Mise en œuvre de Rocket.Chat, interaction avec notre système d'information et développement collaboratif d'un plugin Moodle

Céline Pervès
Pôle Développement, Intégration & Paramétrage
Direction Du Numérique - Université de Strasbourg
cperves@unistra.fr

Benjamin Seclier
Sous direction Infrastructures
Direction du Numérique - Université de Lorraine
benjamin.seclier@univ-lorraine.fr

Résumé

Depuis le début de la crise sanitaire, nos méthodes de travail ont évolué. Le recours massif au travail à distance nous a obligé à mettre en œuvre des outils de communication inédits dans nos établissements. Rocket.Chat, outil de communication synchrone équivalent à Slack ou Discord, est ainsi devenu rapidement indispensable. Plusieurs établissements de l'enseignement supérieur français ont mis en place cet outil au même moment, des relations d'entraide et d'échanges inter-établissements se sont ainsi créées. Nous vous proposons une illustration de cette synergie.

Après avoir présenté le contexte et la mise en œuvre de Rocket.Chat au sein de l'université de Lorraine, nous présenterons les différentes intégrations que nous avons faites de cet outil au sein de nos applications (Zimbra, sites web, espaces collaboratifs, etc.).

Nous expliquerons ensuite pourquoi et comment nous avons initié une collaboration inter-établissements afin de développer ensemble un plugin Moodle permettant aux enseignants de créer un canal Rocket.Chat dont les membres sont synchronisés automatiquement avec ceux de leur cours Moodle.

Nous souhaitons, au travers de cette présentation, faire découvrir ce nouvel outil Rocket.Chat et montrer la faisabilité d'un projet inter-établissements en présentant la méthode qui nous a permis de mutualiser nos ressources afin d'atteindre un objectif commun utile pour l'ensemble de la communauté.

Mots-clefs

Moodle, plugin, Moodle, Rocket.Chat, collaboration, inter-universitaire

Table des matières

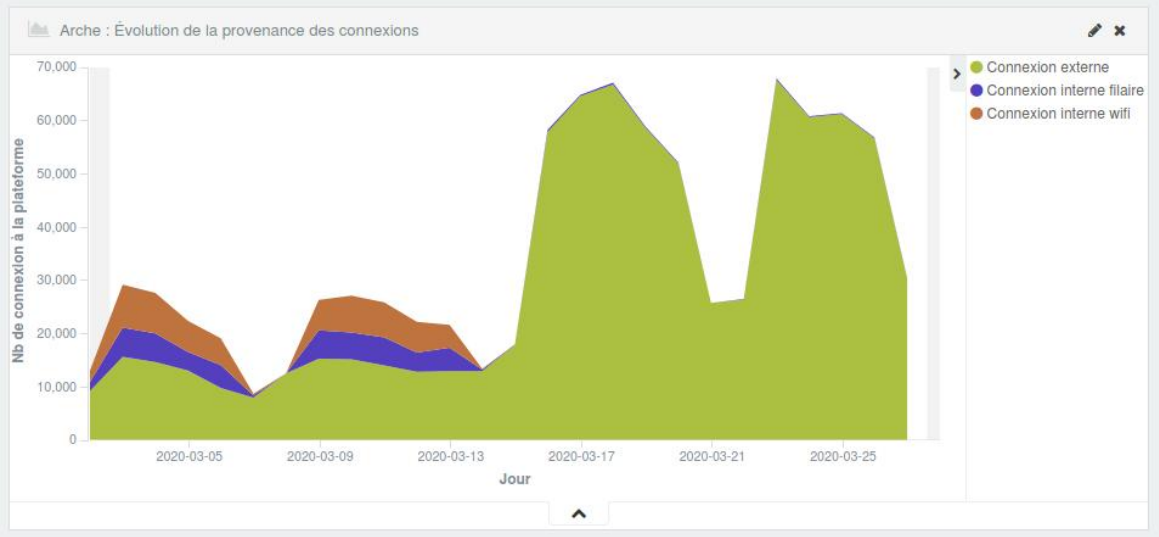
1 Mise en œuvre de Rocket.Chat à l'université de Lorraine

1.1 Des besoins dès 2017

La nécessité d'un outil de communication synchrone « chat » hébergé sur les serveurs de l'université a été remontée il y a déjà plusieurs années. En effet, dès 2017, plusieurs enseignants-chercheurs nous on fait la demande de disposer d'un outil équivalent à Slack ou Discord avec les données stockées sur nos infrastructures. Pour l'anecdote, ce sont également des enseignants-chercheurs qui nous avaient remonté, en 2014, le besoin de disposer d'un outil de « drive » équivalent à Google Drive ou Dropbox hébergé à l'université. En 2017 donc, nous avions déjà commencé à faire l'inventaire des solutions de chat open-source existantes. Nous avions fait quelques installations de test, mais nous n'étions jamais passés en production sur ce genre d'outil.

1.2 La crise sanitaire du COVID-19

En mars 2020, un confinement national a été déclaré. Ainsi, tous les cours dispensés à l'université ont été transformés en urgence en cours à distance sur notre plateforme de cours en ligne Moodle. Sans surprise, la fréquentation de cette dernière a plus que doublé du jour au lendemain, comme l'illustre ce graphique du nombre de connexions.



Au début de ce confinement, les enseignants ont massivement utilisé l'outil interne « chat » de Moodle pour rester en contact avec leurs étudiants. Malheureusement, cet outil est assez ancien et très peu optimisé. Il n'a pas supporté la charge du début de confinement et a rendu la plateforme inutilisable à plusieurs reprises. Pour faire face à ce problème, nous avons dû modifier la configuration de ce plugin afin de le rendre asynchrone. Cela revient à actualiser moins régulièrement les messages du chat dans les navigateurs des usagers. Les appels au serveur sont ainsi très réduits mais cela dégrade fortement le service, les enseignants s'en sont donc détournés. Nous avons dû mettre en place un autre système de chat en urgence, à la fois pour les enseignants, mais également pour permettre aux personnels administratifs de rester en contact, en ces temps de confinement.

1.3 Choix d’un outil de chat

Les deux critères principaux lors du choix de notre outil de chat étaient les suivants :

Open-source

Dans notre environnement professionnel, nous privilégions les outils open-source aux outils propriétaires. Dans ce cas précis, l’objectif étant de « sécuriser » en interne les conversations synchrones (privées) de nos usagers, un outil open-source semblait la solution la plus pertinente.

Disposant d’une image Docker

Nous disposons d’une infrastructure docker permettant d’accueillir de nouvelles applications. Quand une application dispose d’une image docker de bonne qualité, son installation et sa mise à jour sont grandement facilitées. