعات في استخدام المختبرات عن بعد ، مما أدى إلى خفض التكاليف من حيث المساحة والمعدات والصيانة والموظفين ، من بين أمور أخرى. تعتبر المختبرات عبر الإنترنت بديلاً لتوفير الوصول عن بُعد عبر اتصال الإنترنت إلى معدات حقيقية أو تجارب افتراضية موجودة في مكان آخر.

مشاركة المختبرات بشكل فعال أمر مثير للاهتمام لعدة أسباب. يسمح للطلاب بالحصول على مزيد من الوصول إلى النظام الأساسي لمزيد من الخبرات. بالإضافة إلى ذلك ، هناك ميزة موجودة عادة في المعامل البعيدة والتقليدية وهي أنها غير مستخدمة لفترة طويلة. في حالة المعامل البعيدة ، يمكن مشاركتها بغض النظر عن المدينة أو البلد أو القارة التي يوجد بها الطالب. تتيح مشاركة المختبرات البعيدة تقليل تكاليف تنفيذها. على الرغم من أن المختبر البعيد قد يكلف أكثر
من مختبر تقليدي ، فإن المشاركة لديها القدرة على تقليل التكلفة.

الهدف من هذه الأطروحة هو اقتراح بنية موجهة نحو الخدمة للوصول المتزامن في مجال المختبرات عن بعد والتحقق من صحتها باستخدام اختبارات الحمل. يكمن ابتكار هذا العمل في استخدام المعلمات التي تم جمعها للطالب النموذجي واختبارها باستخدام أداة Artillery.io. بعد ذلك ، قمنا بتقييم أداء المختبر من خلال تحديد 5 ثوانٍ أقصى وقت انتظار لا يمكن تجاوزه. تصف هذه الأطروحة أيضًا حالة استخدام توضح كيف تم تصميم هذه البنية وتطويرها مع مائة طالب.

الكلمات المفتاحية -مختبر عن بعد ، تقاسم الوصول ، اختبار الحمل ، الأداء ،

Abstract

Online labs are a solution for institutions that offer educational programs online and wish to provide hands-on training to their students. Most of the time, some institutions do not have properly equipped laboratories. For these reasons, some institutions have started to use online laboratories, reducing costs in terms of space, equipment, maintenance, personnel, among others. Online labs are an alternative for providing remote access via internet connection to real equipment or virtual experiments located in another location.

Effectively sharing laboratories is interesting for several reasons. It allows students to have more access to the platform for more experiences. Additionally, a feature typically found in remote and traditional labs is that they are unused for a long time. In the case of remote labs, they can be shared regardless of the city, country or continent where the student is located. The sharing of remote laboratories makes it possible to reduce the costs of carrying them out. Even though a remote lab may cost more
than a traditional lab, sharing has the potential to reduce cost.

The objective of this thesis is to propose a service-oriented architecture for simultaneous access in the field of remote laboratories and its validation using load tests. The innovation of this work lies in the use of the parameters collected for the typical student and tested with the Artillery.io tool. Then, we evaluated the performance of the laboratory by defining 5s the maximum waiting time that a request cannot be exceeded. This thesis also describes a use case showing how this architecture was designed and developed with a hundred students.

Keywords— Remote laboratories, access sharing, stress load test, performance

Résume

Les laboratoires en ligne sont une solution pour les établissements qui proposent des programmes éducatifs en ligne et souhaitent offrir une formation pratique à leurs étudiants. La plupart du temps, certaines institutions ne disposent pas de laboratoires correctement équipés. Pour ces raisons, certaines institutions ont commencé à utiliser des laboratoires en ligne, réduisant les coûts en termes d'espace, d'équipement, de maintenance, de personnel, entre autres. Les laboratoires en ligne sont une alternative pour fournir un accès à distance via une connexion Internet à des équipements réels ou à des expériences virtuelles situés dans un autre endroit.

Partager efficacement des laboratoires est intéressant pour plusieurs raisons. Il permet de multiplier les accès des étudiants à la plateforme pour faire plus d'expériences. De plus, une caractéristique généralement présente dans les laboratoires distants et traditionnels est qu'ils sont longtemps inutilisés. Dans le cas des laboratoires distants, ils peuvent être partagés quel que soit la ville, le pays ou le continent où se trouve l'étudiant. Le partage des laboratoires à distance rend possible la réduction des coûts de réalisation de ces derniers. Même si un laboratoire distant peut coûter plus cher qu'un laboratoire traditionnel, le partage a le potentiel de réduire le coût.

L'objectif de cette these est de proposer une architecture orientée services pour un accès simultané dans le domaine des laboratoires distants et sa validion à l'aide de tests de charge. L'innovation de ce travail réside dans l'utilisation des paramètres collectés pour l'élève type et testés avec l'outil Artillery.io. Ensuite, nous avons évalué la performance du laboratoire en définissant 5s le temps d'attente maximum qu'une demande ne peut être dépassée. Cette thèse décrit également un cas d'utilisation montrant comment cette architecture a été conçue et développée avec une centaine d'étudiants.

Mots-clés— Laboratoires distants, partage d'accès, test de charge, performances,

Table des matières

Acronymes

API	Application Programming Interface
DMA	Dynamic Mechanical Analysis
GPIO	General Purpose Input/Output
GUI	Graphical User Interface
IoT	Internet Of Things
JWT	Json Web Token
LaaS	Labs-as-a-Service
LEOG	Laboratoire Électromagnétisme et Optique Guidée
LMS	Learning Management System
LTI	Learning Tools Interoperability
MOOC	Massive Open Online Course
MOODLE	Modular Object-Oriented Dynamic Learning Environment
MOOLs	Massive Open Online Labs
MSOL	Massively Scalable Online Laboratories
RLMS	Remote Laboratories Management System
SBC	Single Board Computer
SOAP	Simple Object Access Protocol
SPST	Single Pole Single Throw
TEL	Enhanced Technology Learning
UML	Unified Modeling Language
UX	User Experience

VISA	Virtual Instrument Software Architecture
WSDL	Web Services Description Language
YAML	Yet Another Markup Language

Table des figures

Liste des tableaux

Introduction générale

Introduction

Les travaux pratiques dans un contexte éducatif sont très importants pour le processus d'apprentissage [Bhu+21]. Les laboratoires en ligne sont une solution pour les établissements qui proposent des programmes éducatifs en ligne et souhaitent offrir une formation pratique à leurs étudiants, et pour les établissements qui n'ont pas les infrastructures pour mettre en place des expérimentations spécifiques. Dans certains cas, en raison de problèmes budgétaires, certaines institutions ne disposent pas de laboratoires correctement équipés ou même n'ont pas de laboratoires. Pour ces raisons, certaines institutions ont commencé à utiliser des laboratoires en ligne, réduisant les coûts en termes d'espace, d'équipement, de maintenance, de personnel, entre autres. Les laboratoires en ligne sont une alternative pour fournir un accès à distance via une connexion Internet à des équipements réels ou à des expériences virtuelles situés dans un autre endroit.

Les laboratoires en ligne sont nés il y a près de deux décennies[Akt+96], et depuis, ils ont été adoptés à maintes reprises dans plusieurs domaines : chimie ([Cob+10] et [You+20]), physique ([Abi+19] et [Bje+17]), électronique ([Gus+07] et [NMN08])

Des efforts ont été déployés pour parvenir à un plus grand degré de collaboration ([Low+09], [Sal+15]), même avec les technologies du monde virtuel ([MUD08], [AYO19] [Eva+17]). Au fil du temps, ils ont été adaptés pour répondre a un très grand nombre d'utilisateurs ([SR14],[ZLB16]), ou utilisant le concept Massively Scalable Online Laboratories (MSOL) qui permet la virtualisation des d'expériences réelles en construisant au préalable une base de données de toutes les mesures possible. ([NHH18],[De +20]).

Au départ, l'objectif était d'améliorer ces laboratoires en leur donnant plus de flexibilité ou de réaliser ce qui pouvait être fait dans un laboratoire physique via Internet. Cependant, le coût de mise en place de ces laboratoires distants devient important si nous n'utilisons pas une nouvelle politique de partage. Cette nouvelle politique de partage des laboratoires et son évaluation pour un dimensionnement efficace est l'objet d'étude de cette thèse.

Contexte

À mesure que les technologies Internet se sont répandues, de plus en plus d'universités à travers le monde ont développé leurs propres laboratoires distants. Mais l'accent a toujours été mis sur l'équipement lui-même, plutôt que sur la gestion ou la communication. Cela a conduit à de nombreux laboratoires distants sans tenir compte des aspects de sécurité, d'évolutivité, de communications, de technologies propres (exigeant que les utilisateurs utilisent un navigateur particulier ou installent un logiciel de plateforme particulier). De plus, le développement de chaque laboratoire distant nécessite à chaque fois beaucoup d'efforts de la part des concepteurs de laboratoire.

Pour faire face à cette situation, des *Remote Laboratories Management System* (RLMS) ont été développés. Il s'agit de systèmes logiciels axés sur

la fourniture d'une solution technique pour le développement de laboratoires distants, la mise en œuvre de fonctionnalités communes et la résolution de nombreux problèmes courants. Par exemple, les RLMS mettent généralement en œuvre l'authentification (à l'aide d'outils tels que : LDAP, OpenID, etc.), le suivi des utilisateurs, la réservation (à l'aide de files d'attente ou sur la base d'un calendrier de réservation) ou l'administration d'outils (ajout d'utilisateurs, etc.). Avec ce logiciel, un développeur de laboratoire distant peut se concentrer sur la livraison de l'équipement sur Internet, et le reste du cycle de vie de l'application peut être géré par le RLMS. De plus, les nouvelles versions de ces RLMS peuvent être accompagnées de nouvelles fonctionnalités transparentes : si un RLMS s'intègre à un système de gestion de l'apprentissage, automatiquement tous les laboratoires distants développés avec ce RLMS seront intégrés.

Une nouvelle fonctionnalité que RLMS peut prendre en charge est la configuration d'un partage qui optimise l'utilisation des ressources.

Motivations

Partager efficacement des laboratoires est intéressant pour plusieurs raisons. Il permet de multiplier les accès des étudiants à la plateforme pour faire plus d'expériences. De plus, une caractéristique généralement présente dans les laboratoires distants et traditionnels est qu'ils sont longtemps inutilisés. Dans le cas des laboratoires distants, ils peuvent être partagés quel que soit la ville, le pays ou le continent où se trouve l'étudiant. Le partage des laboratoires à distance rend possible la réduction des coûts de réalisation de ces derniers. Même si un laboratoire distant peut coûter plus cher qu'un laboratoire traditionnel[Low12], le partage a le potentiel de réduire le coût.

L'intérêt pour le partage des laboratoires distants est grandissant. En effet, la commission de l'union européenne à débloquer plus de 60 millions d'euros en termes de recherche et projets en Enhanced Technology Learning (TEL). L'un des résultats est précisément « Le support de la fédération et l'utilisation des laboratoires distants pour l'apprentissage et l'enseignement». En effet, le projet Go-Lab [1] financé à hauteur de 10 millions d'euros, vise à soutenir une large fédération de laboratoires distants. Parallèles et connexes efforts ont été placés sur des systèmes qui répertorient les laboratoires distants situés dans différentes institutions telles que Lab2Go [MN10] ou même accorder l'accès à des laboratoires comme LiLa [RBJ11].

Dans un même temps, des Massive Open Online Course (MOOC) tels que Coursera , edX ou Udacity soutiennent des cours à grande échelle avec des milliers d'étudiants. La partie expérimentale des cours techniques qui nécessiterait généralement un laboratoire est gérée à l'aide de simulateurs. Des travaux comme ([Low14],[SGP16]) ont tentés de mettre des exemples de Massive Open Online Labs (MOOLs) pour un grand nombre d'utilisateurs. L'un des objectifs de cette thèse est précisément d'explorer comment atteindre un plus grand nombre d'utilisateurs pour un seul laboratoire.

Objectifs de la recherche

Cette thèse vise la mise en place d'une nouvelle approche pour la conception des laboratoires distants en se concentrant sur de nouvelles techniques de partage et d'ordonnancement.

Cette approche offrirait des performances plus efficaces et un environnement bien organisé et structuré. En raison de la complexité des laboratoires distants, les problèmes tels que le partage et l'ordonnancement sont

1. https://www.golabz.eu/

des domaines d'amélioration très riches. Cette thèse aborde donc ces domaines pour trouver de nouvelles idées, techniques et solutions.

La nouvelle approche proposée dans cette thèse est basée sur une architecture orientée microservices. Cette dernière sera présentée et évaluée en utilisant un test de charge pour répondre aux questions de recherches suivantes :

— Comment concevoir une architecture interopérable capable de partager les laboratoires distants ?

— Combien d'utilisateurs simultanés peuvent être servi par une seule instance de laboratoire si, en définissant 5s, le temps de réponse maximal qu'une requête ne peut dépasser ?

Organisation de la thèse

À partir du chapitre 1, une introduction sur les laboratoires distants, les RLMSs, et les techniques de partages des laboratoires distants et des travaux antérieurs dans ce domaine de recherche est présentée. Dans le chapitre 2, nous donnons l'architecture de la solution, le modèle et les algorithmes pour le partage des laboratoires distants. Les expériences, les résultats et la discussion seront fournis dans le chapitre 3. Une conclusion de notre travail de recherche et les travaux futurs sont présentés. L'annexe I contient les fichiers YAML pour implémenter la solution de test de charge. La description des données de test sera présentée en annexe II.

Chapitre1

État de l'art

1.1 Introduction

Les travaux pratiques distants sont récemment apparus comme une nouvelle forme d'outils d'apprentissage basés sur des composants matériels et logiciels. Ils offrent à ses utilisateurs la possibilité d'interagir avec des instruments et du matériel n'importe où et n'importe quand ([GZ08],[NMN03]).

Les travaux pratiques distants sont attrayants, car ils éliminent l'obligation pour ses utilisateurs d'être présents physiquement en face des équipements [MN06]. Ils permettent aux étudiants plus du temps de manipulation par rapports aux travaux pratiques en présentiel. Cependant, malgré cela, le développement de techniques pour rendre les travaux pratiques distants efficaces en est actuellement à ses débuts, avec de nombreux problèmes à résoudre.

Dans ce chapitre, nous fournissons un aperçu des principaux concepts et définitions qui sous-tendent le domaine des travaux pratiques distants. Tout d'abord, dans la section 1.2, nous définissons ce qu'un laboratoire distant, nous décrivons son architecture et présentons les technologies les plus importantes derrière cet outil d'apprentissage en ligne. Ensuite, dans la section 2.2, nous nous concentrons sur la gestion des laboratoires distants, qui est le principal sujet de recherche de cette thèse.

1.2 Présentation des laboratoires distants

Les laboratoires distants ont gagné en popularité depuis l'adoption rapide d'Internet entre le milieu et la fin des années 90. La progression rapide des laboratoires distants a été provoquée par la pression financière croissante exercée sur les universités pour fournir une expérience pratique

adéquate aux cours liés à l'ingénierie. Les laboratoires à distance visent à résoudre de nombreux problèmes rencontrés avec l'expérience de laboratoire universitaire moderne car ils :

— Prolongent l'expérience d'apprentissage des étudiants en utilisant plus efficacement leur temps [Alv+16].

— Augmentent l'exposition des étudiants à une plus grande gamme d'expériences et des équipements de laboratoire.

— Réduisent les coûts, car les coûts de démarrage des RL sont minimes par rapport aux laboratoires traditionnels, et ils s'adressent à un plus grand groupe d'étudiants

— Donnent aux étudiants externes, la possibilité de mener des expériences en laboratoire.

— Offrent la possibilité de fournir/démontrer du matériel pédagogique localement et à l'extérieur.

— Offrent aux étudiants la possibilité de mener des expériences à des moments qui leur conviennent. [MNN11]

Dans la section suivante, nous fournissons un aperçu des caractéristiques fondamentales des laboratoires distants et nous présentons quelques défis de recherche ouverts.

1.3 Définition des laboratoires distants

Dans la littérature scientifique, il n'y a toujours pas de définition universellement acceptée de ce qu'est un laboratoire distant et des caractéristiques uniques qu'il devrait offrir. Par exemple, Chen et al. [CSZ10] fournissent la définition suivante :

"Un laboratoires distant, par définition, est une expérience qui est menée et contrôlée à distance via Internet. Les expériences utilisent des composants réels ou des instruments à un endroit différent de celui où ils sont contrôlés ou conduits".

Umborg et Uukkivi [UU20], ont défini les laboratoires distants comme ceci :

« Les laboratoires distants utilisent des moyens de télécommunication pour mener à distance des expériences de laboratoire avec de vrais appareils de mesure et des objets réels pour une investigation. » [UU20]

Un laboratoire distant est un laboratoire physique auquel on peut accéder à distance avec sécurité et à faible risque. L'accès se fait par un système basé sur le Web pour faire fonctionner à distance un équipement spécialisé. [SPM20]

Le laboratoire distant est un laboratoire virtuel qui peut donner accès via le navigateur Web à l'équipement réel dans un vrai laboratoire. Ce laboratoire permet d'envoyer des commandes de contrôle qui peuvent être prétraitées afin de ne pas endommager le réel équipement coûteux, d'exécuter des expériences dans un vrai laboratoire sur l'équipement réel, puis de collecter des résultats expérimentaux. [Ale+19]

S'il existe un grand nombre d'autres définitions, la majorité d'entre elles semblent s'accorder sur deux caractéristiques clés :

— Les laboratoires à distance offrent aux étudiants la même expérience que les laboratoires en face-à-face car les circuits sont construits avec des composants réels sur les cartes et les résultats sont affichés à partir de mesures réelles prises à partir d'instruments réels tels que des oscilloscopes, des multimètres, etc. [Sch+16]

— Une autre caractéristique importante des laboratoires distants est que les expériences peuvent être effectuées à tout moment et de n'importe où, sans la présence de l'enseignant. Cette caractéristique offre

> plus de flexibilité aux étudiants qui peuvent accéder aux laboratoires distants pour terminer les activités chez eux. De plus, certains laboratoires distants ont des salles de discussion virtuelles pour discuter avec un professeur-tuteur pour résoudre des problèmes plus difficiles. [Sch+16]

Ensemble, ces deux caractéristiques, qui sont probablement les plus importantes pour les laboratoires distants, transforment la manière traditionnelle dont les laboratoires sont menés, détournant l'attention de la vision classique des travaux pratiques.

1.4 Classification des laboratoires distants

Les laboratoires distants sont des expériences accessibles depuis Internet. Ils sont divisés en deux catégories principales : les laboratoires virtuels et les laboratoires distants.

La figure 1.1 décrit la classification des laboratoires en général. Les laboratoires virtuels sont souvent utilisés dans le cas où la configuration du laboratoire est trop difficile, voire impossible. La simulation aide les étudiants à améliorer leurs connaissances et à se faire une idée approximative du comportement du monde «réel». Aussi, les laboratoires virtuels diffèrent des laboratoires distants, car ils n'utilisent que des logiciels tandis que les «laboratoires distants» sont constitués de matériel réel.

Par rapport aux laboratoires virtuels, les laboratoires distants permettent aux étudiants de manipuler du matériel réel. Comparé aux laboratoires traditionnels (en présentiel), les laboratoires distants offrent les avantages de réduction des coûts et de flexibilité. De plus, à mesure que les stratégies éducatives collaboratives se généralisent, les laboratoires distants offrent de

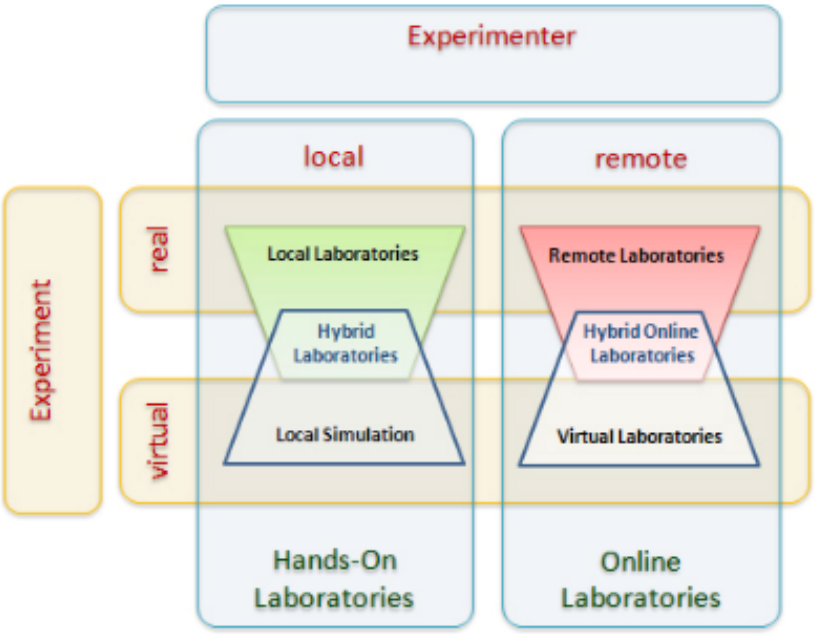


FIGURE 1.1 – Classification des laboratoires distants [Zut+10].

grandes opportunités aux étudiants d'interagir alors qu'ils travaillent pour un objectif commun. [Zut+10]

En ce qui concerne l'interactivité. Il existe deux approches principales : les laboratoires synchrones, qui garantissent un lien permanent avec le laboratoire et les laboratoires en *batch*, où les étudiants soumettent des tâches qui sont exécutées en mode asynchrone (la tache ne peut être modifiée en cours d'exécution). Cette classification a un impact sur les schémas de l'ordonnanceur, car les laboratoires asynchrones utilisent généralement une file d'attente alors que les laboratoires synchrones utilisent une réservation basée sur un calendrier en fonction de la durée de session requise. [San+12]

1.5 Composition des laboratoires distants

Comme nous l'avons déjà souligné précédemment, les laboratoires distants combinent deux composants principaux :

— Matériel : c'est l'environnement physique (matériel) où les expériences seront exécutées par les utilisateurs distants. Ce matériel inclus tous équipements ou matériaux utilisés lors de la réalisation ou l'exécution des expériences tel que les instruments de mesures, des objets connectés *Internet Of Things* (IoT), les circuits électroniques, des produits chimiques, etc.

