

Laboratoire numérique pour le MOOC “ Métrologie d’internet : une introduction pratique ”

Marc Chambon

▶ **To cite this version:**

Marc Chambon. Laboratoire numérique pour le MOOC “ Métrologie d’internet : une introduction pratique ”. [Rapport Technique] Inria. 2017, pp.13. hal-01513292v1

HAL Id: hal-01513292

https://inria.hal.science/hal-01513292v1

Submitted on 24 Apr 2017 (v1), last revised 5 Sep 2017 (v2)

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.



Laboratoire numérique pour le MOOC« Métrologie d'internet : une introduction pratique »

Marc Chambon

TECHNICAL REPORT

April 2017

DSI-SEISM / Inria Learning Lab

ISSN 0249-6399



Laboratoire numérique pour le MOOC« Métrologie d'internet : une introduction pratique »

Marc Chambon[1]
DSI-SEISM – Inria Learning Lab

Technical Report — Avril 2017 —13 pages.

Résumé : Un "laboratoire virtuel" a été développé pour les besoins d'un cours d'introduction à la métrologie d'internet. Ses objectifs sont d'ordre pédagogique, à savoir une meilleure assimilation des concepts par les apprenants via la réalisation de divers exercices. Il s'agit d'une application web qui permet aux apprenants de soumettre ou consulter des expériences sur un sous-réseau de serveurs dédiés, nommé plateforme Planetlab Europe. Ces expériences sont de simples commandes réseaux UNIX, exécutées à distance sur un serveur donné de ladite plateforme, choisi par l'apprenant. Ce service web est conçu pour être facile d'usage, et également pour solliciter au minimum la plateforme en cas de montée en charge. Il communique avec le serveur vitrine de la plateforme via une API REST, et est basé sur le framework python django. Le cœur de l'application est une base de données stockant les correspondances entre les apprenants et leurs expériences ; et dont l'état est régulièrement actualisé afin de soumettre les nouvelles expériences à la plateforme et d'y récupérer celles qui sont achevées. Ce processus d'actualisation est confiée à un processus indépendant du serveur web (Celery). Ce cours d'introduction a eu autant de succès que les autres de sa plateforme de MOOCs (FUN : France Université Numérique).

Mots clés : MOOC, FUN, métrologie d'internet, MVC, django, REST, celery

[1] Inria – Marc.Chambon@inria.fr

RESEARCH CENTRE
GRENOBLE - RHÔNE-ALPES

Inovallée
655 avenue de l'Europe - Montbonnot
38334 Saint Ismier Cedex France

Abstract: A "virtual laboratory" has been developed for the needs of an introduction course about internet measurements. Its goals is to allow learners to solidify their knowledge by performing various exercises. It consists of a web application that allows both submission and visualization of experiments executed on a cluster of dedicated servers, the Planetlab Europe testbed. Such experiments are common UNIX network commands such as "ping" or "traceroute", but remotely performed on a user-selected testbed's server. This web service is designed to be used with ease by learners, and also to protect the testbed against excessive load. It communicates with the testbed front-end server through a REST API, and is developed with the django python framework. The core of the application is the database that stores a mapping between the learners and their experiments, and whose state is regularly refreshed in order to submit new experiments to the testbed and also poll finished ones. The database refresh steps are delegated to a separated, Celery, process. This introductory course has received as much success as the other ones of its platform (FUN : France Université Numérique).

Key-words: MOOC, FUN, internet measurement, MVC, django, REST, celery

1. Contexte 6
2. Description du besoin 6
3. Architecture technique 7
4. Cadre d'utilisation 8
5. Implémentation 9
5.1. Interface 9
5.2. Serveur HTTP et base de données 10
5.3. Tâche de synchronisation 11
6. Retour d'expérience 12
7. Remerciements 12
8. Bibilographie 12

1. Contexte

Selon Renata TEIXEIRA, co-autrice du MOOC, Internet est fondamentalement un système complexe à étudier (terme « métrologie »). En effet, un comportement donné tel un problème de connexion peut aussi bien provenir d'une panne de serveur que d'un souci sur l'un des centaines de milliers de sous-réseaux qui le constituent.

Dans un souci de sensibilisation, Renata et son collègue Timur FRIEDMANN ont décidé de lancer en avril 2016 un cours d'introduction sur le sujet [1]. Celui-ci, en anglais, s'adresse en priorité à des opérateurs et concepteurs de réseaux de niveau Master : Timur FRIEDMANN souhaite qu'ils y apprennent les méthodes pour découvrir la structure de ce réseau mondial, puis à affiner leur représentation via l'usage des notions de délais, pertes de paquets, connectivité, bande passante et trafic.

Ces moyens sont présentés de manière très pratique via des expériences encadrées sur un « échantillon » d'Internet : la plateforme de recherche PlanetLab, spécifiquement sa section européenne [2]. Il s'agit d'un réseau de serveurs (nodes) publiquement mis à disposition par plusieurs dizaines d'instituts académiques afin d'expérimenter de futurs protocoles, et donc de faire avancer les réflexions sur l'Internet du futur. Les nodes concernés se comptent actuellement en quelques centaines et sont répartis sur environ 200 sites géographiques européens. Les auteurs sont des contributeurs actifs sur cette plateforme.

Cependant, cette plateforme est consultable uniquement via des requêtes en ligne de commande. Les auteurs souhaitent que les apprenants aient une interface plus ergonomique.

Le contexte d'implémentation est le suivant :

- interface avec la plateforme Planetlab en cours de développement par l'équipe des auteurs (instabilités à prévoir à la fois au niveau de l'interface et de la disponibilité du service) ;
- plateforme de MOOCs France Université Numérique (FUN) [1] devant être adaptée pour fournir des informations d'authentification.

2. Description du besoin

Lors des discussions préliminaires à la conception de ce MOOC, les auteurs nous ont exposé les besoins suivants :

- un primordial, qui consiste à offrir aux apprenants une interface graphique (Figs. 1, 2 et 3) pour à la fois consulter des expériences de métrologie sur le réseau Planetlab Europe et en soumettre de nouvelles, en quelque sorte offrir un « laboratoire numérique » de métrologie d'Internet ;
- un second d'administration, avec la possibilité pour les auteurs de pré-remplir l'interface graphique avec des expériences déjà exécutées et affichées avant toutes les autres, de manière à ce que les apprenants se familiarisent avec l'interface et les expériences ;
- un troisième de collecte d'informations anonymes d'usage de l'application par les apprenants, qui sera traité dans un rapport ultérieur de l'équipe;
- un dernier technique, requis pour éviter une potentielle surcharge de la plateforme en cas d'une forte participation des étudiants du MOOC (rappel : MOOC anglophone). Pour pallier à cela, le nouveau service se connecte au serveur vitrine de Planetlab via un compte unique, et soumet ou récupère des expériences de manière contrôlée, quelques fois par minute. A ce sujet, les apprenants sont prévenus durant le cours qu'il ne s'agit pas d'un service interactif. Aussi sont-ils plutôt encouragés à soumettre plusieurs expériences à un moment de la journée puis à en consulter leurs statuts à un autre.

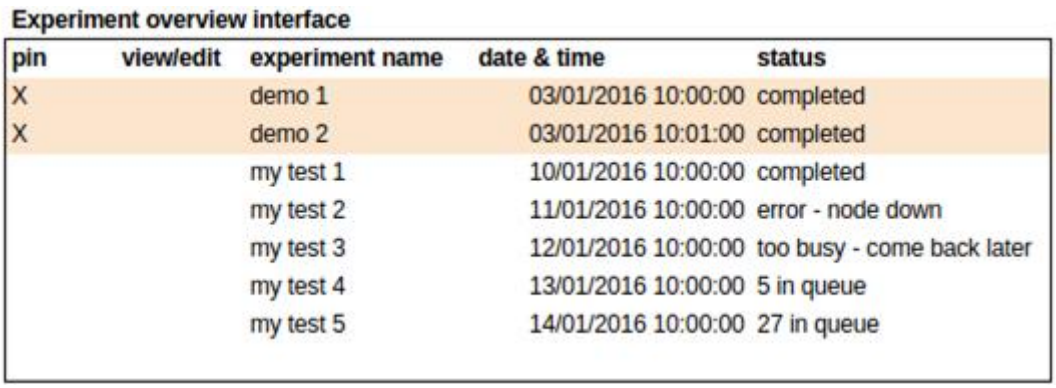


Figure 1 – Brouillon d'interface fourni par les auteurs :
panneau listant les expériences soumises au service Planetlab ainsi que leurs statuts.