— Logiciel : Il peut également être appelé système de gestion de laboratoire distants (*RLMS*), l'environnement ou la partie logique du laboratoire incluant tout composant logique utilisés lors de la réalisation du laboratoire, tel que l'application permettant a l'utilisateur d'accéder au laboratoire (notamment des applications web), les systèmes de gestion des utilisateurs, gestion de l'authentification, gestion des expériences, etc.

La figure 1.2 montre un exemple de composition d'un laboratoire distant utilisant plusieurs instruments de mesures, une carte de travaux pratiques et un serveur pour servir plusieurs utilisateurs distants.

1.6 Les défis des laboratoires distants

Bien que, les laboratoires distants présentent un certain nombre d'avantages déjà cités en section précédente, ils ne sont pas une solution simple à implémenter et ils nécessitent beaucoup de temps et de ressources humaines. Aussi, la disponibilité des Laboratoires distants constitue un pro-

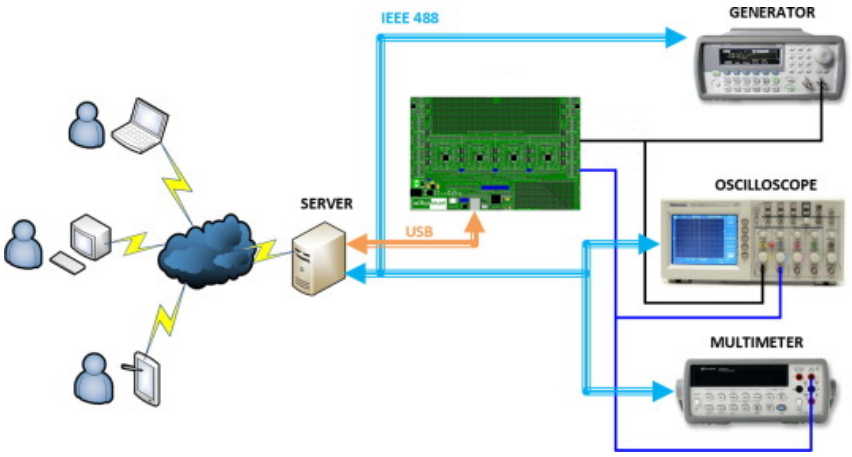


FIGURE 1.2 – Composition d'un Labo distant [Ang+20]

blème important dans la formation des ingénieurs et ceci pour deux raisons : d'une part, la gestion de laboratoire peut être coûteuse en ressources, car elle nécessite un entretien continue du matériel ; d'autre part, en raison de facteurs économiques (duplication) des expériences scientifiques.

Un autre défi lié à l'utilisation des laboratoires distants est la sécurité après la phase de déploiement : par définition, les laboratoires distants sont connectés à Internet et donc sujets aux attaques malveillantes. Pour faire face à ce type d'attaque, les serveurs du laboratoire doivent être à jour, ainsi que le pare-feu. Le serveur d'expérimentation à distance doit être configuré pour bloquer et signaler rapidement toute activité suspecte.[SG07]

1.7 Système de gestion des laboratoires distants

Cette section présente le concept de *Remote Laboratories Management System* (RLMS), qui est l'un des principaux composants de l'architecture logicielle des laboratoires distants. Nous allons donner un aperçu de la

conception de ces systèmes, comment ils sont implémentés, et aussi nous allons montrer les différentes fonctionnalités proposées par ces derniers.

Tout laboratoire distant a besoin d'un système pour assurer la gestion de l'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 général indépendantes du type de laboratoires et elles sont considérées comme des services transversaux.[RP18]

1.8 Les RLMS :une vue fonctionnelle

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

Gestion des laboratoires et d'expériences : un RLMS doit offrir la possibilité de gérer plusieurs laboratoires qui peuvent être géodistribués. C'est souvent le cas quand plusieurs laboratoires sont maintenus par différentes universités ou différentes facultés de façon collaborative. Il offre aussi la possibilité de la gestion des différentes expériences dans ces laboratoires.

Gestion des utilisateurs : un RLMS doit aussi offrir la possibilité de gérer les utilisateurs qui veulent manipuler des expériences. Cette tâche peut inclure :

Inscription : l'inscription des utilisateurs sur le système afin d'exploiter ou utiliser ces laboratoires.

Authentification : si l'inscription des utilisateurs est obligatoire, le RLMS doit posséder un moyen efficace pour authentifier les utilisateurs. Aujourd'hui, il existe plusieurs solutions d'authentification des utilisateurs.

Gestion des accès et des autorisations : Ce service garanti ou limite les actions ou les opérations dans le système qui sont effectuées par un utilisateur sur un ensemble de ressources. En bref, il applique la politique de

contrôle d'accès du système. Aussi, Ce dernier est chargé d'accorder ou de refuser à un utilisateur l'accès à une ressource.[RP18].

Gestion des sessions : un rôle indispensable du RLMS est la gestion des sessions d'utilisateurs sur les laboratoires, surtout dans le cas où il existe un nombre important d'utilisateurs voulant accéder à des nombres limités de laboratoires en même temps, dans ce cas le RLMS doit posséder un moyen efficace qui permet de gérer les sessions.

Autres : un RLMS peut aussi inclure d'autres fonctionnalités complémentaires tel que : l'analyse d'utilisation des laboratoires, l'analyse de la progression des étudiants, outil de comptabilité, outils d'intégration avec autres plateformes tel que moodle par exemple, etc.

Il faut noter qu'en général l'interface graphique *Graphical User Interface* (GUI) en anglais du laboratoire distant ne fait pas partie du RLMS lui-même, mais du laboratoire. Puisque dans la plupart des cas, le RLMS est responsable seulement de la gestion des laboratoires et des utilisateurs, mais pas du fonctionnement du laboratoire lui-même. Cette approche offre plus de flexibilité dans la conception et développement des laboratoires personnalisés intégrables dans le système (RLMS).

1.9 Implémentation des RLMS

Les systèmes de gestion des laboratoires distants sont généralement implémentés comme des portails web (application web) offrant à l'utilisateur final plus de flexibilité. Il suffit donc juste d'avoir un navigateur web déjà présent dans tous les systèmes d'exploitations (Windows, linux, MacOs, Android, …) et un appareil de type (micro-ordinateur, téléphone portable, Tablette, …), pour pouvoir accéder aux laboratoires distants.

La figure 1.3 illustre un exemple d'implémentation d'un RLMS. Ce dernier intervient en tant qu'intermédiaire entre un réseau moins sécurisé qui

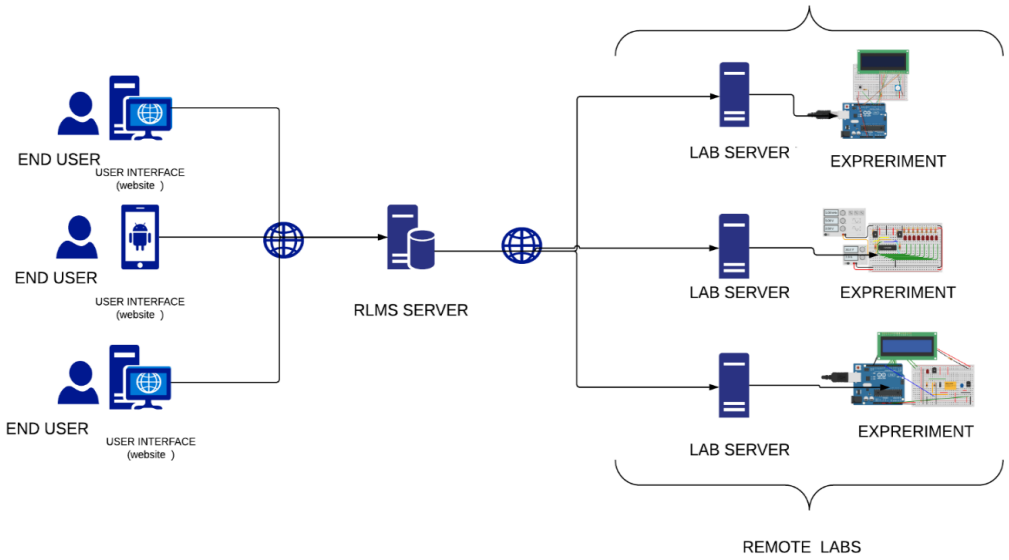


FIGURE 1.3 – RLMS : Exemple d'un déploiement [Ism19]

est internet et le réseau plus sécurisé contenant l'ensemble des serveurs d'expériences.

1.10 PARTAGE DES LABORATOIRES DISTANTS

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

1. Laboratoires distants public : la première approche consiste à laisser le laboratoire accessible 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. Partager les comptes entre les différents RLMS : si l'Université A souhaité utiliser les laboratoires distants de l'Université B, une personne de l'Université A fournira une liste des noms d'utilisateurs à l'Université B et les étudiants iront dans cet établissement à l'aide des informations d'identification de l'Université B. Idéalement, une authentification fédérée peut être utilisée pour éviter de fournir des informations d'identification dans différents domaines tels que OAuth [1].
3. Laboratoires fédérés : si un RLMS prend en charge la fédération, dans deux institutions différentes (par exemple, l'Université A 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 une 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.

1.11 Exemples des solution *RLMS*

1.11.1 iLab Shared Architecture

Le projet iLab du *Massachusetts Institute of Technology* est une boîte à outils logicielle distribuée et une infrastructure de services *middleware* [2] pour prendre en charge les laboratoires distants et promouvoir leur partage entre les écoles et les universités du monde entier. L'architecture iLab se concentre sur le développement rapide de laboratoires indépendants de

1. OAuth (Open Authorization) est un cadre d'autorisation standard ouvert pour l'autorisation basée sur des jetons sur Internet.

2. Le terme middleware, traduit en français par logiciel médiateur ou intergiciel, provient de la contraction entre les mots middle (milieu) et software (logiciel).

la plateforme, un accès évolutif pour les étudiants et une gestion efficace pour les fournisseurs de laboratoires tout en préservant l'autonomie des enseigants.

Le *ISA batched architecture* [Har04]ressemble beaucoup à l'architecture Web à trois niveaux (voir figure 1.4) :

— Le premier niveau est l'application cliente de l'étudiant qui s'exécute généralement sous forme d'applet ou d'application téléchargée sur le poste de travail de l'étudiant.

— Le niveau intermédiaire, appelé service courtier en anglais *Service Broker*, fournit les services communs partagés. Il s'appuie sur une base de données relationnelle standard telle que Microsoft SQL Server ou MySQL. Le client de l'étudiant communique uniquement avec le service courtier, qui transmet les spécifications de l'expérience au troisième niveau final qui comprend l'équipement de laboratoire. Contrairement à l'architecture Web standard à trois niveaux dans laquelle le niveau intermédiaire réside sur le côté entreprise plutôt que sur le côté client du réseau, le service courtiers réside normalement sur un serveur de l'établissement de l'étudiant. Si une université est disposée à fournir des comptes aux utilisateurs d'autres institutions, l'architecture permet au service courtier de s'exécuter sur un campus distinct du client.

— Le troisième niveau est le serveur de laboratoire, qui s'interface avec les instruments qui exécutent les expériences spécifiées. Le serveur de laboratoire avertit le service courtier lorsque les résultats sont prêts à être récupérés.

Dans l'architecture iLab, le client et le serveur de laboratoire représentent tous deux les modules logiciels dépendant du domaine et du laboratoire. Le service courtier est un code complètement générique et peut interagir

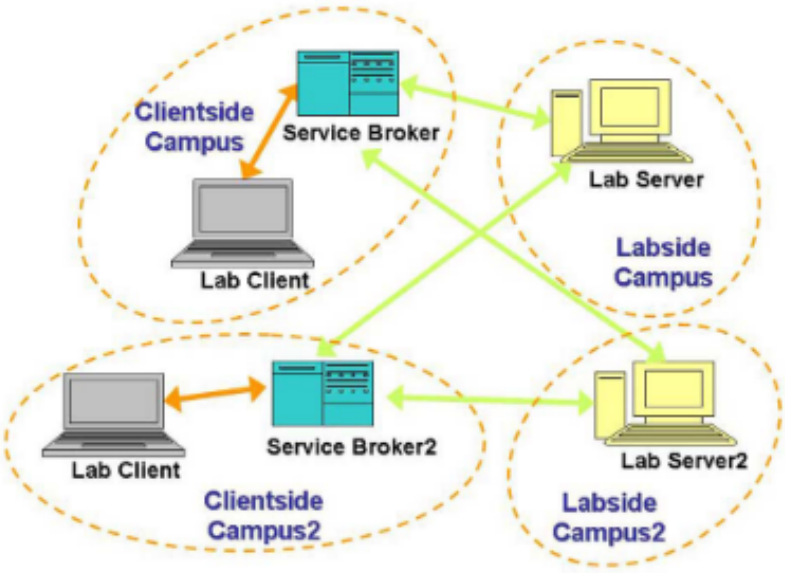


FIGURE 1.4 – Architecture de iLab[Har+08]

avec n'importe quelle combinaison de client et de serveur de laboratoire qui implémentent les interfaces appropriées exprimées en termes d'appels de service Web Simple Object Access Protocol (SOAP) définis dans Web Services Description Language (WSDL) [Har04].

1.11.2 RemLabNet

Le système RemLabNet, destiné à l'intégration et à la gestion d'expériences à distance pour les nouveaux étudiants des niveaux universitaire et secondaire [Sch+14] . Sa construction a été initiée à la fois par l'utilisation intensive et l'expertise du système ISES (Internet School Experimental System)[Sch+08], par la réalisation d'expériences à distance et par l'absence d'un système similaire pour les écoles secondaires en Europe. RemLabNet 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 cloud virtualisé seront utilisés.[BS17]

Toutes les fonctionnalités de REMLABNET en Figure 1.5 sont physiquement situées dans le cloud virtualisé avec tous les composants nécessaires à la communication, en utilisant les technologies *vSphere® et vCloud® de VMWare*. Chaque serveur existant dans RemLabNet fonctionne comme une machine virtuelle (VM) dans le cloud virtualisé avec un système d'exploitation spécifique (*Windows server 2012 R2*, *Suse Linux 11* etc.) et utilisant des moyens de communication spécifiques. Ainsi, le système utilise moins de serveur physique. [BS17]

Internet School Experimental System ISES est un système open source fonctionnant sous le système d'exploitation Windows. Le système se compose d'une carte d'interface, d'un ensemble de modules variables et d'éléments de détection. ISES utilise une carte d'acquisition 12 bits, pour la conversion analogique/numérique elle prend seulement 0,010 ms pour le temps de conversion. La carte de contrôle universelle Dynamic Mechanical Analysis (DMA) et un ensemble de capteurs (environ 40 pour la physique, la chimie et la biologie, etc.). Le système offre la possibilité de mesurer et d'afficher simultanément les données de 8 canaux d'entrée et de contrôler le processus via deux canaux de sortie analogiques et quatre canaux de sortie binaires.

L'état actuel des expériences à distance dans ce projet de laboratoire électronique est accessible via la page Web du projet [3]

— Contrôle du niveau d'eau.

3. http://www.ises.info

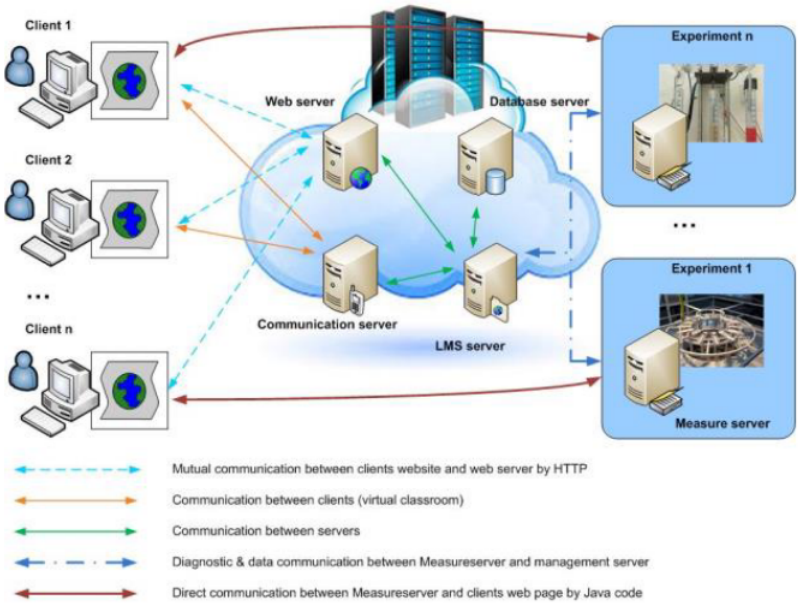


FIGURE 1.5 – Fonctions générales de RemLabNet [BS17]

— Station météorologique de Prague.
— Induction électromagnétique.
— Oscillations naturelles et forcées.
— Diffraction sur des micro-objets.
— Conversion de l'énergie solaire et principe d'incertitude de Heisenberg.

1.11.3 WebLab-Deusto

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.

Dans les deux premières versions de WebLab-Deusto, il s'agissait d'un laboratoire à distance adhoc qui est devenu un système de laboratoire à

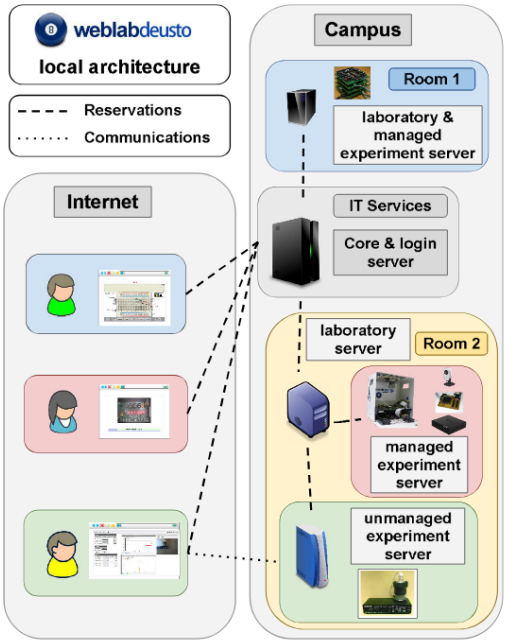


FIGURE 1.6 – Architecture de WebLabDeusto [Kal+13]

distance sur lequel différents laboratoires pouvaient être construits à l'aide de différentes technologies logicielles. Aujourd'hui WeblLab Deusto supporte la fédération de laboratoires distants. L'architecture WebLab-Deusto (voir Figure 1.6) est une architecture client-serveur. Les étudiants interagissent généralement avec le serveur de connexion pour l'authentification et avec le serveur principal pour le reste des opérations. Le serveur principal gérera l'autorisation, la planification et le stockage des informations envoyées par les étudiants, ainsi que la transmission de toutes les demandes au laboratoire approprié. Derrière le serveur principal, l'architecture WebLab-Deusto définit deux autres serveurs : le serveur de laboratoire et le serveur d'expérimentation. Le premier vise à être situé dans la salle de laboratoire physique où plusieurs serveurs d'expérimentation sont attendus. Son but est principalement d'assurer la sécurité des serveurs d'expérimentation, ainsi que de vérifier périodiquement s'ils fonctionnent toujours. Le

serveur d'expérimentation est le serveur qui interagit physiquement avec le matériel.

1.12 Intégration des outils d'apprentissage LTI

Aujourd'hui, le domaine des sciences de l'éducation comprend un nombre croissant d'applications Web de haute qualité qui améliorent l'enseignement et l'apprentissage. Citons par exemple de simples applications pour la communication ou de chat jusqu'à des moteurs d'apprentissage spécifiques à un domaine pour des matières particulières comme les mathématiques ou les laboratoires distants.

Les systèmes de gestion de l'apprentissage en anglais *Learning Management System* (LMS) devrait fournir un accès à ces innombrables applications d'apprentissage. Il devrait être possible de combiner ces applications dans le contexte d'un cours donné. À cette fin, certains fournisseurs de LMS ont développé des bibliothèques d'extension propriétaires qui permettent d'intégrer des applications externes. Les enseignants et les étudiants naviguent dans les applications d'apprentissage en parcourant des hyperliens soigneusement conçus et des flux de données entre les deux systèmes via des protocoles de communication personnalisés.

La norme *Learning Tools Interoperability* (LTI) [4] vise à fournir un cadre unique pour l'intégration de n'importe quel produit LMS avec n'importe quelle application d'apprentissage.

4. IMS Global est le leader mondial de l'interopérabilité, de l'innovation et de l'apprentissage dans le domaine de l'innovation et de l'apprentissage.https://www.imsglobal.org

Objectifs de la norme LTI

— Fournir un modèle de déploiement simple composé d'une URL, d'une clé et d'un mot de passe que l'administrateur LMS ou l'enseignant concepteur du cours peut saisir dans le LMS.

— Définir un protocole pour lancer une application externe à partir d'un LMS d'une manière qui prend en charge l'authentification unique et qui préserve le contexte d'apprentissage et les rôles des utilisateurs dans ce contexte.

— Rendre les liens vers des applications externes portables en définissant des éléments de données pouvant être intégrés dans des composants communs.

Dans ce qui suit, nous présentons les fondements conceptuels et les utilisés dans le LTI.

1.12.1 Fonctionnement de LTI

Le flux de travail de base pour l'utilisation de LTI démarre lorsque l'enseignant ou l'administrateur du LMS accède à l'outil d'apprentissage hébergé en externe. L'administrateur de l'outil fournit à l'enseignant une URL, une clé et un mot de passe pour l'outil.

Pour le cas d'utilisation de l'enseignant, ils ajoutent généralement un outil LTI dans leur structure de cours en tant que lien de ressource à l'aide du panneau de configuration LMS. L'instructeur saisit l'URL, le code secret comme méta-données pour le lien de ressource. Lorsque les étudiants sélectionnent l'outil, le LMS utilise l'URL, le secret et les informations clés

pour lancer de manière transparente dans un élément <iframe> [5] ou une nouvelle fenêtre de navigateur.

Les administrateurs de l'outil LTI ajoutent un « outil virtuel » au LMS, en saisissant l'URL, une clé et un mot de passe. Une fois cela fait, les enseignants voient simplement l'outil LTI nouvellement configuré comme un autre outil ou activité en tant que lien de ressource dans leur structure de cours. Les enseignants et les étudiants peuvent même ne pas savoir que l'outil qu'ils utilisent fonctionne en dehors du LMS. Ils sélectionnent et utilisent simplement l'outil comme n'importe quel autre outil intégré au LMS.

Dans les deux cas, l'outil externe reçoit une demande de lancement qui inclut l'identité de l'utilisateur, les informations sur le cours, les informations sur le rôle, ainsi qu'une clé et une signature. Les informations de lancement sont envoyées à l'aide d'un formulaire HTTP généré dans le navigateur de l'utilisateur avec les éléments de données LTI dans des champs de formulaire masqués et automatiquement soumises à l'outil externe à l'aide de JavaScript. Les données du formulaire HTTP sont signées à l'aide de la norme de sécurité OAuth (`www.oauth.net`) afin que l'outil externe puisse être assuré que les données de lancement n'ont pas été modifiées entre le moment où le LMS a généré et signé les données et le moment où l'outil a reçu les données.

Une fois la demande de lancement reçue, l'outil peut choisir de rediriger le navigateur de l'utilisateur vers une autre URL, ou il peut afficher immédiatement l'interface utilisateur demandée.

5. L'élément HTML <iframe> représente un contexte de navigation imbriqué qui permet en fait d'obtenir une page HTML intégrée dans la page courante. Source : `https://developer.mozilla.org/fr/docs/Web/HTML/Element/iframe`

1.12.2 Terminologie

Alors que les principaux cas d'utilisation de la norme IMS LTI consistent à connecter des outils aux systèmes de gestion de l'apprentissage, nous imaginons qu'il y aura de nombreux autres cas d'utilisation qui utiliseront cette spécification comme une simple authentification unique (SSO) qui inclut le rôle et le contexte. Ainsi, dans les documents de spécification, nous utilisons une terminologie plus abstraite pour décrire un LMS, un outil d'apprentissage et un cours.

Outil Fournisseur

Un outil fournisseur (OF) expose des interfaces à un ou plusieurs outils. Comme l'illustre la figure 1.7, il est utile de considérer l'outil fournisseur comme une enveloppe autour d'un outil. En ce sens, OF représente l'ensemble du système ainsi que ses interfaces.

Outil Consommateur

Les enseignants et les étudiants accèdent généralement aux outils en activant des liens à partir d'un LMS. Mais la norme LTI prend également en charge d'autres scénarios. Par exemple, nous pouvons lancer un outil d'apprentissage à partir de Facebook. Tout système qui offre l'accès à un outil est appelé un outil consommateur (OC).

Le contexte

Les liens vers les outils apparaissent généralement dans le contexte d'un cours. Mais ils peuvent aussi apparaître au sein d'autres types de groupes. Par exemple, des liens peuvent apparaître dans le contexte d'une organisation, d'un club, d'un groupe d'étude, etc. Étant donné que les outils

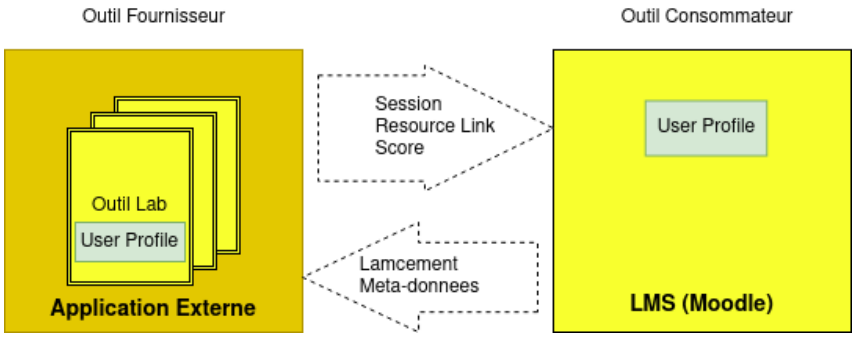


FIGURE 1.7 – LTI interopérabilité figure adaptée depuis [Sch+17]

peuvent être lancés à partir de nombreux types de contextes différents, la spécification LTI utilise rarement le terme « cours » et utilise à la place le terme plus général "contexte". [Ale+17]

LIEN DE RESSOURCE

L'outil consommateur utilise des entités *Resource Link* pour générer des liens cliquables dans l'interface utilisateur. Chaque *Resource Link* a un titre qui définit le texte qui doit apparaître dans le lien cliquable ainsi qu'une description facultative qui doit apparaître à côté du lien.[Con21]

Après avoir présenté les concepts clés présentés dans la norme LTI. Dans la section suivante, nous donnons un exemple de cas d'utilisation qui explique en détail les étapes de fonctionnement du LTI.

CAS D'UTILISATION LTI

Cette section décrit les cas d'utilisation pour la transmission de messages du consommateur d'outils au fournisseur d'outils.

Dans ce cas d'utilisation (voir figure 1.8), l'utilisateur accède initialement dans l'interface utilisateur de l'outil consommateur. Ensuite, il est transféré vers l'interface utilisateur à l'outil fournisseur. La séquence comprend un protocole de connexion sécurisé afin que l'outil fournisseur puisse

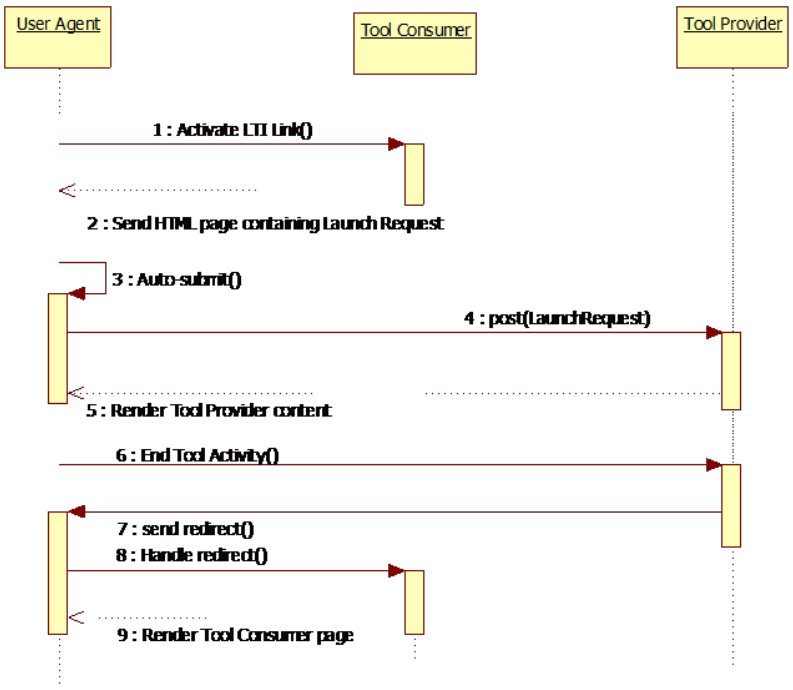


FIGURE 1.8 – Diagramme de séquence illustrant le contenu fourni à l'utilisateur par le fournisseur d'outils. [IMS14]

authentifier l'utilisateur sur la base des informations d'identification signées par l'outil e consommateur. La demande de lancement de l'outil consommateur à l'outil fournisseur comprend suffisamment d'informations sur le contexte d'apprentissage pour garantir une expérience utilisateur cohérente.

Le diagramme de séquence comprend neuf étapes, comme suit :

1. **Activation du lien**. L'utilisateur clique sur un lien affiché dans son navigateur par l'outil consommateur et la requête HTTP est envoyée au service de lancement d'outils au sein de l'outil consommateur.
2. **L'envoi de la page HTML contenant la demande de lancement**. Le service de lancement d'outils génère une page HTML contenant

un formulaire dont l'attribut « méthode » est « POST ». Les champs du formulaire correspondent à une sérialisation [6] du message *LTI Launch Request*.

3. **Soumission automatique** : La page HTML inclut du code JavaScript pour soumettre automatiquement le formulaire. La cible du formulaire doit être l'URL du gestionnaire de messages approprié dans le système du fournisseur d'outils, tel que défini dans le profil de l'outil.
4. **Publication de la demande de lancement.** Le navigateur de l'utilisateur publie le formulaire contenant la demande de lancement sur le serveur du fournisseur d'outils.
5. **Rendu du contenu du fournisseur d'outils.** Le fournisseur d'outils valide la demande de lancement conformément au profil de sécurité négocié lors du processus de déploiement de l'outil. Entre autres, la demande de lancement comprend un paramètre « URL de retour ». Le fournisseur d'outils enregistre cette URL (éventuellement en tant que cookie), afin qu'il puisse rediriger le navigateur de l'utilisateur vers le système du consommateur d'outils lorsque l'utilisateur a fini d'interagir avec l'outil. (Voir l'étape 7.) Enfin, le fournisseur d'outils répond à la demande de lancement en envoyant le contenu demandé au navigateur de l'utilisateur.
6. **Terminer l'activité de l'outil.** L'utilisateur interagit avec l'outil et exécute éventuellement une action pour « fermer » l'outil.
7. **Envoyer la redirection.** Lorsque le fournisseur d'outils reçoit une demande de fermeture de l'outil, il redirige le navigateur de l'utilisateur vers l'« URL de retour » qu'il a reçue lors du lancement de

6. La sérialisation des messages est la façon dont les données dans la mémoire d'un ordinateur sont converties sous une forme dans laquelle elles peuvent être communiquées.https://fr.wikipedia.org/wiki/Sérialisation

l'outil à l'étape 4. Le message de redirection HTTP 301 doit être envoyé à la même fenêtre ou cadre dans lequel l'outil a été lancé.

8. **Gérer la redirection.** Le consommateur d'outils reçoit la demande de la redirection.
9. **Outil de rendu de la page Consommateur.** L'Outil Consommateur reprend le contrôle de l'expérience utilisateur.

D'après les étapes précédentes, nous pouvons conclure que le concept LTI est très adapté pour le développement des outils fournisseurs pour les laboratoires distants. Il permet d'implémenter d'une manière transparente et sans impacter ou changer le LMS de l'établissement.

1.13 Conclusion

Dans ce chapitre, nous avons donné un aperçu sur les laboratoires éducatifs en général et les laboratoires distants en particulier. Ainsi, nous avons montré les avantages de ces laboratoires et comment ils ont participé à l'amélioration de l'expérience éducative des étudiants. Nous avons dénombré les architectures existantes des laboratoires distants tout en incluant les différentes fonctionnalités de leurs systèmes informatiques. Entre autres, l'aspect architectural et/ou conceptuel. Aussi, sur le plan interopérabilité, nous avons montré comment le concept LTI peut être flexible dans l'implémentation des outils fournisseur comme les laboratoires distants.

Chapitre2

MostaLab : Architecture et la gestion des accès simultanés

2.1 Introduction

Le chapitre précédent nous a permis de donner une vue globale sur les laboratoires distants en général et les systèmes des gestions de ces laboratoires plus précisément. Dans ce chapitre, nous allons décrire les étapes de conception de notre propre solution de gestion des laboratoires distants en se basant sur des solutions open source existantes.

Le résultat final sera un système de gestion de laboratoires fronté par une interface Web (application Web) et soutenu par une solution orientée services pour la gestions des accès aux laboratoires distants.

2.2 Conception de la solution

Dans cette section, nous proposons une solution fiable et maniable pour faciliter les tâches reliées à la problématique de la gestion des laboratoires distants.

Afin d'élaborer la solution 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 fonctionnelles ou contraintes non fonctionnelles (techniques).

2.2.1 Identification des acteurs

Les acteurs principaux que nous avons réussi à identifier sont :

Administrateur système : c'est l'acteur qui a pour rôle principal d'administrer le système, il est chargé des tâches de configuration telle que par exemple : l'intégration des nouveaux laboratoires au système. Il est à noter que cet acteur possède un contrôle total sur le système. L'administrateur peut aussi hériter des autorisations d'un autre acteurs.

Exemple : un agent avec des autorisations de gestion de quelques catégories de laboratoires distants.

Client : Différents type d'utilisateurs finaux qui n'ont pas tous les mêmes privilèges :

— Fournisseur,

— Enseignant,

— Étudiant,

2.2.2 Contraintes fonctionnelles

Dans cette section, nous détaillons les contraintes fonctionnelles majeures à respecter lors de la conception de la solution.

La première contrainte est que la solution doit assurer toutes les fonctionnalités basiques de tels systèmes qui peuvent être résumés dans les trois principaux points suivants :

— Gestion des laboratoires.

— Gestion des utilisateurs des laboratoires.

— Gestion des accès aux laboratoires.

La deuxième contrainte est que l'intégration des nouveaux laboratoires doit être assurée. 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 d'une manière fréquente.

Une autre contrainte à prendre en compte est la montée en charge *scalability*. C'est un facteur très important qu'il faut prendre en considération lors de la conception de cette solution. Comme nous l'avons déjà mentionné, ce genre de systèmes est un système très évolutif. Donc il doit être conçu d'une manière optimisée. C'est pour ces raisons que nous avons choisies une architecture basée sur les micro-services.

Finalement, l'expérience utilisateur doit être optimale et très facile à maitriser. On parle de User Experience (UX) car notre système est plutôt complexe.

2.2.3 Contraintes techniques

La liste suivante décrit quelques contraintes techniques (non fonctionnelles) que nous devons prendre en considération pendant le développement de la solution.

— La solution proposée doit assurer un maximum de disponibilité et de portabilité aux utilisateurs c'est pour cela nous avons opté pour une solution web (portail web).

— L'interface de l'application qui doit être simple et ergonomique.

— Le système doit assurer la maintenabilité : Techniquement parlant, la maintenabilité vise les deux aspects : maintenabilité et extensibilité.

— 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.

— Prendre en considération la sécurité du système dès le début de la conception.

2.2.4 Besoins fonctionnels

La solution permet la :

— Gestion des laboratoires : Ajouter, supprimer, modifier des laboratoires distants.

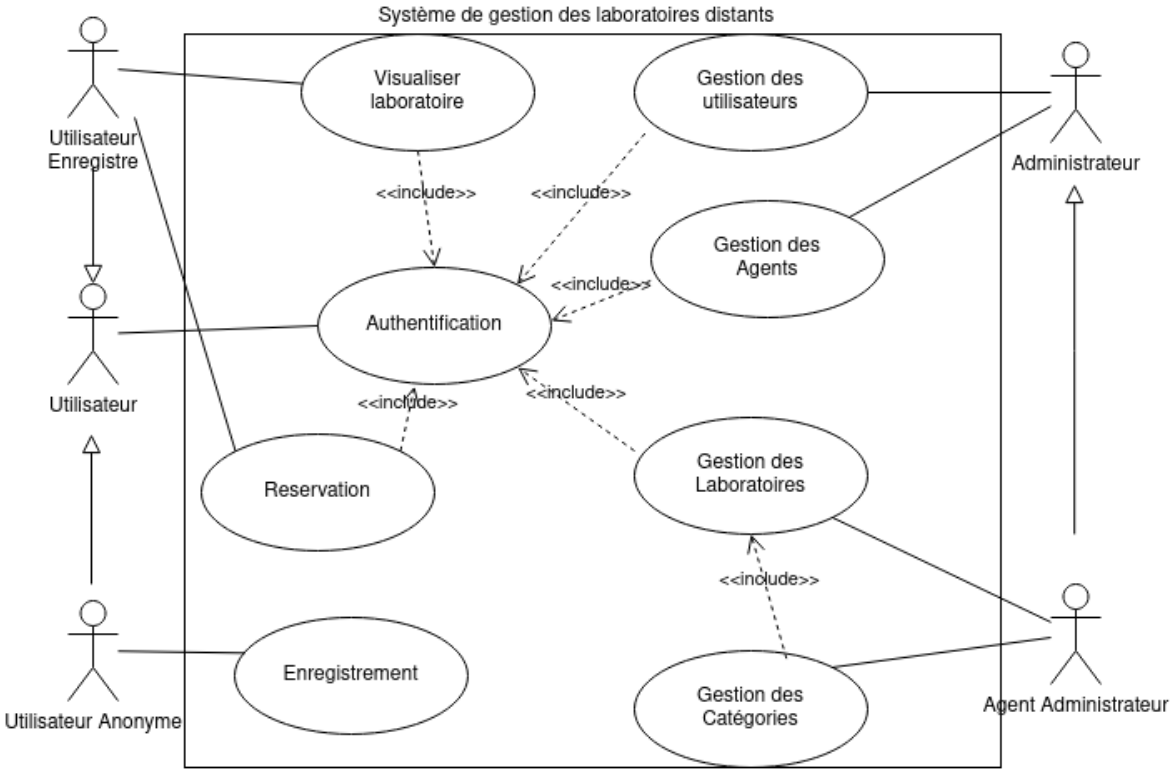


FIGURE 2.1 – Diagramme de cas d'utilisation de MostaLab

— Gestion des utilisateurs des laboratoires : Inscrire, modifier, supprimer des nouveaux profiles des utilisateurs.
— Gestion des catégories et instances des laboratoires : Regrouper les laboratoires en catégories (dans le cas où il existe plusieurs instances du même laboratoires).
— Gestion des réservations des laboratoires : l'utilisateur accède aux instances des laboratoires.

2.2.5 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.

Étude des cas d'utilisations

Le diagramme Unified Modeling Language (UML) de cas d'utilisation de la figure 2.1 montre les relations entre les acteurs du système et les différents cas d'utilisations.

Dans [COT10], le terme cas d'utilisation en anglais *use case* 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. Donc, c'est un service rendu par le système, sans imposer le mode de réalisation de ce service.

La figure 2.1 représente le diagramme de cas d'utilisation des principales tache du système des gestion des laboratoires.

- L'administrateur a comme rôle principal la gestion de toutes les taches de la plateforme, (Gestion des laboratoires, agents administratifs et les utilisateurs)
- L'administrateur attribue les laboratoires déjà créés à 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.
  - Ajouter, supprimer ou modifier des profils d'utilisateurs.
- L'agent administratif peut consulter, modifier des laboratoires qui luis ont été attribués par l'administrateur.
- L'utilisateur peut
  - Inscrire, modifier ou supprimer son profil.

— Consulter la liste des laboratoires.

— Réserver une session d'un laboratoire.

2.3 Présentation de la solution proposée

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

Lors de la phase de conception de notre solution nous avons conclu que la problématique globale liée à notre thématique (gestion des laboratoires distants) peut être résumée en deux sous problèmes (tâches) :

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

Pour la première tâche (Gestion des laboratoires), et vue que la plupart des laboratoires distants sont basés (implémentés) sur des objets connecté (IoTs); nous avons décidé d'utiliser une solution existante qui peut servir notre objectif. Pour cela et après avoir étudié plusieurs solutions open source existantes nous avons choisi *ThingsBoard* [1] , une plate-forme open source pour la gestion des IoTs.

Pour la deuxième tâche (Gestion des utilisateurs des laboratoires), nous avons développé une application web « LabsUi » notre propre solution. Une plate-forme (portail) web avec tout un système de gestion d'utilisateurs basique (inscription, modification profile, ...), avec la possibilité d'exposer les laboratoires de la plate-forme (Thingsboard).

La relation entre les deux plateformes a été rendue possible grâce à un troisième composant intermédiaire que nous avons développé. Il sert comme

1. ThingsBoard est une plate-forme IoT open source qui permet un développement, une gestion et une mise à l'échelle rapides de projets IoT. Source https://thingsboard.io

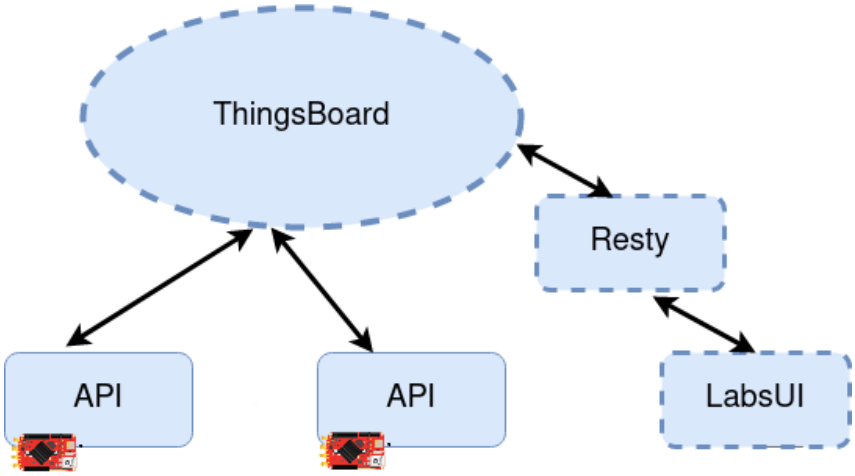


FIGURE 2.2 – Intégration de Thingsboard avec les laboratoires distants

une API REST intermédiaire entre les deux (voir figure 2.2). Nous l'avons nommé «*Resty*». Resty gère non pas la relations entre les deux premiers composants, mais aussi la relation entre le portail web (LabsUi) et les laboratoires distants dans certains cas tel que la réservation des sessions sur un laboratoire.