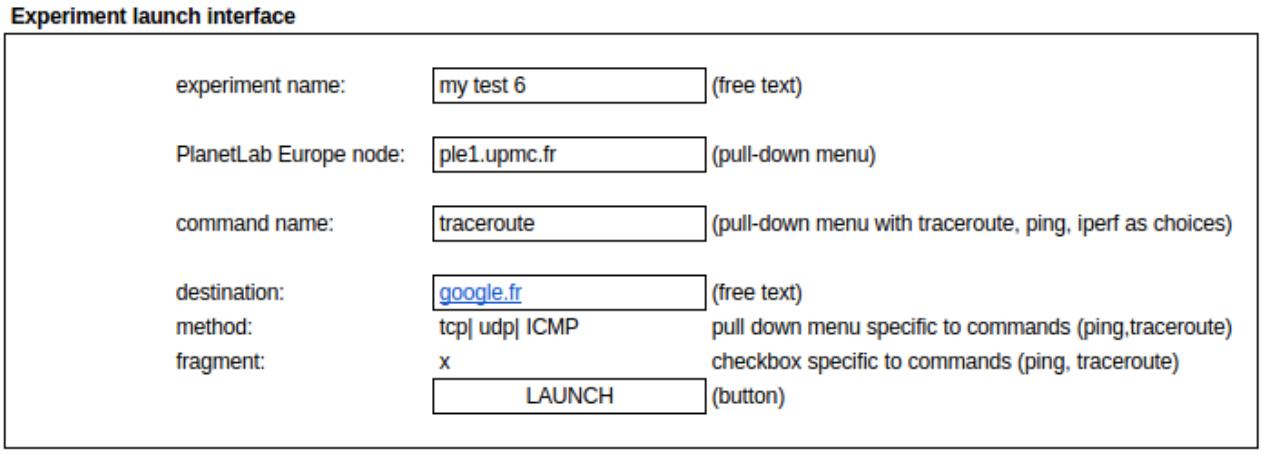


Figure 2 – Brouillon d'interface fourni par les auteurs:
panneau permettant de soumettre de nouvelles expériences.

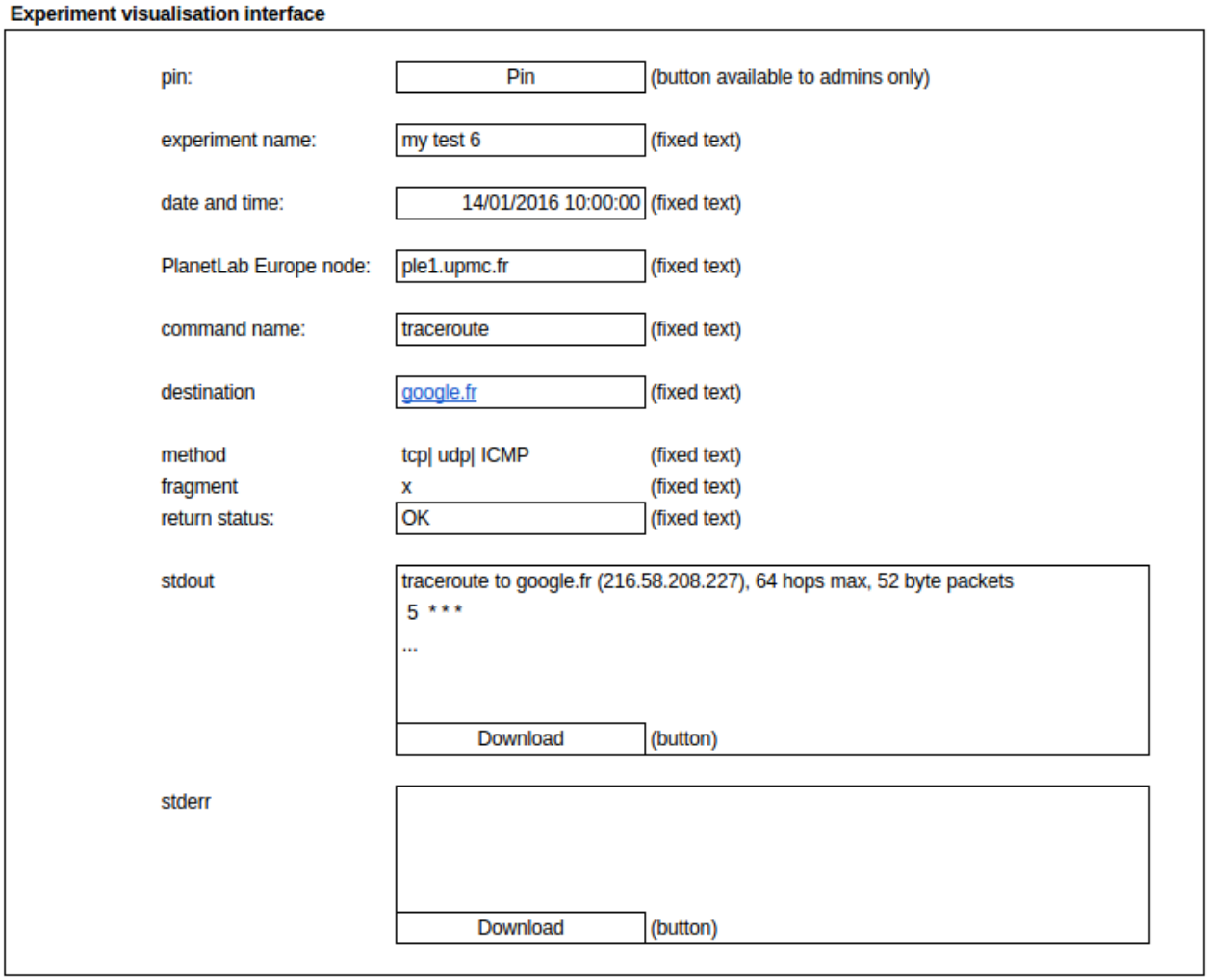


Figure 3 – Brouillon d'interface fourni par les auteurs:
panneau de consultation des détails d'une expérience donnée.

3. Architecture technique

Le service Web est articulé autour de 2 serveurs et d'un service communiquant via Internet :

- le serveur vitrine de la plateforme Planetlab, hors de notre responsabilité, qui reçoit des requêtes de nouvelles expériences et envoie les résultats via une API REST ;
- un service fourni par la plateforme FUN, qui permet d'authentifier les apprenants ;
- un serveur Inria.

Le serveur Inria est constitué :

- d'une base de donnée relationnelle listant les expériences soumises ;
- d'une tâche de fond (thread dédié) qui synchronise la base de données avec le serveur Planetlab ;
- d'un serveur web HTTP qui soumet une interface graphique aux apprenants, leur permettant d'ajouter ou de consulter des expériences dans la base de données. Il est accédé, côté client, via une iframe dans le cours FUN. L'URL de l'iframe est fonction de l'apprenant, car au nom de domaine de notre serveur est ajouté comme paramètre l'identifiant anonyme de l'apprenant, fourni par le service FUN.

L'interface graphique fait usage du Javascript en premier lieu pour l'autovalidation des paramètres d'expériences, dans un souci d'accompagner l'apprenant sur ce qu'il peut et ne peut pas faire et éviter que le serveur ait à rejeter trop d'expériences invalides.

4. Cadre d'utilisation

Lorsque l'apprenant se connecte, il lui est présenté une liste d'expériences « épinglées », c'est-à-dire déjà effectuées par les professeurs à titre d'exemples (Fig. 4). L'apprenant doit explicitement rafraîchir l'application (bouton en haut à droite) pour effectuer une mise à jour. L'apprenant peut soumettre de nouvelles expériences en cliquant sur l'onglet « new experiment » (Fig. 5). Une nouvelle expérience doit avoir un nom, un serveur sur lequel elle sera exécutée, et une commande réseau Unix (seulement ping ou traceroute dans la première version du service) avec optionnellement une URL de destination (ex : ping www.inria.fr) ainsi que des arguments optionnels. L'apprenant peut également supprimer ou consulter via des boutons une expérience de la liste.