Nous avons opté pour une architecture orientée service API Restful pour prendre en charge la communication entre les différents composants. Ces API sont sécurisées grâce à l'utilisation d'authentification via des jetons *Json Web Token* (JWT).

Il est à noter que la solution que nous proposons ne vise que l'aspect de la gestion et publication des laboratoires distants. Cependant, il y a d'autres aspects qui doivent être gérés par les laboratoires eux-mêmes, comme la gestion des sessions qui doit être gérée par le laboratoire lui-même.

2.3.1 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

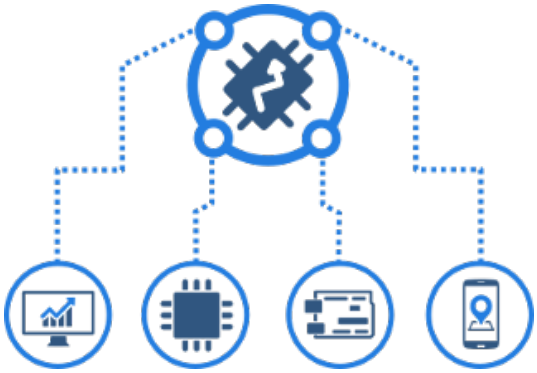


FIGURE 2.3 – Thingsboard : composants logiciels.

donner une vue globale sur cette plate-forme, pourquoi nous l'avons choisi et comment nous l'avons intégré dans notre projet.

THINGSBOARD

Selon le site officiel [Thi21], ThingsBoard est une plate-forme Internet Of Things (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 tel que : MQTT [2], CoAP [3] et HTTP et prend en charge les déploiements sur le cloud et sur site. ThingsBoard associe l'évolutivité, la tolérance aux pannes et la performance.

THINGSBOARD PRINCIPALES CARACTÉRISTIQUES ET FONCTIONNALITÉS

— Gestion de l'approvisionnement, la surveillance et le contrôle des objets connectés de manière sécurisée à l'aide de riches Application Programming Interface (API) côté serveur. Définition des relations entre les appareils, les clients ou toute autre entité.

2. MQTT (Message Queuing Telemetry Transport) est un protocole de messagerie publish-subscribe basé sur le protocole TCP/IP.Source https://mqtt.org/

3. https://coap.technology/

— Collection et visualisation des données des périphériques de manière évolutive et tolérante aux pannes. Grâce aux *widgets* [4] intégrés ou personnalisés et des tableaux de bord flexibles (voir figure 2.4). Ainsi, le partage des tableaux de bord avec les clients de la plateforme.

— Le moteur de règles (*Rule Engine*) permet de :

  — Définir les chaînes de règles de traitement de données.

  — Génération des alarmes sur les événements de télémétrie entrants,

  — Mise à jour des attributs en cas de l'inactivité des périphériques.

  — Transformer et normaliser les données des appareils.

— Gestion des alarmes liées aux appareils, clients actifs...etc. La surveillance des alarmes en temps réel et la propagation d'alarmes vers la hiérarchie des entités associées. Propagation des alarmes lors d'événements de déconnexion ou d'inactivité de l'appareil.

— Offrir une API administrative REST que l'on peut utiliser pour développer des modules externes (exemple de laboratoires distants), ce qui est un point très important pour l'intégration de ThingsBoard dans d'autres projets plus importants comme notre cas.

— L'évolutivité horizontale est prise en compte si le nombre de requêtes côté serveur augmente de manière linéaire. De nouveaux serveurs Thingsboard peuvent être ajoutés en mode cluster. Ceci est bénéfique pour les applications critiques sans temps d'arrêt.

— Open-source : ThingsBoard est sous licence Apache 2.0 [5], nous pouvons donc l'utiliser gratuitement.

4. Élément de base de l'interface graphique d'un logiciel. Source https://www.larousse.fr

5. https://www.apache.org/licenses/LICENSE-2.0

ThingsBoard pour la gestion des laboratoires

Après avoir vu le concept fondamental de ThingsBoard, nous allons maintenant discuter comment nous avons l'intégré dans notre projet pour la partie "Gestion des laboratoires".

Lors de la phase d'intégration de ThingsBoard pour la gestion des laboratoires distants , nous n'avons pas modifié Thingsboard. Nous avons seulement modifier les règles (ou bien supposition) qui doivent être respecté. Nous pouvons citer ces règles comme suit :

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

— R1 : Sous ThingsBoard, chaque laboratoire est représenté comme un périphérique (Device).

— R2 : Chaque périphérique représentant un laboratoire doit être du type "LAB".

— R3 : Chaque laboratoire est identifié par son ID ThingsBoard.

— R4 : Chaque laboratoire a un type de laboratoire qui est défini par un attribut côté client nommé "labTypeId" qui doit être publié par le laboratoire 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 *Json Web Token* (JWT) de demande de session comme un attribut côté client nommé "sessionCreationToken".

— R8 : Chaque laboratoire doit maintenir son état d'activité (active ou non) sous ThingsBoard.

— R9 : A fin d'atteindre l'objectif de R6, un laboratoire doit envoyer des mise-à-jour d'attribut côté client nommé "status" périodiquement chaque minute.

La deuxième partie concerne la représentation des unités administratives 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 administratives des fournisseurs (par exemple des départements d'une université) sont représentés comme des assets ThingsBoard.

— R12 : Un agent administratif est représenté comme un utilisateur (User) ThingsBoard.

Les relations entre ces unités sont représentées sous ThingsBoard en utilisons les relations Thingsboard. Aussi, la plus parts des règles citées précédemment sont des propositions plus que des règles.

Nous avons cité seulement les règles qui gèrent les manipulations sous Thingsboard et les interactions entre les laboratoires et le serveur ThingsBoard. Les autres règles qui gèrent d'autres aspects seront cités plus-tard.

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

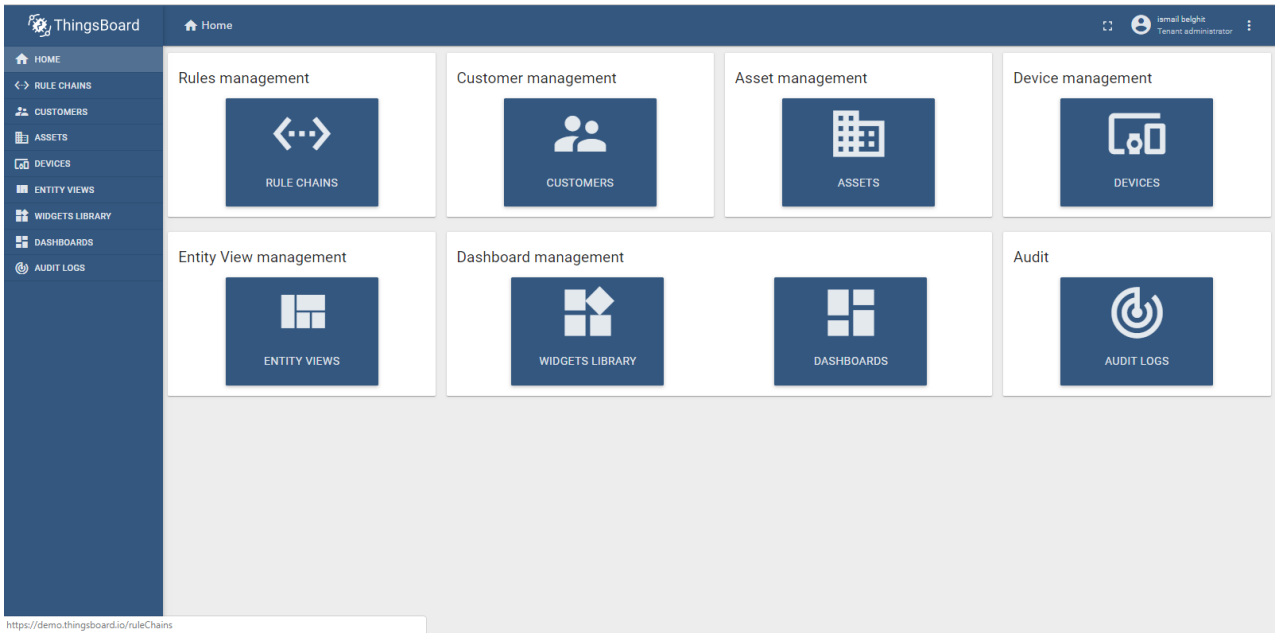


FIGURE 2.4 – Tableau de bord principal sous ThingsBoard.

2.4 ARCHITECTURE MATÉRIELLE DE MOSTALAB

Les travaux pratiques scientifiques sont l'un des outils fondamentaux permettant aux étudiants à la fois d'acquérir des connaissances pratiques et de faire de la recherche scientifique. Cependant, faire les travaux pratiques en classe comporte plusieurs défis : Les enseignants ont du mal à trouver du temps, de l'espace et du matériel en fonction du nombre d'élèves. Ils sont également préoccupés par les problèmes de sécurité des étudiants et font face à l'inexpérience des étudiants, car ils doivent se familiariser avec des instruments souvent très coûteux.

La plupart des expériences en électronique passent par trois étapes : Premièrement, les étudiants ont besoin de certains instruments (générateur de fonctions, multimètre, oscilloscope...) et de certains composants comme (résistances, condensateurs, diodes, amplificateurs, etc.). Deuxièmement,

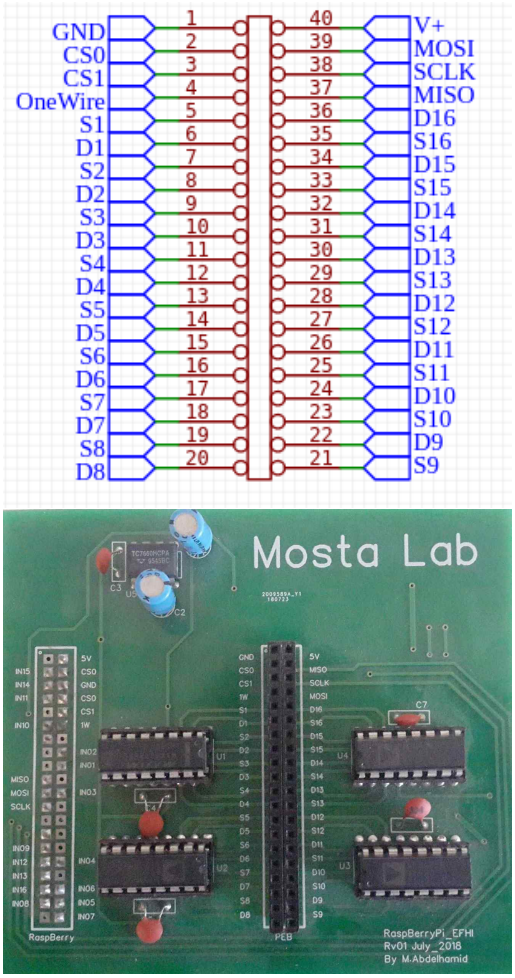


FIGURE 2.5 – Haut : Schéma de branchement coté carte de travaux pratique
En bas : Carte de commutation

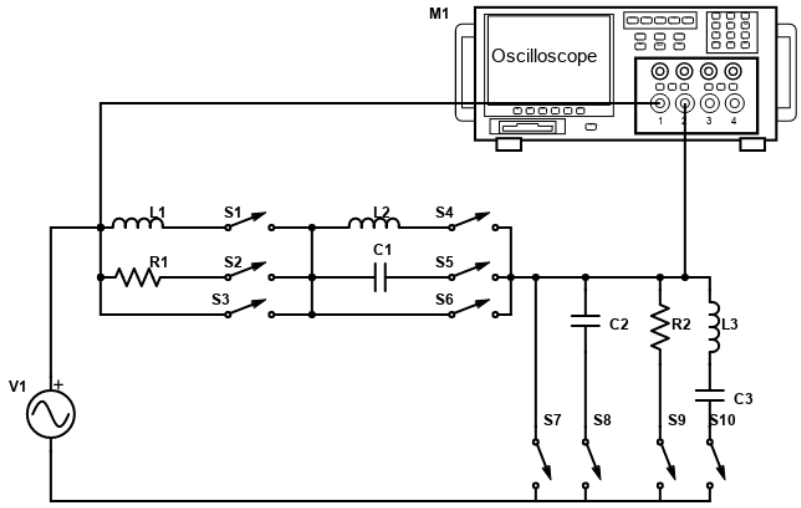


FIGURE 2.6 – Exemple de laboratoire filtres passifs configurables

ils doivent faire des assemblages, visualiser des signaux, mesurer des grandeurs électriques. Enfin, les étudiants rédigeront le rapport de l'expérience pour leur évaluation.

L'architecture matérielle du laboratoire distant doit être flexible pour prendre en charge plusieurs scénarios possibles/configurations (voir la figure 2.6). Elle doit également être réutilisable pour pouvoir s'adapter à différentes catégories de travaux pratiques : (électronique analogique, électronique numérique, électronique de puissance, etc.)

2.4.1 La carte de commutation

La carte de commutation (voir figure 2.5) est une pièce importante dans la conception matérielle de MostaLab de par sa réutilisation, elle permet d'assurer toutes les configurations possibles que le concepteur de l'expérience prévoit pour les consignes des travaux pratiques. Elle se compose de 16 commutateurs numériques contrôlés par les sorties GPIO du RaspberryPi. Chaque expérience a une configuration d'état de commutateur très

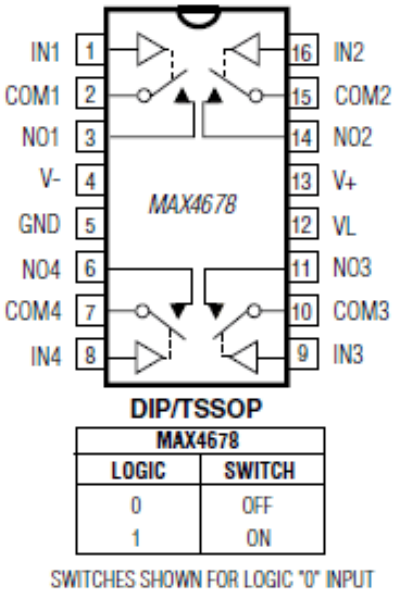


FIGURE 2.7 – MAX4678

spécifique qui est appliquée directement à la carte de TP en ouvrant ou en fermant ces commutateurs.

Pour la conception de la carte de commutation, quatre circuits de type MAX4678 ont été utilisés. Cela garantit un fonctionnement silencieux et un temps de réponse de commutation nettement court par rapport aux switchs mécaniques.

Le MAX4678 est un circuit analogique quadruple de type CMOS *Single Pole Single Throw* (SPST) dans un boîtier DIP à 16 broches qui peut transmettre des signaux numériques et analogiques (voir figure 2.7).

2.4.2 La carte de TP

La carte de TP (exemple sur la figure 2.9) se connecte à la carte de commutation via une nappe standard à 40 lignes. C'est cette carte de TP qui diffère d'une expérience à l'autre. L'intérêt de cette solution est que le changement se fait par simple opération de déconnexion/connexion de la nappe.

FIGURE 2.8 – Exemple d'un déploiement de MostaLab

Cette carte de TP est un support pour les composants TP (résistances, condensateurs, inductances, diodes, etc.) qui sont connectés directement aux interrupteurs de la carte de commutation. Cependant, les potentiomètres numériques et la mémoire *OneWire* sont liés au contrôleur car ils sont contrôlés via un protocole de communication spécifique (SPI pour le potentiomètre numérique et le composant *OneWire*). Le brochage se fait selon le schéma qui est donné dans la figure 2.10

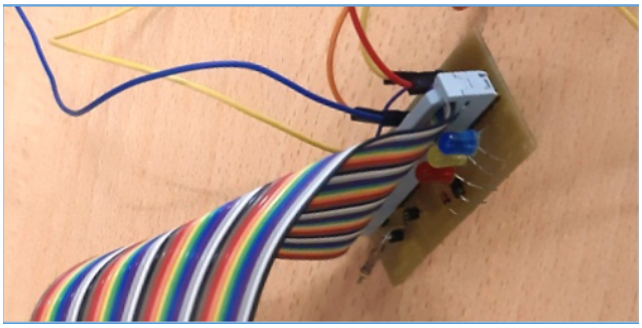


FIGURE 2.9 – Carte de TP

2.5 ARCHITECTURE LOGICIELLE DE MOSTALAB

2.5.1 CONCEPTION ORIENTÉE SERVICES

D'un point de vue architecturale ou conceptuelle, n'importe quel système informatique est implémenté (architecturé) d'une des deux manières suivantes :

1. L'approche classique (monolithique) qui regroupe et implémente toutes les fonctionnalités (logiques) du système en une seule unité logique.
2. L'approche moderne qui décompose le système en plusieurs unités ou sous-systèmes coopératifs appelés aussi des micro-services. En général, cette décomposition est par rapport aux différentes fonctionnalités du système ou chaque fonctionnalité est implémentée comme un service indépendant.

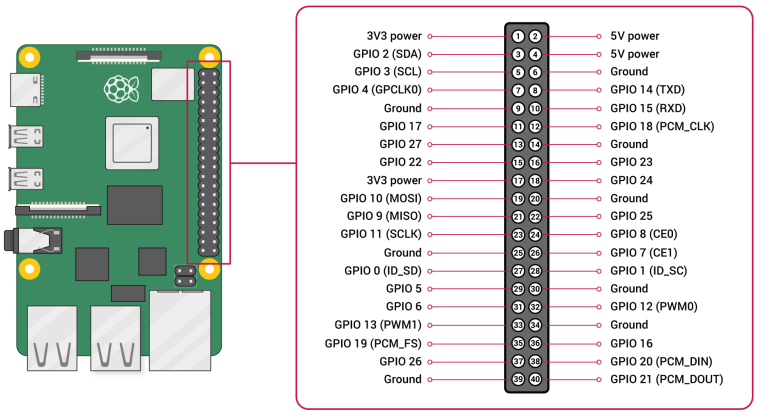


FIGURE 2.10 – Raspberry Pi GPIO [PiO21]

2.5.2 LES MICROSERVICES DE MOSTALAB

L'architecture logicielle MostaLab illustrée en figure 2.11 est une extension d'un précédent travail décrit dans [Ben+19]. Le cœur de cette architecture, comme le montre la figure 2.11, repose sur plusieurs services Web. L'architecture est modulaire pour permettre la réutilisation des services dans une variété de scénarios et de laboratoires. Cette architecture est basée sur des technologies et des standards open source tels que RESTful et JSON. Étant donné que les services Web promettent une meilleure isolation, cela n'oblige pas les développeurs à utiliser le même langage de programmation pour différentes interfaces de programmation d'applications (API).

API GATEWAY

L'API Gateway est le composant intermédiaire entre l'interface de l'utilisateur et les services internes. Ce microservice est déployé sur un ordinateur monocarte tel qu'un Raspberry Pi. La passerelle API est chargée de

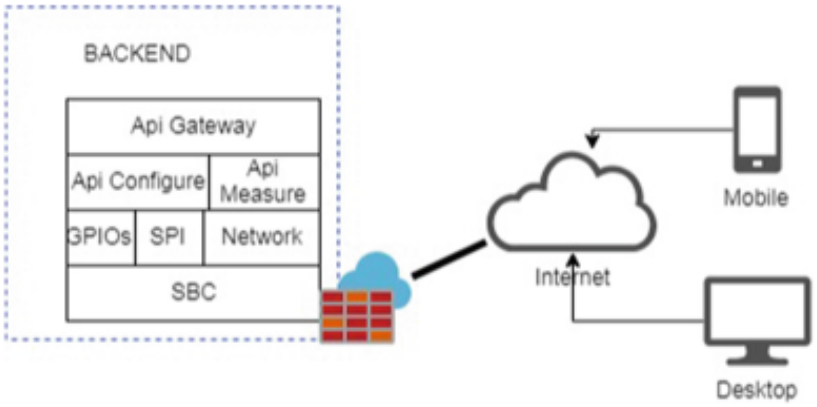


FIGURE 2.11 – Architecture orientée microservices de MostaLab

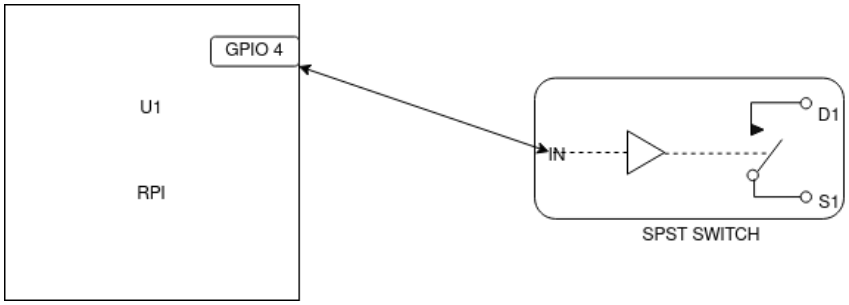


FIGURE 2.12 – Branchement Raspberry Pi avec un switch SPST

fournir au client la description complète sous format JSON de l'expérience pratique.