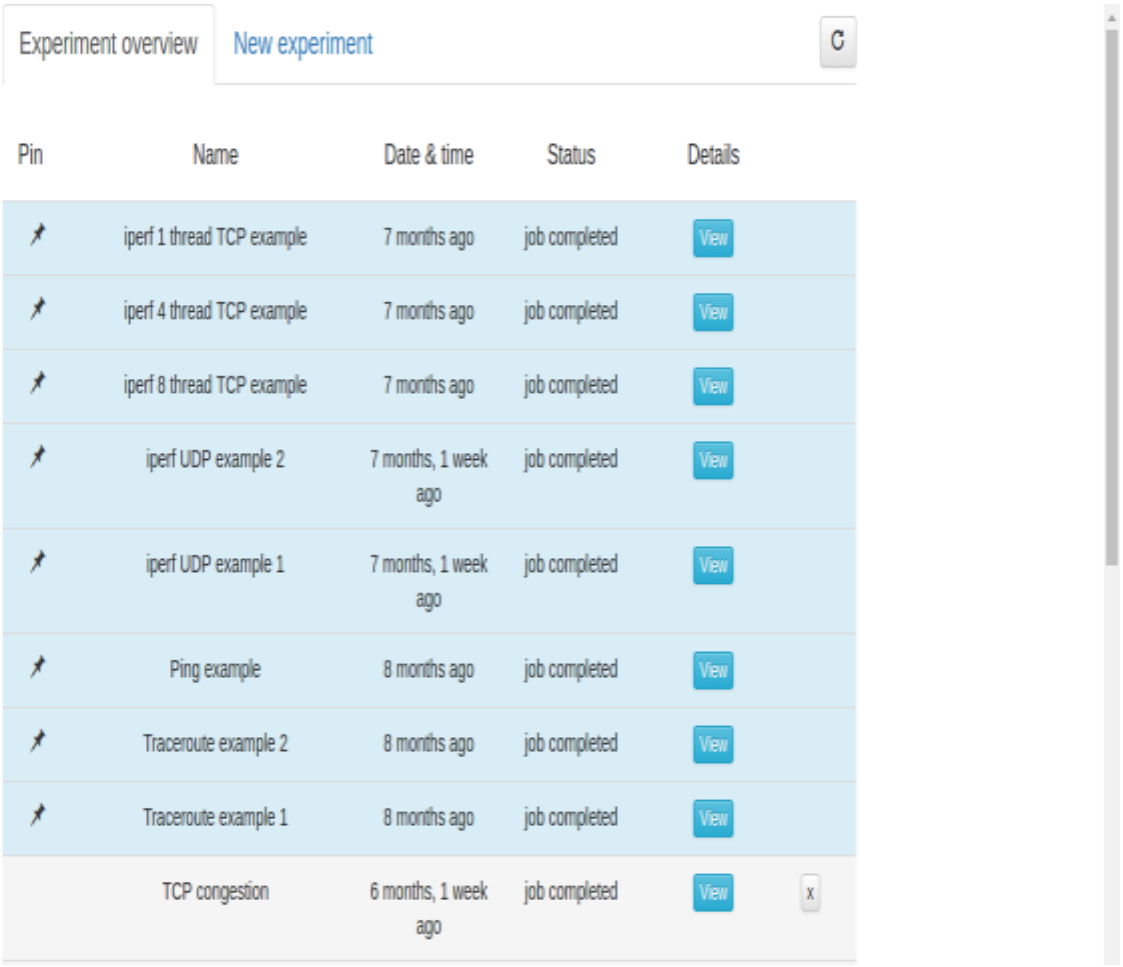


Figure 4 – Panneau listant les expériences soumises au service Planetlab ainsi que leurs statuts.



Figure 5 – Panneau permettant de soumettre de nouvelles expériences sur le service Planetlab.