MICROSERVICE DE CONFIGURATION DES SWITCHS

À la réception d'une demande REST GET du client, ce microservice interagit avec les *General Purpose Input/Output* (GPIO) du *Single Board Computer* (SBC). Il permet de sélectionner une configuration particulière du circuit en agissant sur les switchs (voir figure 2.12) et en réglant les valeurs des potentiomètres numériques si le circuit en contient.

Le code du listing 1 est un exemple sous NodeJs. la fonction wpi.setup doit être appelé avant toute autres fonctions, cela ouvre les périphériques

GPIO et permet à notre programme d'acceder. Ensuite, la fonction wpi.pinMode définit le mode de la broche (pin) qui est dans notre cas numéro 4 en mode sortie. La dernière partie du code est la fonction setInterval (délais) qui s'exécute chaque 500 ms afin de faire clignoter la LED.

```
var wpi = require('wiring-pi');
wpi.setup('wpi');
var pin = 4;
wpi.pinMode(pin, wpi.OUTPUT);
var value = 1;
setInterval(function() {
  wpi.digitalWrite(pin, value);
  value = +!value;
}, 500);
```

Listing 1 – Wiring PI sous NodeJs

Microservice de Mesure

Le microservice API Measure contrôle les équipements de mise en forme du signal tels que l'alimentation numérique et le générateur de signaux arbitraires, et renvoie les mesures d'instruments tels que le multimètre numérique et l'oscilloscope.

La programmation des instruments de mesure est un véritable défi. En effet, Il existe de nombreux protocoles différents, envoyés via de nombreuses interfaces et systèmes de bus différents (par exemple GPIB, RS232, USB, Ethernet). Pour chaque langage de programmation que nous souhaitons utiliser, nous devons trouver des bibliothèques prenant en charge à la fois l'instrument et son système de bus.

Pour faire face à cette situation, la spécification Virtual Instrument Software Architecture (VISA) [6] a été définie au milieu des années 90. VISA est une norme pour la configuration, la programmation et la maintenance des systèmes d'instrumentation comprenant des interfaces GPIB, VXI, PXI, série, Ethernet et/ou USB.

```
>>> import pyvisa
>>> rm = pyvisa.ResourceManager()
>>> rm.list_resources()
('ASRL1::INSTR', 'ASRL2::INSTR', 'GPIB0::12::INSTR')
>>> inst = rm.open_resource('TCPIP::192.168.1.10::inst0::INSTR')
>>> print(inst.query("*IDN?"))
```

Listing 2 – python pyvisa lib

Aujourd'hui, VISA est implémenté sur tous les systèmes d'exploitation importants. Quelques fournisseurs proposent des bibliothèques VISA, en partie avec téléchargement gratuit. Ces bibliothèques fonctionnent avec des périphériques arbitraires, bien qu'elles puissent être limitées à certains périphériques d'interface, tels que la carte GPIB de l'instrument.

Il est facile avec le langage Python d'accéder directement à la plupart des systèmes de bus utilisés par les instruments, c'est pourquoi on peut envisager d'implémenter la norme VISA directement en Python. PyVISA [7] est à la fois un *wrapper* Python pour les bibliothèques partagées VISA, mais peut également servir de frontal pour d'autres implémentations VISA.

PyVISA est un package python qui permet de contrôler toutes sortes d'appareils de mesure indépendamment de l'interface (par exemple GPIB,

6. Virtual Instrument Software Architecture, appelé généralement VISA, est une interface de programmation largement utilisée dans le domaine de l'instrumentation et l'industrie du test et de la mesure. Source : https://www.ivifoundation.org

7. https ://github.com/pyvisa/pyvisa

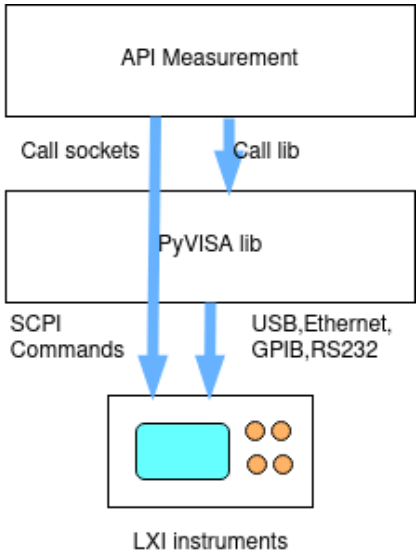


FIGURE 2.13 – Les couches PyVISA

RS232, USB, Ethernet). Par exemple sur la figure 2.13, la lecture de l'auto-identification à partir d'un instrument avec l'adresse IP 192.168.1.10 se fait selon le code du listing 2

2.6 La gestion des accès simultanés

2.6.1 Modélisation

La figure 2.14 montre l'architecture de déploiement de MostaLab. Les services formant cette architecture peuvent être déployés dans différents endroits. La conception de l'architecture est basée sur une approche orientée services utilisant le modèle *Labs-as-a-Service* (LaaS) introduit dans les travaux ([Taw+14], [Cam+14],[Zut+16], [Sal+19]). Ces services sont faiblement couplés et offrent un haut niveau d'abstraction et de virtualisation.

L'architecture fonctionnelle présentée dans la figure 2.14 repose sur trois composants :

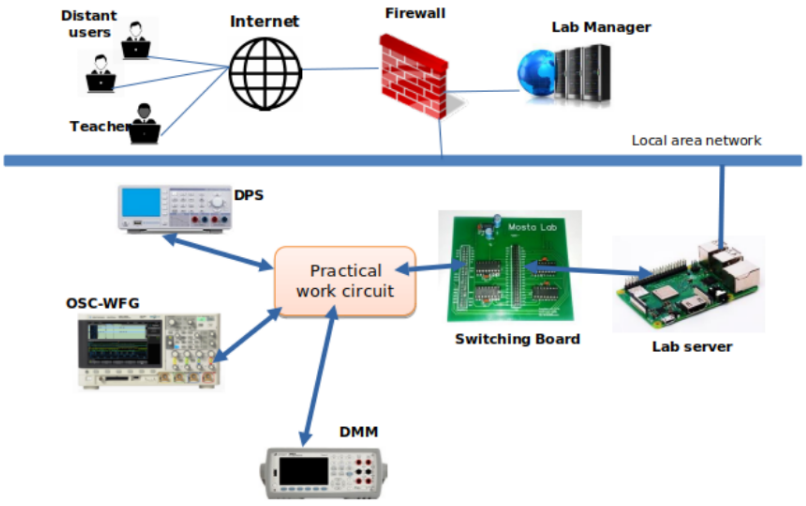


FIGURE 2.14 – Architecture de déploiement de MostaLab

— Le gestionnaire du laboratoire (*Lab Manager*)

— Le serveur du laboratoire (*Lab server*)

— L'interface utilisateur (*User Interface*)

Dans la section suivante, nous décrivons en détail l'interaction entre les différents composants.

Le diagramme de séquence de la figure 2.15 illustre l'interaction entre les différents services de l'application. L'objectif est double : 1) orchestrer les différents services web ; 2) Il permet aux gestionnaires de laboratoire distant d'avoir des données de temps de chargement des applications pour la surveillance des performances du système et d'avoir des données fiables pour l'analyse des performances futures.

1. L'étudiant envoie une demande d'accès au serveur du Lab ; la requête contient : id_lab, t_0 (heure de début de la transaction), session_id et les données pour le laboratoire (mode direct/indirect, tension d'entrée).

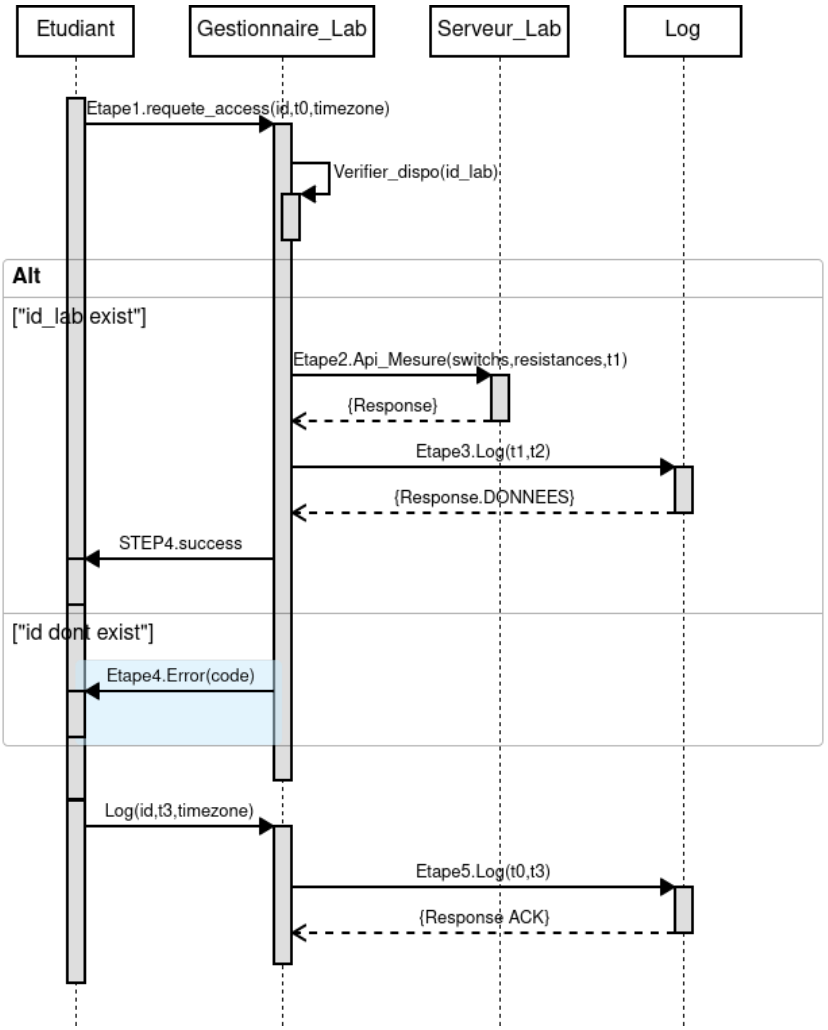


FIGURE 2.15 – Diagramme de séquence UML du partage des laboratoires distants

2. Le gestionnaire du laboratoire note l'heure d'arrivée t_1 de la demande, vérifie l'autorisation de l'utilisateur et envoie la demande au serveur du laboratoire.
3. Le serveur du laboratoire configure le circuit avec les paramètres de la requête et renvoie les résultats des mesures au gestionnaire du laboratoire. Le gestionnaire du laboratoire note le temps de réponse t_2, calcule le temps nécessaire pour effectuer la mesure ($t_2 - t_1$) et enregistre les paramètres de la requête, les différents temps dans un fichier journal.
4. Le gestionnaire du laboratoire renvoie les mesures demandées au client ou un code d'erreur si la demande a échoué.
5. Enfin, le client note le temps de réponse t_3, affiche les mesures aux endroits appropriés. Il calcule le temps global de la transaction ($t_3 - t_0$) et le renvoie au gestionnaire du laboratoire qui l'enregistre dans le fichier journal.

Une remarque importante est que dans ce type de flux, l'horloge du client et celle du gestionnaire du laboratoire ne doivent pas être forcément synchronisées. Les instants t_0 et t_3 ont la même référence, ainsi que t_2 et t_4

2.6.2 Algorithme de gestion de la concurrence

Nous choisissons le laboratoire de caractérisation de plusieurs diodes pour expliquer le mécanisme de gestion de la concurrence. L'objectif de ce laboratoire permet aux étudiants l'étude statique du dipôle.

Caractériser un dipôle en mode statique, c'est étudier la fonction appelée I (V). C'est-à-dire on impose une tension continue V aux bornes de la diode, et on mesure la valeur du courant qui la traverse et la tension aux bornes de la diode.

Le schéma de la figure 2.16 montre le câblage du laboratoire de diodes. Les switchs dans ce schéma se classes comme suit :

— S8 et S10 on pour la polarisation direct.

— S7 et S9 on pour la polarisation en inverse.

— S1 à S6 pour sélectionner la diode (Silicium, Germanium, Led rouge, ...etc)

Le serveur du Laboratoire effectue la séquence suivante :

— Sélection de la diode et du sens avec les switchs

— Commande SCPI pour mesurer la tension V et son affichage.

— Commande SCPI pour mesurer le courant I et son affichage.

```
1   Create_PS_Socket() // Générateur de tension
2     Execute_ parallel(Configure_circuit, Configure_Pot)
3     Create_DMM_Socket()
4       Read_Volt()
5     Close_DMM_Socket()
6     Configure_Switch_step2
7     Create_DMM_Socket()
8       Read_Current()
9     Close_DMM_Socket()
10  Close_PS_Socket() //LOG time response
```

Listing 3 – Algorithme gestion des access

L'algorithme du listing 3 illustre la gestion des accès et le principe de la reconfiguration d'un même instrument en plusieurs modes. Le multimètre numérique passe du mode voltmètre au mode ampèremètre pour obtenir des mesures du même circuit. L'accès exclusif aux instruments est

fourni par le serveur du laboratoire. Le mécanisme de *socket* est choisi, car il est très adapté pour éviter que deux requêtes accèdent au même temps à l'instrument. Les opérations de configuration des switchs et les potentiomètres numériques peuvent être exécutées par le serveur du laboratoire en parallèle. Pour assurer l'isolement des requêtes entre utilisateurs, l'exclusion mutuelle est mise en place.

Create_PS_Socket() est une fonction qui garantit qu'au plus une requête à tout moment peut accéder à la section critique (lignes 1 à 10). Ce n'est qu'à la fin de cette section critique que d'autres requêtes peuvent accéder à la configuration matérielle. Cela évite les conflits lorsque deux ou plusieurs requêtes accèdent aux instruments (deux étudiants). La partie critique est chargée de plusieurs tâches (configuration des switchs en mode voltmètre, la mesure des signaux et enfin la reconfiguration des switchs en mode ampèremètre et mesure de courant).

Dans notre implémentation, l'interface semble être un laboratoire synchrone et interactive, mais lorsque les étudiants changent les valeurs des résistances, ils ne sont en fait pas connectés au serveur. C'est une fois que les étudiants soumettent le circuit que la tâche est exécutée sur le serveur de laboratoire - sans être modifiée par l'utilisateur - et ensuite le résultat est renvoyé. Donc techniquement, c'est un laboratoire asynchrone. Cependant, le cycle et la perception de l'utilisateur sont plus proches d'un laboratoire synchrone. Car, les temps de réponses sont très courts et donnent la sensation d'être synchrone.

2.7 Conclusion

Dans ce chapitre, nous avons présenté les différentes étapes de modélisation de MostaLab. Ainsi, nous avons montré comment nous pouvons

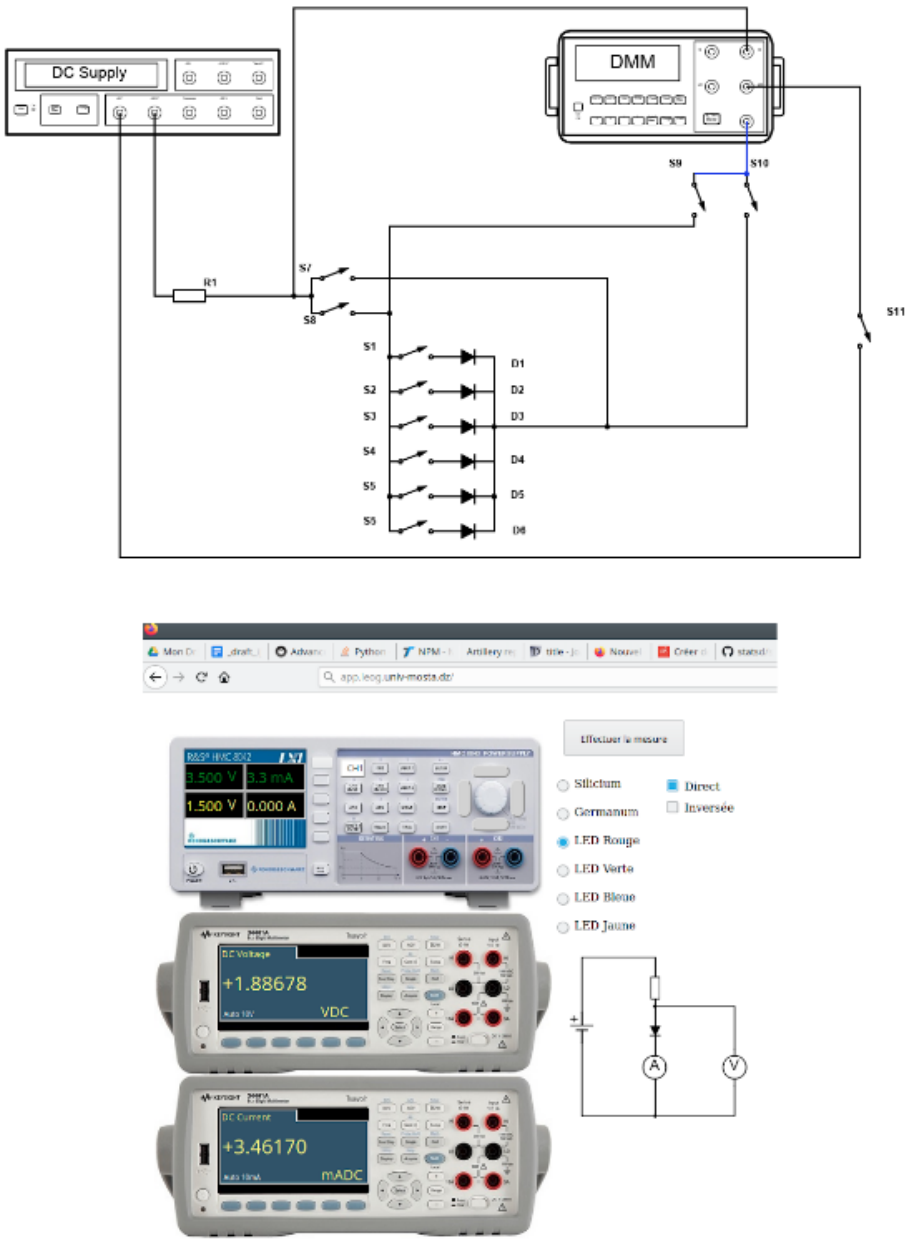


FIGURE 2.16 – Configuration TP caractérisation d'une diode

concevoir un gestionnaire des laboratoires distants en utilisant une approche par microservices.

Chapitre3

Évaluation de la performance de MostaLab

"Everything is theoretically impossible, until it is done." – Robert A. Heinlein

3.1 Introduction

Ce chapitre montre à quel point le modèle de partage proposé dans cette thèse peut atteindre sa performance maximale et en combien de temps. Par conséquent, ce chapitre reprend l'implémentation décrite en chapitre 2 et mesure le comportement du système dans différentes conditions avec différentes charges d'utilisateurs.

Pour ce faire, une introduction aux techniques de tests de performances des applications décrite dans la section 3.2. la section 3.3 détaille l'outil artillery que nous avons choisi pour bien mener cette étude . Alors que le temps de réponse augmente si nous ajoutons des étudiants. Nous allons dans la section 3.4 tester la performance de MostaLab pour répondre à la question : Combien d'étudiant peut-on-inscrire pour une seule instance de laboratoire si on se donne comme contrainte 5 s comme étant le temps de réponse maximale acceptable.

3.2 Les tests de performances des services Web

Les tests de performances sont un type de test destiné à déterminer la réactivité, la fiabilité, le débit, l'interopérabilité et l'évolutivité d'un système ou un service web sous une charge de travail donnée. Il peut également être défini comme un processus de détermination de la vitesse ou de l'efficacité d'un ordinateur, d'un réseau, d'une application logicielle ou d'un appareil. Les tests peuvent être effectués sur des applications logicielles, des ressources système, des composants d'application ciblés, des bases de données et bien plus encore. Cela implique une suite de tests automatisés, car

cela permet des simulations faciles et reproductibles d'une variété de conditions de charge normales, de pointe et exceptionnelles.[Eri17]

Les tests de performance peuvent également être un outil de diagnostic pour déterminer les goulots d'étranglement et les points de défaillance possibles.

Avant de commencer à mesurer efficacement les performances Web, il est très important d'expliquer l'architecture orientée service donnée en figure 3.1. Cette architecture comprend :

— Un navigateur Web en tant que client logiciel où s'exécute une application souvent écrite en javascript ;

— Fournisseurs de services Internet ;

— Les serveurs web qui sont des applications capables de répondre aux demandes du client (navigateur). Par la suite les requêtes sont transmises à une application web qui peut être sur la même machine ou sur un autre serveur,

— le serveur d'applications est un endroit où s'exécute le code de l'application. Les requêtes viennent des serveurs web où sont traitées et apportent des réponses,

— le *back-end* [1] du laboratoire distant, souvent on trouve le *Single Board Computer* (SBC) Raspberry Pi.

3.2.1 Techniques de test de performance

Dans cette section, nous présentons les différentes techniques largement utilisées pour tester la performance des applications Web :

— Test de charge

1. En informatique, un back-end est un terme désignant une couche de sortie d'un logiciel devant produire un résultat

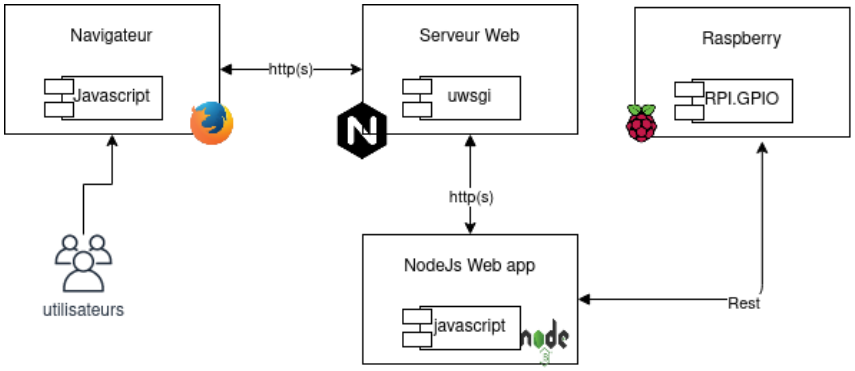


FIGURE 3.1 – Architecture Web orientée micro-services

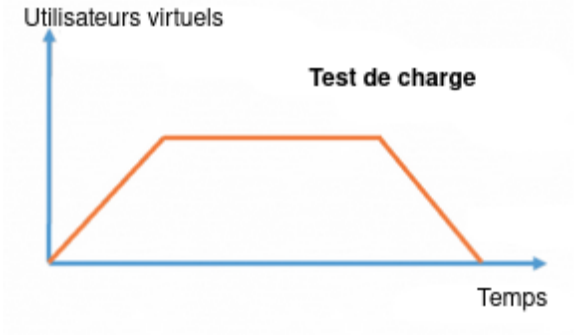
— Tests de résistance

TEST DE CHARGE

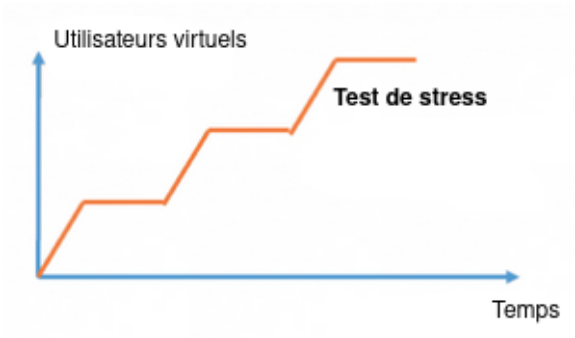
Comme le montre la figure 3.2a, le test de charge est une technique très répandue pour mesurer les performances des sites Web. Cette méthode permet de déterminer le comportement de l'application Web dans des conditions de charge très élevées et anticipées. Le processus pour réaliser ces tests est basé sur une augmentation progressive des utilisateurs. Cela signifie qu'un test démarre généralement avec un petit nombre d'utilisateurs virtuels et progressivement on augmente ce nombre d'utilisateurs pour atteindre la capacité maximale du système. En utilisant ce processus, il est possible d'observer les performances de l'application au cours de ce test de charge progressivement. Par exemple, si le but du test est de trouver le point où le temps de réponse est inférieur à 5s, les requêtes sont incrémentées tant que point n'a pas été franchi.

TESTS DE RÉSISTANCE

Le test de résistance (voir figure 3.2b) est une sorte de test où l'environnement testé est exposé à des conditions extrêmes. Ce test est effectué pour déterminer la fiabilité, la robustesse, la disponibilité de l'application



(a) Test de charge



(b) Test de résistance

FIGURE 3.2 – Catégories de test de performance

et identifier les problèmes d'application qui surviennent dans des conditions extrêmes. Il peut inclure des charges lourdes, une simultanéité élevée et des ressources matérielles. Un test de stress approprié est utile pour trouver des erreurs de synchronisation, des problèmes de verrouillage, des problèmes de priorité et des bugs de perte de ressources. L'objectif principal de l'exercice du test de stress jusqu'au point de charge maximale est de trouver des problèmes qui peuvent être potentiellement dangereux. Le stress test implique généralement un ou plusieurs scénarios de production qui sont exécutés dans diverses conditions de stress. Par exemple, exécuter une application sur une machine où une autre application consomme une quantité considérable de ressources.

3.2.2 Identification des paramètres

Dans cette section, nous dressons une liste des paramètres qui peuvent avoir un effet sur les performances d'un laboratoire distant. Ces paramètres doivent être pris en compte lors de la conception du système de test de charge.

— L'arrivée moyenne des requêtes par seconde : L'arrivée moyenne des requêtes est un paramètre d'entrée d'un système de test de charge. Il exprime le nombre de requêtes par unité de temps (typiquement par seconde). Nous augmentons ce paramètre de manière contrôlée pour étudier la réponse du système et déterminer s'il est apte à prendre en charge des scénarios avec les caractéristiques de charge étudiées. Une augmentation importante de l'arrivée moyenne des requêtes peut entraîner une augmentation du temps de réponse de la requête et peut surcharger le système au point où il tombera en panne et cessera de fonctionner.

— Distribution des arrivées : si l'arrivée moyenne des demandes décrit la quantité des requêtes par unité de temps, elle ne précise pas la manière dont les requêtes sont généralement distribuées au cours de cette période. La même quantité de requêtes entrantes pourrait, par exemple, être également espacée dans la période de temps ou se produire principalement autour d'un décalage spécifique dans chaque période.

3.2.3 Indicateurs clés des tests de performances

Nous analysons chaque métrique dans ce qui suit :

Moyenne

Le temps de réponse moyen est couramment utilisé dans le test de performance des applications Web. Nous supposons que cela représente une transaction «normale», cependant, cela ne serait vrai que si le temps de réponse est toujours le même (toutes les transactions s'exécutent à la même vitesse) ou si la distribution du temps de réponse est à peu près en cloche.

Pour calculer la moyenne, nous additionnons simplement toutes les valeurs d'échantillon, puis nous divisons ce nombre par le nombre d'échantillons (voir équation 3.1).

$$\frac{1}{n}\sum_{i=1}^{n} x_i \tag{3.1}$$

Ainsi, lors de l'analyse des temps de réponse moyens, nous pouvons avoir un résultat qui se situe dans un niveau acceptable, mais il faut être prudent avec les conclusions que nous tirons.

Écart-type

L'écart type est une mesure de la dispersion par rapport à la moyenne, de la variation des valeurs par rapport à leur moyenne ou de leur distance. Si la valeur de l'écart type est petite, cela indique que toutes les valeurs de l'échantillon sont proches de la moyenne, mais si elle est grande, elles sont alors éloignées et ont une plage plus large.

$$\sigma = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - \overline{x})^2} \tag{3.2}$$

Centiles

Un centile est une métrique de test de performance très utile qui donne une mesure sous laquelle se trouve un pourcentage de l'échantillon. Par exemple, le 90e centile (abrégé en p90) indique que 90 % de l'échantillon est en dessous de cette valeur et que le reste des valeurs (c'est-à-dire les 10 % restants) est au-dessus.

Il est pratique d'analyser plus d'une valeur centile, montrant combien d'échantillons sont p90, p95 et p99. Si ceux-ci sont complétés par le minimum, le maximum et la moyenne, alors il est possible d'avoir une vue beaucoup plus complète des données et ainsi de comprendre comment le système s'est comporté.

Certains centiles ont des noms spéciaux, tels que p100 qui est le maximum (100 % des données sont en dessous de cette valeur) et p50, qui est la médiane (la moitié des données est en dessous et l'autre au-dessus).

Le centile est généralement utilisé pour établir des critères d'acceptation, indiquant que 90 % de l'échantillon doit être inférieur à une certaine valeur. Ceci est fait pour éliminer les valeurs aberrantes. Ainsi, nous notons que pour analyser les résultats de notre test de performance, il faut garder ces considérations clés :

— La moyenne ne doit pas être considérée comme une valeur à surveiller, car elle cache souvent des informations importantes.

— l'écart type exprime à quel point la moyenne est utile, plus l'écart type est élevé, moins il est significatif.

— Définir des critères d'acceptation en fonction du centile. Si par exemple nous sélectionnons 90e centile, si comme si nous disons que : « il n'est pas important si 10 % de mes utilisateurs ont un temps de réponse supérieur» .

3.3 Artillery.io

Artillery.io est une solution de test de charge moderne conçue pour les équipes inter-fonctionnelles exécutant des systèmes basés sur les microservices. Il permet de mesurer certains aspects de performances des systèmes. Les développeurs utilisent Artillery.io pour modéliser le comportement des entités ou des utilisateurs qui interagissent avec un système donné afin de vérifier que le système se comporte comme prévu en termes de fonctionnalités et de performances lorsqu'il est soumis à des contraintes de charge.

Le moteur Artillery.io est capable de générer une charge en utilisant divers protocoles et une logique complexe qui est personnalisable par l'utilisateur grâce à la définition des scénarios. Un plan de test décrit la série d'étapes que la solution Artillery.io exécutera pendant le test de charge.

Nous avons choisi Artillery pour exécuter les tests de charge, car il est léger, flexible et ne nécessite pas une configuration complexe. Cela nous permet également d'écrire des scénarios de test à l'aide de modèles *Yet Another Markup Language* (YAML) [2], ce qui les rend faciles à lire et, dans la plupart des cas, plus faciles à écrire (Listing 4).

3.3.1 Installation et lancement de l'outil Artillery.io

Artillery est un package NodeJs conçu pour tester les systèmes *Backend*, tels que les APIs. Prêt à l'emploi, il prend en charge les APIs HTTP standard, ainsi que des protocoles tels que WebSockets. Il s'intègre également à la bibliothèque Socket.io.

2. YAML est un format de représentation de données par sérialisation Unicode. Il reprend le format de message électronique documenté par RFC 2822. https://datatracker.ietf.org/doc/html/rfc2822

Une distinction importante à faire avec Artillery.io est qu'il n'est pas conçu pour tester des applications Web via le navigateur ou JavaScript côté client. Il est relativement simple d'installer Artillery.io dans le système d'exploitation exécutant NodeJs (version 12.0 ou supérieure). Une fois NodeJs configuré, l'installation d'Artillery se fait en une seule ligne de commande :

```
$ sudo npm install -g artillery
```

Avant de commencer à utiliser Artillery, nous devons créer un script de test. Artillery fonctionne en exécutant un simple script au format YAML ou JSON. Les scripts de test se composent de deux sections : la partie configuration du test (appelée config) et la partie de scénarios à exécuter pour chaque utilisateur virtuel. Ces sections indiqueront à l'outil Artillery où et quoi tester, combien d'utilisateurs virtuels générer et d'autres paramètres que nous allons détailler dans la section suivante.

Le code du listing 4 illustre un simple exemple de script. Ce script se connecte à un point de terminaison d'API pendant une minute, en envoyant 2 utilisateurs virtuels par seconde.

```
config:
target: "https://lab.leog.univ-mosta.dz/diode-lab"
phases:
- duration: 60
arrivalRate: 2
scenarios:
- flow:
- loop:
- post:
url: "/resource"
json:
```

```
switch: {{switch_config}}
- think: 20
count: 50
- log: "Nouveau utilisateur virtuel"
```

Listing 4 – Script Artillery sous YAML

Comme mentionné précédemment, les scripts de test Artillery se composent de deux sections principales. La première section, config, contient deux paramètres de configuration :

— target : Cela permet de spécifier l'URL de base pour toutes les requêtes définies ultérieurement dans le script de test.

— phases : Ils précisernt la durée du test, le nombre d'utilisateurs virtuels que nous souhaitons envoyer pendant le test et à quelle fréquence.

Le réglage de phase est la configuration la plus importante pour ce test, car il permet de contrôler la quantité de charge que nous souhaitons envoyer. Pour ce test, la touche de durée permet de spécifier combien de secondes Artillery effectue ce test. La deuxième clé, ArrivalRate, indique à Artillery combien d'utilisateurs virtuels nous souhaitons envoyer à l'application chaque seconde. Nous pouvons définir différents paramètres pour cette configuration, comme nous le verrons plus loin dans d'autres exemples.

Le paramètre scénarios permet de spécifier une ou plusieurs requêtes que le programme Artillery exécute pendant le test. Chaque scénario est une séquence d'étapes qui s'exécutent dans l'ordre, tel que défini par la clé de flux. La clé de flux contient chaque étape du scénario à exécuter. Puisque nous testons une API HTTP sous charge, nous utiliserons des méthodes de requête HTTP telles que GET et POST pour indiquer à Artillery.io ce qu'il faut faire tout au long du processus. Dans notre exemple, nous avons

trois scénarios (GET, POST, Think). le scénario Think permet de faire une pause.

Pour lancer la simulation nous exécutons la commande suivante :

```
$ artillery run script.yml
```

À la fin du test, Artillery.io affiche les résultats de l'ensemble du test comme le montre le listing 5

Ces résultats fournissent les informations dont nous avons besoin pour savoir comment le système se comporte. Artillery fournit le nombre de réponses moyennes par seconde, ainsi que toutes sortes de statistiques sur les temps de réponse comme le minimum, le maximum, la médiane et les centiles (P90,P95). Il affiche également les codes de réponse reçus, afin de verifier si notre API commence à échouer sous l'effet du stress.

```
All virtual users finished
Summary report @ 17:12:54(+0900) 2021-02-27
Scenarios launched:  1500
Scenarios completed: 1500
Requests completed:  1500
Mean response/sec: 24.81
Response time (msec):
min: 110.1
max: 443.3
median: 137.6
p95: 178.1
p99: 251
Scenario counts: 1500 (100%)
Codes:
200: 1500
```

Listing 5 – Résultat de la commande run de Artillery

3.3.2 Génération d'une charge

Pour générer une charge sur le laboratoire distant, il est nécessaire de bien configurer la section config du fichier YAML. La structure de la section commence par la clé target, qui définit l'url service web front de notre backend , et est suivie une liste de phases. Dans le Listings 4 La valeur de configuration target nous indique qu'Artillery se connectera à l'application exécutée sur `http://lab.leog.univ-mosta.dz/diode-lab`. En regardant la définition des phases, nous voyons que Artillery.io simulera 2 nouveaux utilisateurs arrivant pour utiliser l'application toutes les secondes pendant 60 secondes (120 utilisateurs arrivant en l'espace d'une minute).

```
config:
  phases:
   - duration: 10
      arrivalRate: 1
      name: "phase1"
   - duration: 10
      arrivalRate: 40
      rampTo: 10
     name: "Phase2"
   - duration: 20
      arrivalRate: 80
      name: "Phase3"
```

Listing 6 – Code YAML définissant les phases

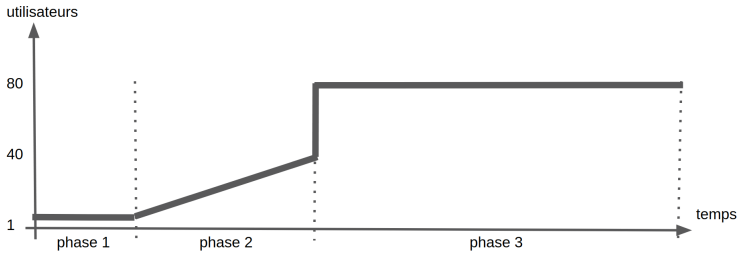


FIGURE 3.3 – Phases dans artillery.io.

L'exemple précédent ne reflète pas la réalité. Il est possible de définir plusieurs phases dans la section configuration. Les phases permettent de définir la charge souhaitée sur la cible. Chaque phase peut avoir des caractéristiques différentes pour varier l'intensité de la charge durant l'exécution du test de charge. La code du linsting 6 illustre un exemple de trois phase définis comme suit :

— phase 1 : 1 utilisateur pendant 10 secondes.
— phase 2 : 40 utilisateurs pendant 10 secondes avec une rampe de 10 utilisateurs par seconde.
— phase 3 : 80 utilisateurs pendant 20 secondes.

3.4 MostaLab : Processus de test de charge

La figure 3.4 montre le processus en deux phases que nous avons choisi pour répondre à notre problématique de recherche. Dans la première phase, un laboratoire distant a été modélisé, développé puis déployé pendant deux semaines auprès d'une centaine d'étudiants. Les résultats de cette phase nous amènent à bien caractériser un étudiant type. Dans la deuxième phase, appelée test de charge, nous avons construit un *Bot* [3] avec l'outil Artillery.

3. Un bot se connecte et interagit avec le serveur comme un programme client. Le terme « bot » est la contraction de « robot » Source : https://www.cloudflare.com/fr-fr/learning/bots/what-is-a-bot/

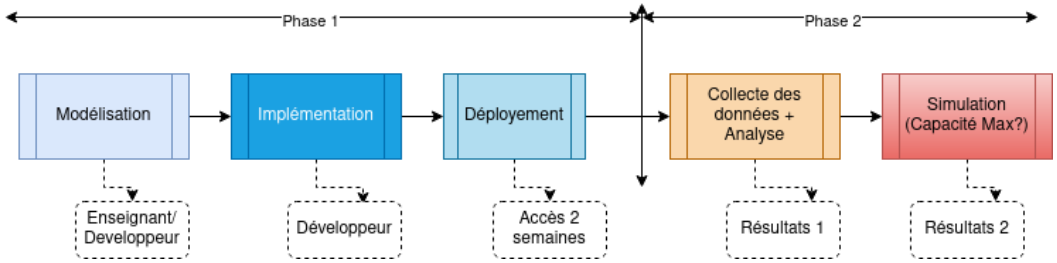


FIGURE 3.4 – MostaLab : Processus de test de charge [Mou+21]

Ce *bot* est construit avec les paramètres de l'étudiant typique, puis ce *Bot* est exécuté plusieurs fois pour déterminer la capacité maximale qu'une seule instance de laboratoire peut atteindre.

Les activités modélisation et implémentation ont été déjà développés en chapitre 2. Dans la section suivante, nous étudierons l'activité déploiement et les activités de la phase 2.

3.4.1 DÉPLOIEMENT

Un cas d'utilisation (Caractérisation d'une diode) a été déployé et testé pendant deux semaines au cours de l'année universitaire 2018/2019 à la Faculté des Sciences et Technologies de l'Université de Mostaganem. Les étudiants ont été invités à prendre plusieurs mesures, relever les caractéristiques I(V) de la diode, rédiger et soumettre le rapport sur la plateforme Moodle. Le laboratoire était disponible 24 heures sur 24 pour une classe de 109 étudiants.

Pendant les deux semaines, les étudiants ont d'abord été invités à accéder à la plateforme Moodle pour utiliser l'expérience à partir du lien donné dans le cours. Sur l'interface utilisateur (voir figure 3.5) de Moodle, les instructions données aux étudiants se font selon les étapes suivantes :

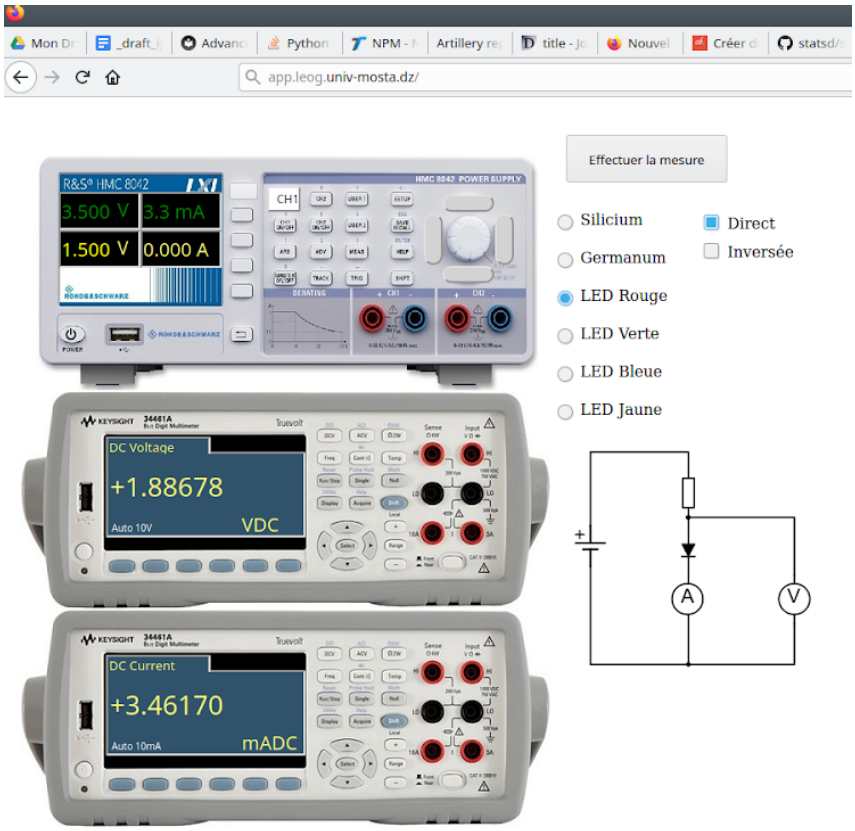


FIGURE 3.5 – Interface utilisateur du laboratoire distant caractérisation de plusieurs diodes

1. L'étudiant configure le mode direct de connexion directe et un type de diode et il applique une tension d'entrée continue en entrant la valeur dans le générateur de tension sur l'interface utilisateur.
2. L'étudiant clique sur le bouton "Effectuer la mesure" et attend l'affichage des valeurs du courant et de la tension sur les instruments.
3. L'étudiant enregistre ensuite les valeurs recueillies sur une feuille de son tableur.
4. L'étudiant répète plusieurs fois les étapes précédentes pour pouvoir tracer la courbe I (v)
5. l'etudiant rédige un rapport et il le télé-verse dans l'activité devoir de la plateforme Moodle.