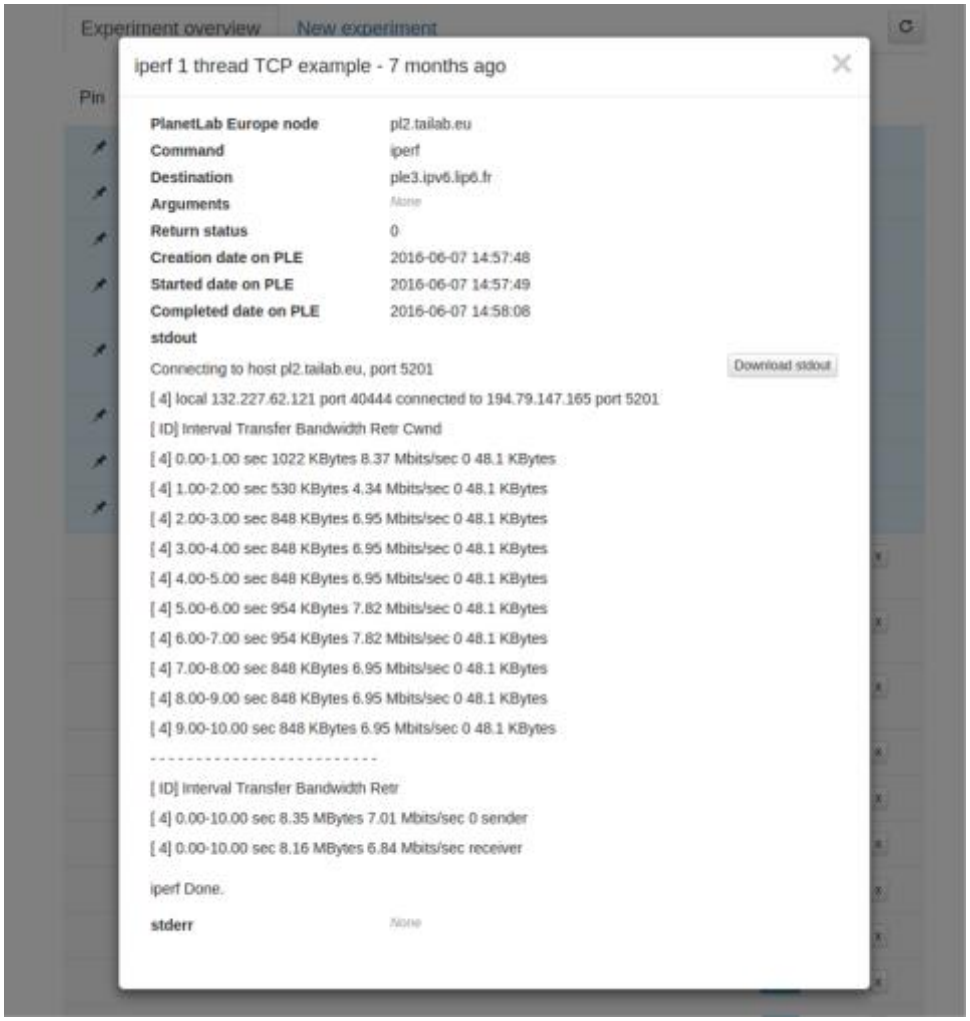


Figure 6 – Panneau modale présentant les détails d'une expérience soumise, invoqué en cliquant sur le bouton « view » de l'expérience concernée (Fig. 4).

5. Implémentation

5.1. Interface

Le framework Bootstrap de Twitter (version 3) a été choisi car parfaitement adapté pour concevoir rapidement une interface conviviale (apparence des boutons, jeu d'icônes, responsive design). L'interface de base présente un système de deux onglets permettant de naviguer entre la liste des expériences déjà soumises (Fig. 4) et le panneau de soumission de nouvelles expériences (Fig. 5).

Comme précisé dans la partie architecture, l'interface fait usage du Javascript, ce qui permet également d'offrir une meilleure expérience utilisateur lors de l'affichage des détails d'une expérience (Fig. 6), de sa suppression de la liste, ou encore lors du filtrage des nodes opérationnels sur la plateforme Planetlab Europe (Fig. 7) [3].

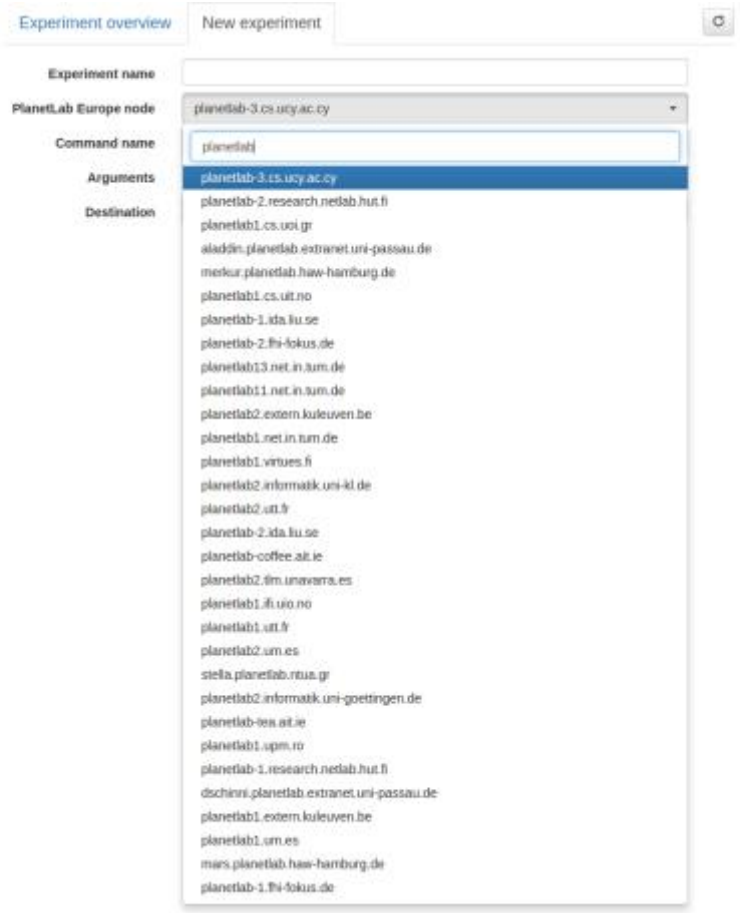


Figure 7 – Filtrage des nodes Planetlab.