3.4.2 Résultat et discussion de la phase 1

```
1 cbs4,"41.96.155.224",2018/11/22,22:36,v=3.000,direct,Led
  ↪ Verte,+1.92281,VDC,+2.29523,mADC
2 cbs4,"41.96.155.224",2018/11/22,22:37,v=3.500,direct,Led
  ↪ Verte,+1.94311,VDC,+3.31360,mADC
3 cbs4,"41.96.155.224",2018/11/22,22:37,v=3.500,direct,Led
  ↪ Verte,+1.94290,VDC,+3.31421,mADC
4 cbs4,"41.96.155.224",2018/11/22,22:37,v=3.500,direct,Led
  ↪ Verte,+1.94279,VDC,+3.31442,mADC
5 cbs4,"41.96.155.224",2018/11/22,22:38,v=4.500,direct,Led
  ↪ Verte,+1.97476,VDC,+5.37012,mADC
6 cbs4,"41.96.155.224",2018/11/22,22:39,v=1.500,direct,Led
  ↪ Verte,+1.50366,VDC,+0.3409,µADC
```

```
7   cbs4,"41.96.155.224",2018/11/22,22:39,v=1.000,direct,Led
    ↪ Verte,+1.00343,VDC,+0.0060,µADC
8   cbs4,"41.96.155.224",2018/11/22,22:40,v=1.500,direct,Led
    ↪ Verte,+1.50365,VDC,+0.3018,µADC
```

Listing 7 – Exemple d'un fichier journal

À partir des données enregistrées (voir listing 7) lors de la première phase, nous tirons les conclusions suivantes :

— 91% des étudiants terminent le laboratoire en deux tentatives ;

— Le temps maximum cumulé d'utilisation du laboratoire pour la plupart des étudiants est de deux heures ;

— L'étudiant effectue en moyenne 200 requêtes ;

— Les temps $t_2 - t_3$ et $t_0 - t_4$ étaient presque égaux. Cela nous ramène à accorder moins d'attention à la latence du réseau dans le reste de la simulation.

— Le temps entre deux requêtes successives pour un étudiant donné est de 26 secondes. Nous allons l'interpréter comme le temps de réflexion et prise de note et l'appelons t_{Think} dans la simulation.

La figure 3.6 la distribution moyenne des requêtes par heure de la journée. Cette répartition est calculée pendant toute la période (deux semaines). Cela permet d'identifier trois phases :

— Une première phase entre 8h00 et 10h00 ou les requêtes augmentent progressivement (cette phase représente 10% du trafic)

— Une deuxième phase entre 10h00 et 18h00. où l'arrivée des étudiants est importante (cette phase représente 60% du trafic)

— Une troisième et dernière phase entre 18h00. et 00h00 où l'arrivée des étudiants suit une courbe descendante (cette phase représente 30% du trafic)

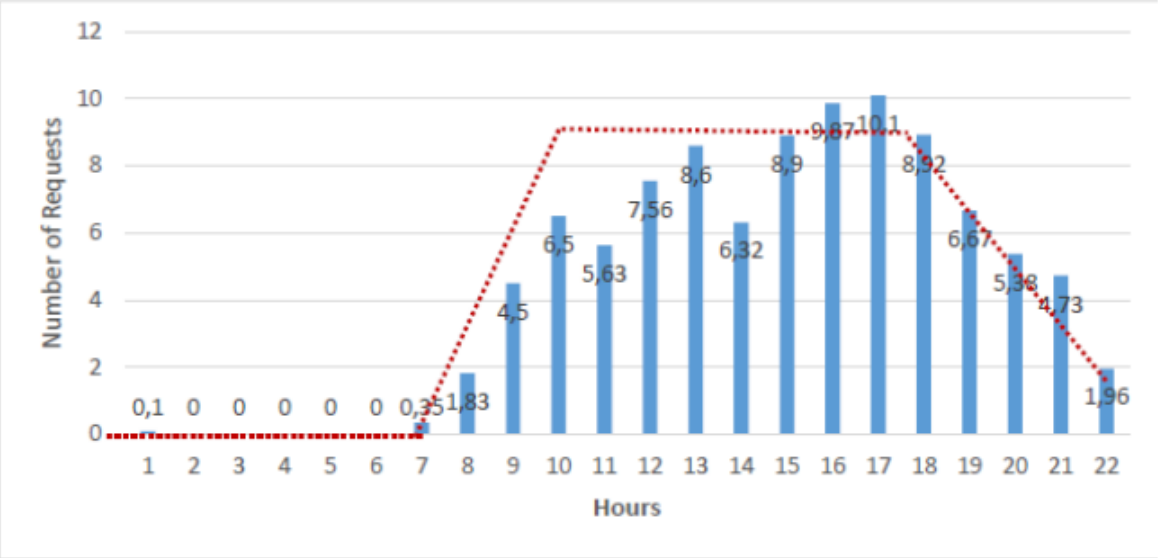


FIGURE 3.6 – Répartition moyenne des demandes par heure de la journée.

3.4.3 SIMULATION

Les tests de charge impliquent la génération des requêtes sur l'application déployée avec un certain niveau de charge afin que ses performances sous ce niveau de charge puissent être mesurées.

Ce niveau de charge correspond généralement au nombre d'utilisateurs qui participent simultanément à une session avec le laboratoire distant. Une suite de tests de charge typique consiste en des expériences menées à des niveaux de charge progressivement croissants jusqu'à ce que l'application atteigne la saturation. Ainsi, l'objectif principal des tests de charge est de déterminer la capacité de cette application, généralement en termes de nombre d'utilisateurs qu'elle peut prendre en charge.

Les tests de charge sont effectués à l'aide de l'un des nombreux outils générateurs de charge disponibles gratuitement ou dans le commerce. Ces outils prennent généralement en entrée une description de session (liste d'URL auxquelles un utilisateur accède dans une session), le temps de réflexion de l'utilisateur et un plan de test qui spécifie généralement un ni-

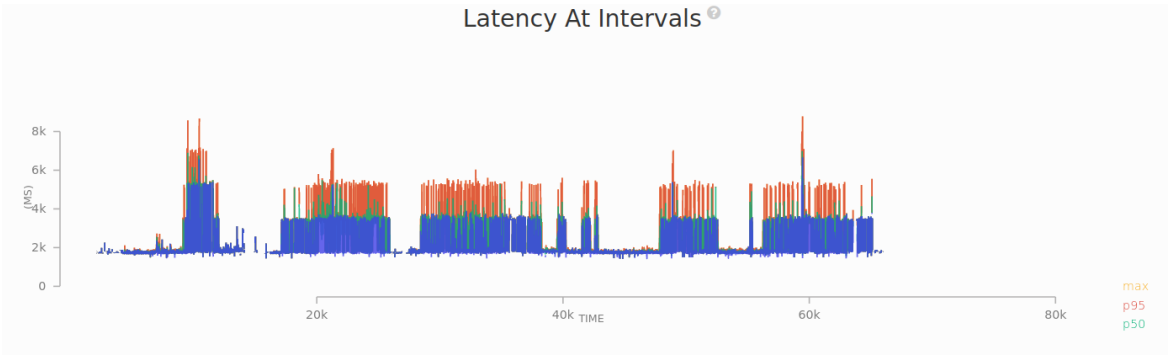


FIGURE 3.7 – Rapport Artillery temps de réponse pendant une journée.

veau de charge de départ, un incrément et un niveau de charge de fin. Pour chaque niveau de charge, les outils ont également besoin d'entrées telles que la durée globale du test.

Le test de charge de cette manière peut nécessiter plusieurs itérations avant que le test ne soit correctement configuré. Par exemple, la plage de niveaux de charge peut être erronée, c'est-à-dire que le niveau de charge le plus élevé s'avère trop faible par rapport à la capacité de l'application ou le niveau de charge le plus bas peut être trop élevé par rapport à la capacité. Dans les deux cas, si le test de la performance n'est pas bien formalisé, les tests de performances peuvent être mal interprétés, ou cela peut prendre beaucoup de temps et d'autres ressources pour comprendre et exécuter le test de charge dans la plage correcte.

La durée du test de charge est un autre paramètre qui peut mal tourner. Il peut être trop court, auquel cas il se peut que l'on n'obtienne pas les valeurs en régime permanent des métriques de performance à chaque niveau de charge, ou il peut être trop long, auquel cas le test global peut prendre beaucoup de temps.

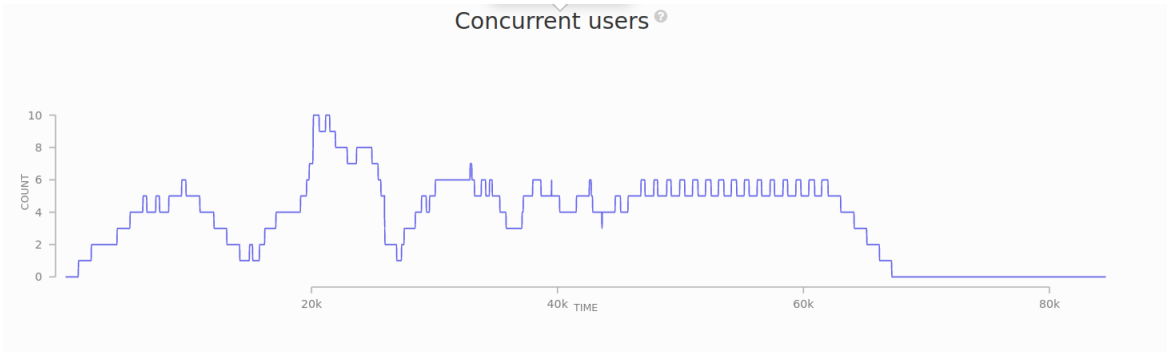


FIGURE 3.8 – Rapport Artillery Max utilisateurs pendant une journée

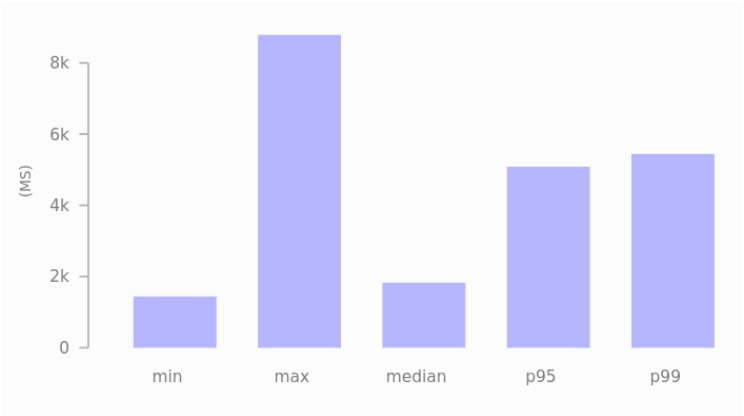
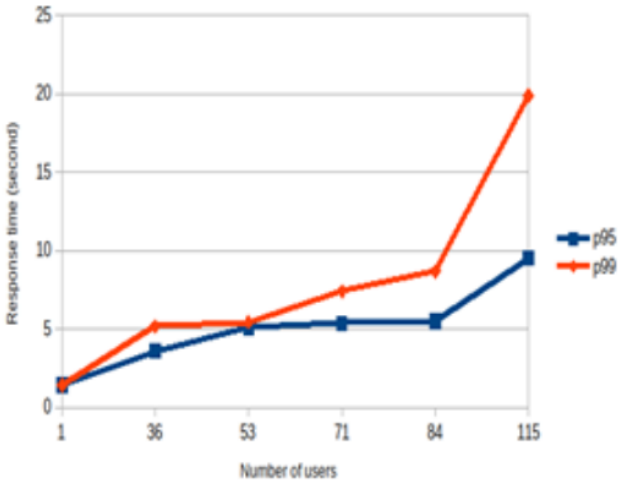
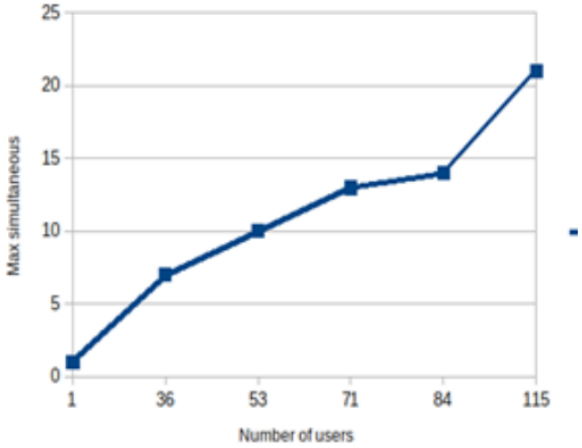


FIGURE 3.9 – Le graphique à barres Artillery temps de réponse



(a) Temps de réponse du laboratoire par rapport au nombre d'utilisateurs par jour.



(b) Nombre maximum d'utilisateurs simultanés par rapport au nombre
d'utilisateurs par jour.

FIGURE 3.10 – Résultats de la simulation : cas optimal

3.4.4 Résultat et discussion phase 2

Le serveur de laboratoire est configuré pour recevoir les requêtes envoyées par Artillery.io. Pour configurer les paramètres de chaque phase, on fixe le nombre N d'utilisateurs virtuels par jour, puis on calcule pour chaque phase les différents paramètres (*arrivalRate*, durée et *rampTo*) en fonction des observations réelles déjà vues dans la section précédente.

L'étape précédente est répétée plusieurs fois, augmentant le nombre d'utilisateurs virtuels à chaque étape pour tester la capacité maximale du serveur de laboratoire. Ensuite, à la fin de chaque test de charge, des métriques de performances sont prises telles que P95 et P99 (une valeur de latence de requête p99 de 5000 ms signifie que 99 requêtes sur 100 ont pris 5000 ms ou moins). Enfin, sur la base de ces métriques, il reste à décider s'il faut ou non réduire le nombre d'utilisateurs virtuels.

La figure 3.10a montre clairement que pour 53 utilisateurs par jour, les valeurs P90 ou P95 sont proches du temps de réponse max (5s) déjà fixé . Sachant qu'un laboratoire distant est disponible pendant une semaine à la disposition des étudiants, on peut atteindre donc environ 370 utilisateurs en une semaine ou 740 en deux semaines.

En plus du temps de réponse d'une seule instance de laboratoire distant, la figure 3.10b montre le nombre de maximum que le laboratoire peut atteindre. Nous pouvons conclure que pour notre scénario recherche nous pouvons atteindre jusqu'à 10 utilisateurs simultanés.

3.5 Conclusion

Dans ce chapitre, un modèle de partage pour les laboratoires distants est proposé et évalué pour l'enseignement de l'électronique analogique. Ce modèle d'accès simultané permet de partager d'une seule instance de

Nombre d'utilisateur/jours	Min	Max	P95	P99	Nb maximal d'utilisateurs simultanés
36	1,435	5,660	3,596	5,190	7
53	1,435	8,779	5,079	5,439	10
71	1,423	32,967	5,304	7,422	13
84	1,436	36,366	5,496	8,680	14
115	1,436	72,242	9,517	19,863	21

TABLE 3.1 – Résultats des itérations de simulation

laboratoire distant pour plusieurs étudiants en même temps. La mise en œuvre de cette solution sur un cas d'utilisation (caractérisation des diodes) montre comment atteindre 740 utilisateurs en deux semaines si nous fixons à 5 secondes le temps de réponse maximum que la requête ne peut pas dépasser.

	Categories	Attributs	Value
Données entrée Artillery	Phase 1	arrivalRate	0.005555555
		rampTo	0.0009523
		duree	7200 s (2 heures)
		mode	poisson
	Phases 2	arrivalRate	0.00095230
		rampTo	-
		duration	25200 s (7 heures)
		mode	poisson
	Phases 3	arrivalRate	0.00095230
		rampTo	0.00055555
		duration	21600 s (6 heures)
		mode	poisson
Donnees de sortie Artillery	Latence	Min	1435.7 ms
		Max	8779.7 ms
		Median	1822 ms
		P95	5079.8
		P99	5439.9
	Requêtes	Nombre de requêtes	10600

Conclusion générale

Dans l'introduction de cette thèse, le problème cible a été décrit et contextualisé. Le principal problème cible est la conception et l'implementation d'un environnement pour les laboratoires distants, qui prend en charge le partage des accès aux laboratoires. La politique de ce partage des accès simultanés doit assurer une qualité de service acceptable avec comme critère un temps de réponse 5 secondes au maximum.

Ce modèle de partage a été implémenté sur MostaLab, un système de management des travaux pratiques développé au sein de Laboratoire Électromagnétisme et Optique Guidée (LEOG) à l'Université de Mostaganem. MostaLab utilise une approche orientée microservices pour assurer une interopérabilité et un couplage faible entre les modules des laboratoires distants. L'algorithme de l'ordonnancement a été implémenté en tant que service à part. Cela permet de bénéficier d'autres fonctionnalités, telles que la file d'attente, ainsi que la possibilité de définir pour chaque instance d'un laboratoire distant sa propre gestion.

Ces expérimentations par la suite ont été intégrées dans les outils LMS en tant qu'outil fournisseur. Cette methode n'exige pas une intégration forte dans les outils LMS pour assurer une sécurité et une authentification unique.

Les performances de notre modèle de partage ont été évaluées dans un certain nombre de situations dans le chapitre 3, en tenant compte comme critère le temps de réponse des instruments et le temps de réponse global a ne pas dépasser qui est de 5 secondes. Pour ce faire, un test de charge sous Artillery.io a été développé, qui émule un groupe d'étudiants simultanés utilisant le système réel avec différentes configurations. Avec les données obtenues, il a été possible de mesurer et d'évaluer la mise en œuvre du modèle de partage de MostaLab, Cette simulation a été appliquée sur un laboratoire de caractérisation des diodes qui utilise trois instruments dans l'interface utilisateur mais seulemet deux instruments réel en production. Nous avons obtenu des résultats satisfaisants pour une seule instance de laboratoire, qui peuvent être en concurrence sans que le temps de réponse ne dépasse 5s pour 95% des étudiants. Aussi nous avons intégré MostaLab avec le LMS Moodle, mais nous n'avons pas testé avec d'autres comme edX [4] ou Canvas [5].

Dans nos futurs travaux, nous proposons de compléter cette recherche par l'amélioration de l'étude de performance pour inclure plus d'instances de laboratoire et une enquête sur les expériences d'apprentissage soutenues par les laboratoires distants en informatique et les réseaux informatiques.

Enfin, à la lumière de ces résultats, nous proposons de developper un processus qui permet une recherche efficace des laboratoires distants pour les besoins spécifiques des enseignants et des apprenants, en ne considérant que les ressources avec des licences de propriété intellectuelle ouvertes.

4. edX est une plateforme d'apprentissage en ligne (dite Massive Open Online Course (MOOC))https://www.edx.org/

5. Canvas est un LMS open source moderne développé et maintenu par Instructure Inc. https://github.com/instructure/canvas-lms

Bibliographie

[Abi+19] P. Abichandani et al. « Solar energy education through a cloud-based desktop virtual reality system ». *IEEE Access* 7, 2019, p. 147081-147093.

[Akt+96] B. Aktan et al. « Distance Learning Applied to Control Engineering Laboratories ». *IEEE Trans. on Educ.* 39 :3, 1996, p. 320-326. ISSN : 0018-9359. DOI : 10.1109/13.538754. URL : https://doi.org/10.1109/13.538754.

[Ale+17] V. Aleven et al. « Integrating moocs and intelligent tutoring systems : edx, gift, and ctat ». In : *Proceedings of the 5th Annual Generalized Intelligent Framework for Tutoring Users Symposium, Orlando, FL, USA*. 2017, p. 11.

[Ale+19] K. P. Alekseev et al. « Virtual Practices, Virtual Laboratories, and Virtual Internship Experience in Engineering Training ». In : *Handbook of Research on Engineering Education in a Global Context*. IGI Global, 2019, p. 390-403.

[Alv+16] G. R. Alves et al. « Spreading remote lab usage a system—A community—A Federation ». In : *2016 2nd International Conference of the Portuguese Society for Engineering Education (CISPEE)*. IEEE. 2016, p. 1-7.

[Ang+20] L. Angrisani et al. « A flexible remote laboratory with programmable device under test ». *Measurement* 156, 2020, p. 107584.

[AYO19] S. Admoko et al. « The implementation of guided discovery learning using virtual lab simulation to reduce students' misconception on mechanical wave ». In : *Journal of Physics : Conference Series*. T. 1417. 1. IOP Publishing. 2019, p. 012089.

[Ben+19] A. A. Benattia et al. « Design of a Low Cost Switching Board Enabling a Reconfigurable Remote Experiment », 2019.

[Bhu+21] V. J. Bhute et al. « Transforming traditional teaching laboratories for effective remote delivery—A review ». *Education for Chemical Engineers* 35, 2021, p. 96-104. ISSN : 1749-7728. DOI : https://doi.org/10.1016/j.ece.2021.01.008. URL : https://www.sciencedirect.com/science/article/pii/S1749772821000087.

[Bje+17] M. Bjekić et al. « Using computer for measurement and visualization of rotating magnetic field in AC machines ». *Computer Applications in Engineering Education* 25 :4, 2017, p. 608-624.

[BS17] P. Beňo et F. Schauer. « Remote laboratory management center REMLABNET embedded in University network-security and redundancy aspects ». In : *The 6th International Virtual Scientific Conference on Informatics and Management Sciences*. T. 26. 2017, p. 30.

[Cam+14] A. C. Caminero et al. « On the Creation of Customizable Laboratory Experiments : Deconstruction of Remote Laboratories to Create Laboratories as a Service (LaaS). » *International Journal of Online Engineering* 10 :6, 2014.