La génération des interfaces présentées aux figures 4, 5 et 6 repose respectivement sur les fichiers de template (cf. partie suivante) job-list.html, job-new.html et job-visualize.html du dossier jobs/templates.

5.2. Serveur HTTP et base de données

Le serveur web est développée en Python 3 et fait appel au framework MVC Django [4] (ndlr : les vues au sens de Django ressemblent à des contrôleurs, la vue classique est plutôt le template complété renvoyé dans la réponse HTTP, cf. méthode as_view). Django dispose d'un ORM, configuré ici pour utiliser Postgresql comme base de données (fichier metrology/settings.py). De plus, il offre par défaut un mode d'administration qui permet de modifier simplement les données stockées en base (ex : épingler une expérience).

A chaque requête HTTP entrante, un contrôleur général (fichier jobs/urls.py) choisit la vue (fichier job/views.py) à exécuter en fonction de l'URL associée. En fin d'exécution, celle-ci soit 1) sous-traite le traitement final en refaisant appel au contrôleur mais avec une nouvelle URL interne signifiant par exemple que la soumission d'une nouvelle expérience s'est effectuée avec succès, soit 2) renvoie un fichier HTML construit via un fichier de template (fichiers dans dossiers job/templates).

Ces vues sont des classes, développées de la façon suivante :

- une classe mère BaseView contient les attributs et méthodes indispensables à toute vue de l'application. Dans notre cas, elle dérive elle-même de la classe Django FormView, qui contient plusieurs méthodes génériques (ex : as_view pour renvoyer une vue au sens classique du terme, get_context_data pour récupérer les informations transmises avec la requête, etc.). Cette classe contient comme attributs une référence vers une classe de validation des paramètres contenus dans la requête (form_class, propre à toute classe FormView, fichier jobs/forms.py), la liste des nodes Planetlab opérationnels (active_node_urls) et enfin les propriétés de la nouvelle expérience soumise si tel

est le cas (cf. paragraphe suivant). BaseView contient autrement des méthodes métier explicites : récupération de la liste des expériences depuis la base de données (get_jobs_from_db), ajout d'une nouvelle expérience (MOOC_create_job), etc.
- une classe fille DeleteJob qui a comme attribut une référence vers un fichier de template, qui permet de retourner des vues (au sens classique) spécifiques à un succès ou à un échec de la tâche.

Les données manipulées par ces classes sont déclarées sous forme de modèles (fichier jobs/models.py), qui spécifient la façon dont elles seront représentées en base de données. Ils se résument à une classe principale Job contenant les informations propres à toute expérience.

Les champs de la classe Job sont les suivants :
- hash : identifiant anonyme de l'apprenant qui a lancé la nouvelle expérience, fourni par la plateforme Planetlab (ex : 1f953fd4-fcd1-466d-836f-ec13495e2c30), mais qui n'est pas utilisé comme clé primaire pour éviter tout dysfonctionnement si cette plateforme est inaccessible,
- PLE_creation_date : date d'enregistrement de la nouvelle expérience sur la plateforme Planetlab,
- PLE_started_date et PLE_completed_date : dates respectives de début et de fin d'exécution de l'expérience sur la plateforme,
- node (classe Node) : référence du node Planetlab sur lequel l'expérience sera effectuée,
- command (classe Command) : référence de la commande réseau (ping, traceroute, etc.) qui sera exécutée durant l'expérience par la plateforme Planetlab,
- student_id : identifiant anonymisé de l'apprenant initiateur de l'expérience, issu du service FUN,
- arguments : arguments propres à la commande réseau choisie, sous forme textuelle,
- destination_url : la commande réseau choisie testera cette URL,
- status (class JobStatus): statut de l'expérience,
- return_status : true uniquement si expérience terminée avec succès,
- stdout : résultat textuel de la commande réseau choisie,
- stderr : erreurs rencontrées durant l'exécution de la commande réseau choisie, textuelles,
- is_pinned : true uniquement si expérience est épinglée,
- is_sent : false tant que l'expérience n'a pas été soumise avec succès à la plateforme Planetlab.

La classe annexe Node possède ces champs :
- url : URL du node Planetlab sur lequel sera exécutée la commande choisie,
- testbed : référence vers la plateforme d'expérimentation (toujours PlanetLab dans notre cas),
- access_status : true uniquement si le node est disponible pour expérimentation,
- state : état du node considéré.

Enfin les classes annexes JobStatus et Command sont des énumérations, c'est-à-dire qu'elles peuvent uniquement prendre un nombre fixé de valeurs (ex : chaînes de caractère ping et traceroute pour Command, et chaine de caractères completed, waiting, running, error, pour JobStatus).

En ce qui concerne la sécurité du service, elle concerne avant tout celle de la plateforme Planetlab. Elle est sensée être assurée par leur serveur vitrine, pour éviter en premier lieux des soumissions malicieuses permettant l'exécution de commandes Unix malveillantes en lieu et place de commandes réseaux de métrologie. Elle est cependant également assurée de notre côté lors des étapes de validation (fichier jobs/forms.py): les arguments optionnels des expériences sont refusés dès qu'ils comportent des séquences de caractères associées à des commandes linux permettant l'exécution de processus tiers (« & », « || » et « ; ») ou dès qu'ils ne respectent pas le motif d'arguments (ex : -d valeur1 -e valeur2 …) (validation par expression régulière).

La base de données est la source de vérité du service ; encore faut-il que son état soit régulièrement actualisé.

5.3. Tâche de synchronisation

L'envoi d'expériences à Planetlab et la récupération des expériences déjà soumises sont orchestrés par une instance du système de gestion de files d'attente Celery [5], conçu pour gérer de manière autonome des événements, et qui s'intègre parfaitement à Django. Ce système nécessite lui-même un système de transmission de messages, requis pour la communication au sein de Celery. Dans notre cas, Redis [6] a été adopté dans un soucis de moindre effort. Les détails de configuration sont regroupés dans le fichier metrology/settings.py, qui contient notamment la fréquence d'appel du service, ainsi que dans le fichier metrology/celery.py.

La tâche exécutée de manière récurrente se situe dans le fichier job/tasks.py. Elle consiste, à chaque appel de Celery, à :
- récupérer la liste des nodes actuellement disponibles, en vue de leur affichage dans l'interface graphique (Fig. 4),
- récupérer la liste des expériences soumises afin de mettre à jour leurs propriétés (statuts, résultats, etc.),
- envoyer la liste des nouvelles expériences et récupérer en réponse REST leurs champs hash et PLE_creation_date.

Dans un soucis de simplifier ce fichier, ces fonctionnalités ont été regroupées dans une librairies (fichier jobs/utils.py) permettant d'interroger avec une granularité adaptée l'API REST de la plateforme Planetlab tout en utilisant l'ORM intégré pour rafraîchir la base de données.

6. Retour d'expérience

1824 apprenants ont participé à ce cours, dont un quart qui a activement utilisé l'application.
Au final, 8% des inscrits ont obtenu leur attestation. Il s'agit, selon l'équipe pédagogique du Learning Lab, de la norme par rapport à d'autres MOOCs.

L'équipe technique aurait aimé avoir quelques retours des apprenants, inexistants pour le moment. Si ce projet était à refaire, il serait judicieux de demander à nos clients de fournir en priorité un moyen automatisable de connaître la stabilité de leur service. Il serait également intéressant de pouvoir pousser les expériences depuis le serveur vers les clients lorsque leurs statuts changent [7], sous réserve que cela ait du sens en terme d'expérience utilisateur d'avoir une interface auto-actualisée.

7. Remerciements

Nos collaborateurs DSI : les membres du Helpdesk INRIA et de SESI pour la mise à disposition et les services concernant le serveur.

8. Bibilographie

[1] www.fun-mooc.fr/courses/inria/41011/session01/about
[2] www.planet-lab.eu
[3] 1000hz.github.io/bootstrap-validator/
[4] www.djangoproject.org
[5] www.celeryproject.org
[6] www.redis.io
[7] www.socket.io



RESEARCH CENTRE
GRENOBLE - RHÔNE-ALPES

Inovallée
655 avenue de l'Europe - Montbonnot
38334 Saint Ismier Cedex France

Publisher

Inria

Domaine de Voluceau - Rocquencourt

BP 105 - 78153 Le Chesnay Cedex

inria.fr