[Cob+10] A. Coble et al. « Delivering authentic experiences for engineering students and professionals through e-labs ». In : *IEEE EDUCON 2010 Conference*. IEEE. 2010, p. 1085-1090.

[Con21] I. G. L. Consortium. *Basic Overview of how LTI ® Works*. http://www.imsglobal.org/basic-overview-how-lti-works. [Online; accessed 19-June-2021]. 2021.

[COT10] B. Charroux et al. *UML 2 : pratique de la modélisation*. Pearson Education Paris, 2010.

[CSZ10] X. Chen et al. « Virtual and remote laboratory development : A review ». *Earth and Space 2010 : Engineering, Science, Construction, and Operations in Challenging Environments*, 2010, p. 3843-3852.

[De +20] L. De La Torre et al. « Automatic generation and easy deployment of digitized laboratories ». *IEEE Transactions on Industrial Informatics* 16 :12, 2020, p. 7328-7337.

[Eri17] B. Erinle. *Performance Testing with JMeter 3*. Packt Publishing Ltd, 2017.

[Eva+17] I. Evangelista et al. « Science education at high school : A VISIR remote lab implementation ». In : *2017 4th Experiment@ International Conference (exp. at'17)*. IEEE. 2017, p. 13-17.

[Gus+07] I. Gustavsson et al. « The visir project–an open source software initiative for distributed online laboratories ». In : *REV 2007*. 2007.

[GZ08] L. Gomes et J. G. Zubia. *Advances on remote laboratories and e-learning experiences*. T. 6. Universidad de Deusto, 2008.

[Har+08] V. J. Harward et al. « The ilab shared architecture : A web services infrastructure to build communities of internet accessible laboratories ». *Proceedings of the IEEE* 96 :6, 2008, p. 931-950.

[Har04] V. J. Harward. « iLab : A Scalable Architecture for Sharing Online Experiments Authors », 2004.

[IMS14] G. IMS. *IMS Learning Tools Interoperability™ (LTI) Messaging Framework*. https://www.imsglobal.org/specs/ltiv2p0/messaging-framework. [Online; accessed 03-June-2021]. 2014.

[Ism19] M. M. Ismail Belghit. « La mise en oeuvre d'une solution open source pour la gestion des laboratoires distants ». Memoire de Master. Universite Abdelhamid Ibn Badis Mostaganem, 2019.

[Kal+13] M. Kalúz et al. « Sharing control laboratories by remote laboratory management system weblab-deusto ». *IFAC Proceedings Volumes* 46 :17, 2013, p. 345-350.

[Low+09] D. Lowe et al. « Adapting a remote laboratory architecture to support collaboration and supervision ». *International Journal of Online Engineering* 5, 2009, p. 51-56.

[Low12] D. Lowe. « Impacts of scheduling algorithms on resource availability ». *Ascilite 2012*, 2012, p. 575-579.

[Low14] D. Lowe. « MOOLs : Massive open online laboratories : An analysis of scale and feasibility ». In : *2014 11th International Conference on Remote Engineering and Virtual Instrumentation (REV)*. IEEE. 2014, p. 1-6.

[MN06] J. Ma et J. V. Nickerson. « Hands-on, simulated, and remote laboratories : A comparative literature review ». *ACM Computing Surveys (CSUR)* 38 :3, 2006, 7-es.

[MN10] C. Maier et M. Niederstätter. « Lab2go–A Repository to Locate Online Laboratories. » *International Journal of Online Engineering* 6 :1, 2010.

[MNN11] J. Machotka et al. « The history of developments of Remote experiments ». Thèse de doct. World Institute for Engineering et Technology Education, 2011.

[Mou+21] M. Moussa et al. « MostaLab : Performance Evaluation of Simultaneous Access in Analog Remote Laboratories ». *iJOE* 17 :06, 2021, p. 99.

[MUD08] C. Martin-Villalba et al. « An approach to virtual-lab implementation using Modelica ». *Mathematical and Computer Modelling of Dynamical Systems* 14 :4, 2008, p. 341-360.

[NHH18] L. T. Neustock et al. « Remote experimentation with massively scalable online laboratories ». In : *Online Engineering & Internet of Things*. Springer, 2018, p. 258-265.

[NMN03] Z. Nedic et al. *Remote laboratories versus virtual and real laboratories*. T. 1. IEEE, 2003.

[NMN08] Z. Nedic et al. « Remote laboratory netlab for effective interaction with real equipment over the internet ». In : *2008 Conference on Human System Interactions*. IEEE. 2008, p. 846-851.

[Ord+18] P. Orduña et al. « Increasing the value of remote laboratory federations through an open sharing platform : LabsLand ». In : *Online Engineering & Internet of Things*. Springer, 2018, p. 859-873.

[PiO21] R. Pi.ORG. *Raspberry Pi GPIO*. https://www.raspberrypi.org/documentation/usage/gpio/. [Online ; accessed 21-June-2021]. 2021.

[RBJ11] T. Richter et al. « Lila : A european project on networked experiments ». In : *Automation, Communication and Cybernetics in Science and Engineering 2009/2010*. Springer, 2011, p. 307-317.

[RP18] L. F. Z. Rivera et M. M. L. Petrie. « The Remote Laboratory Management System (RLMS) Pattern ». In : *Latin American Pattern Languages of Programs Conference (PLoP). Valparaiso, Chile*. 2018.

[Sal+15] R. M. Salah et al. « Why VISIR ? Proliferative activities and collaborative work of VISIR system ». *EDULEARN15*, 2015.

[Sal+19] H. Saliah-Hassane et al. « 1876-2019 : IEEE standard for networked smart learning objects for online laboratories », 2019.

[San+12] E. Sancristobal et al. « State of art, initiatives and new challenges for virtual and remote labs ». In : *2012 IEEE 12th International Conference on Advanced Learning Technologies*. IEEE. 2012, p. 714-715.

[Sch+08] F. Schauer et al. « An easy-to-build remote laboratory with data transfer using the Internet School Experimental System ». *European Journal of Physics* 29 :4, 2008, p. 753.

[Sch+14] F. Schauer et al. « REMLABNET-open remote laboratory management system for e-experiments ». In : *2014 11th International Conference on Remote Engineering and Virtual Instrumentation (REV)*. IEEE. 2014, p. 268-273.

[Sch+16] L. C. Schlichting et al. « Remote Laboratory : application and usability ». In : *2016 Technologies Applied to Electronics Teaching (TAEE)*. IEEE. 2016, p. 1-7.

[Sch+17] F. Schauer et al. « REMLABNET IV-LTI federated remote laboratory management system with embedded multiparameter simulations ». *International Journal of Online Engineering*, 2017.

[SG07] C. Salzmann et D. Gillet. *Challenges in remote laboratory sustainability*. Rapp. tech. 2007.

[SGP16] C. Salzmann et al. « MOOLs for MOOCs : A first edX scalable implementation ». In : *2016 13th international conference on remote engineering and virtual instrumentation (rev)*. IEEE. 2016, p. 246-251.

[SPM20] A. Soriano et al. « Virtual Laboratory to Face New Challenges in the Industry : Virtual Laboratory as Part of the New Education Age ». In : *Revolutionizing Education in the Age of AI and Machine Learning*. IGI Global, 2020, p. 114-129.

[SR14] H. Saliah-Hassane et A. Reuzeau. « Mobile open online laboratories : A way towards connectionist massive online laboratories with x-API (c-MOOLs) ». In : *2014 IEEE Frontiers in Education Conference (FIE) Proceedings*. IEEE. 2014, p. 1-7.

[Taw+14] M. Tawfik et al. « Laboratory as a Service (LaaS) : A Novel Paradigm for Developing and Implementing Modular Remote Laboratories. » *International Journal of Online Engineering* 10 :4, 2014.

[Thi21] I. ThingsBoard. *Thingsboard*. https://thingsboard.io. [Online; accessed 19-June-2021]. 2021.

[UU20] J. Umborg et A. Uukkivi. « Continuity in the Development of Technical Thinking ». In : *Developing Technology Mediation in Learning Environments*. IGI Global, 2020, p. 96-116.

[You+20] M. Youssef et al. « Developing engaging remote laboratory activities for a nonmajors chemistry course during COVID-19 ». *Journal of Chemical Education* 97 :9, 2020, p. 3048-3054.

[ZLB16] M. Zappatore et al. « Enabling MOOL in acoustics by mobile crowd-sensing paradigm ». In : *2016 IEEE Global Engineering Education Conference (EDUCON)*. IEEE. 2016, p. 733-740.

[Zut+10] D. G. Zutin et al. « Lab2go—A repository to locate educational online laboratories ». In : *IEEE Educon 2010 Conference*. IEEE. 2010, p. 1741-1746.

[Zut+16] D. G. Zutin et al. « Online lab infrastructure as a service : A new paradigm to simplify the development and deployment of online labs ». In : *2016 13th International Conference on Remote Engineering and Virtual Instrumentation (REV)*. IEEE. 2016, p. 208-214.

Annexes

Artillery

Listing 8 – Code YAML définissant les phases

```yaml
config:
   target: "http://app.leog.univ-mosta.dz/tp3/"
    payload:
  - path: "voltage.csv"
fields:
- "vin"
- path: "users.csv"
fields:
- "username"
plugins:
metrics-by-endpoint:
useOnlyRequestNames: true
http:
phases:
- arrivalRate: 0.012
duration: 3600
mode: "poisson"
- arrivalRate:0.012
duration: 3600
mode: "poisson"
rampTo: 0.034
- arrivalRate:0.034
duration: 3600
mode: "poisson"
scenarios:
```

```
26      - flow:
27      - log: "New virtual user running"
28      - loop:
29      - get:
30      url: "/scpi.php"
31      headers:
32      User-Agent: "Mozilla/5.0 (X11; Ubuntu; Linux x86_64;
33      rv:79.0) Gecko/20100101 Firefox/79.0"
34      Accept: "*/*"
35      Accept-Language: "en-US,en;q=0.5"
36      X-Requested-With': "XMLHttpRequest"
37      Connection': "keep-alive"
38      Referer': "http://app.leog.univ-mosta.dz/tp3/"
39      qs:
40        msg1:     "INST OUT1\n Volt {{ vin }}\nINST OUT1\n
          ↪  OUTP:CHAN ON\n OUTP:MAST ON\n Volt?\n"
41        msg2:     "CONF:VOLT:DC\n READ?\n"
42        msg3:     "MEASure:CURRent:DC?\n"
43        DMfunc1:  "volt"
44        PSfunc:   "volt"
45        DMfunc2:  "curr"
46        rolePs:   " "
47        roleDm1:  "1"
48        roleDm2:  "2"
49        swt:      "0010001101000001"
50        swc:      "0010001001000011"
51        user:     "artillery-{{ username }}"
52        psvolt:   "{{ vin }}"
```

```
53          - think: 26
54          count: 200
```

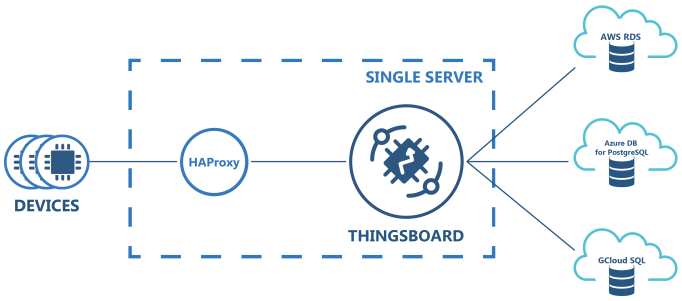


FIGURE .11 – standalone thingsboard

THINGSBOARD

INSTALLATION

ThingsBoard peut être installé et exécuté sur divers systèmes d'exploitation (Windows, Linux, Mac, ...) et sur l'environnement (système, machines virtuelles, Docker1, ...). Le processus d'installation du serveur ThingsBoard est une opération très simple. Suivant la documentation officielle2, nous avons pu l'installer sur un système client Ubuntu sous Docker, en suivant les étapes suivantes :

Listing 9 – ThingsBoard Installation

```
#Installation Docker
sudo apt-get update
sudo apt-get install docker-ce docker-ce-cli containerd.io
#Installation Things Board
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-cassandr
```

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.

Publications scientifiques

MostaLab : Performance Evaluation of Simultaneous Access in Analog Remote Laboratories

Mohammed Moussa, Abdelhalim Benachenhou Université Abdelhamid Ibn Badis

Date de publication 25/06/2021

International Journal of Online and Biomedical Engineering (iJOE)

Abstract

A service-oriented architecture for simultaneous access in the field of remote labs has been proposed and validated using stress load testing. The innovation of this work lies on the use of the parameters collected for the typical student and tested with the Artillery.io tool. Then, we have evaluated the performance of the laboratory by defining 5s the maximum waiting time that a request cannot be exceeded. This article also describes a use case showing how this architecture was designed and developed with 109 students.

Keywords Remote laboratories, access sharing, stress load test, performance

An Implementation of Microservices Based Architecture for Remote Laboratories

Mohammed Moussa, Abdelhalim Benachenhou, Smail Belghit, Abderrahmane Adda Benattia, Abderrahmane Boumehdi

Date de publication 2020/2/26

International Conference on Remote Engineering and Virtual Instrumentation

Pages 154-161

Springer, Cham

Abstract In the fields of science, technology, engineering and mathematics, remote laboratories offer a new opportunity to increase the number of hours devoted to scientific experiments. Universities facing the growing number of students need to pool their equipment. This article describes an implementation, based on microservices, of a remote lab management and federation solution. This solution uses the ThingsBoard platform interfaced by a REST API. The feasibility has been demonstrated by the deployment of eight instances of practical work used by 120 students.

Keywords Remote laboratories, Federation, Microservices, ThingsBoard REST.

Design of a Low Cost Switching Board Enabling a Reconfigurable Remote Experiment

Abderrahmane Adda Benattia, Mohamed Moussa, Abdelhalim Benachenhou, Abdelhamid Mebrouka Date de publication 2019/8/23 Pages 32-41 Éditeur International Association of Online Engineering

Abstract Most of currently remote laboratories implementations include interactive experimentation. In this case, students use real devices and equip-ment to perform real experiments, which need some flexibility of interaction with the hardware platform. The hardware platform is composed of a Raspberry Pi as a lab server, a switching board (SB), a practical work circuit board and som e measurement instruments. The SB is used to make configuration of exper-imentation by establishing connection between the practical work circuit and measurement instruments. During the experimentation process, students change the setup using a web page. In the background, the hardware configuration is realized using SB, which is controlled by the lab server. The purpose of this work is to develop a new SB in order to provide more possibilities, interaction flexibility with the hardware platform, ease of use, improve performance in re-sponse time and finally reduce the cost of the hardware. The SB is based on switches instead of relays. This board can be plugged directly on a Raspberry Pi to facilitate the assembly. It extends the "SPI" bus in order to control some elec-tronic components such as digital potentiometers. Its use is illustrated with a circuit with multiple combinations

Keywords Remote Laboratory, analog electronics, Switching board, Reusability, Raspberry Pi

Development of an Automatic Assessment in Remote Experimentation Over Remote Laboratory

Abderrahmane Adda Benattia, Abdelhalim Benachenhou, Mohammed Moussa Date de publication 2018/3/21 Conférence International Conference on Remote Engineering and Virtual Instrumentation Pages 136-143 Éditeur Springer, Cham

Abstract

In the last few years, performing experimentation over remote laboratory became a reality. In fact, for managing a large class, this new situation needs some appropriate pedagogical scheme. Else, the entire system must be reviewed. We have developed a method to set up an automatic assessment in remote experimentation over a flexible remote laboratory. Automatic assessment in remote experimentation can provide important benefits including : improved situation awareness, more accurate management for large class, and improved overall performance. We present a new approach to set up online experimentation under a flexible remote laboratory. In fact, we develop a particular pedagogical scheme for remote experimentation. On the basis of this analysis, we developed an automatic assessment system to evaluate remote experimentation for a large number of students. Hardware architecture consists of servers ; some measurement instruments and electronic circuits, where the software architecture is based on web development tools. Furthermore, to embed pedagogical material for our system, we have used Moodle LMS platform. We illustrate an example of remote experimentation prototype which allows students to manipulate a dipole circuit workbench remotely. A set of students can connect to the remote lab at any time and perform manipulations and obtain personal and appropriate results from the shared equipment. The system includes auto test with customized interfaces for accessing to the same platform. Some results of manipulation are presented.

Keywords Online experimentation Automatic assessment Remote laboratory Moodle LMS

Work-in-Progress : A Smart Scheduling System for Shared Interactive Remote Laboratories

Mohammed Moussa, Abdelhalim Benachenhou, Abderrahmane Adda Benatia

Date de publication 2017/9/27

International Conference on Interactive Collaborative Learning

Pages 601-606

Éditeur Springer, Cham

Abstract Online laboratories are an innovative solution for providing hands-on experience in the STEM Education. A large number of students are using remote laboratories (RL) to meet their practical needs and enriching their learning. The deployment of a unique instance of the RL cannot provide all requirements for the growing number of students. For that purpose, the use of multiple instances of RLs can produce many advantages. We propose in this paper a new method for managing access to these RLs. The implemented system provides a central point of control all requests to access to the RLs. This approach is transparent to the user. The system under development supplies an abstraction layer. The goal of the work is to build a web service through which the access to RLs is granted. The student will don't need to reserve a time slot to access the RL. In addition, the system supports the registration of new RLs in a transparent way for teachers.

Keywords Remote laboratories Smart scheduling Web services REST

Test run on 06 Oct 2020 09:51:49

Summary

Test duration	84590 sec
Scenarios created	53
Scenarios completed	53

Scenario counts:

0	53 (100%)

Codes

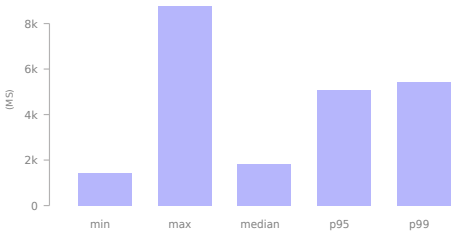
200 (https://en.wikipedia.org/wiki/List_of_HTTP_status_codes#2xx_Success)	10600

Errors

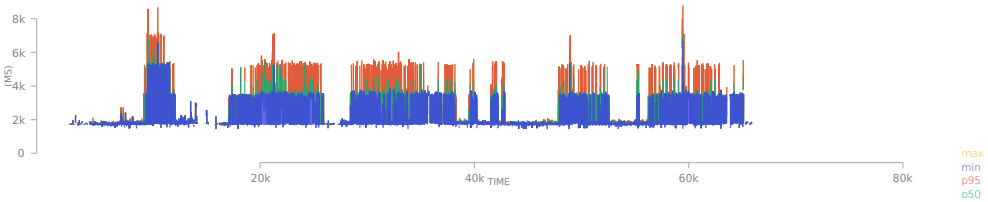
✔ Test completed without network or OS errors.

Charts

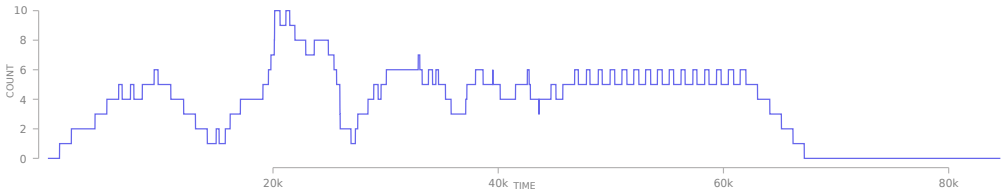
Overall Latency Distribution



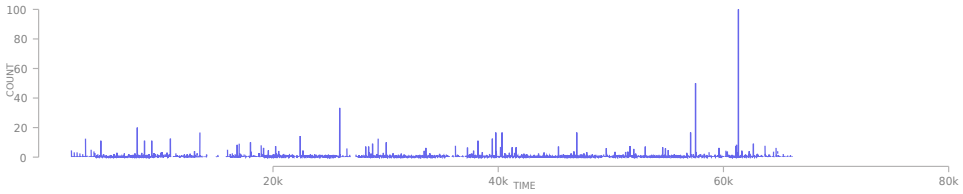
Latency At Intervals



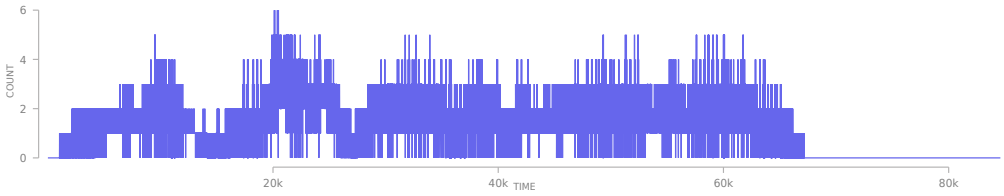
Concurrent users



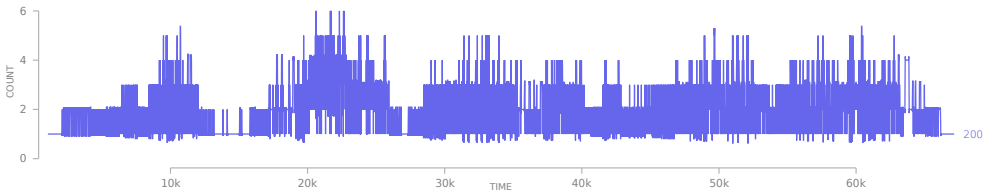
Mean RPS



RPS Count



HTTP codes



//scpi.php distribution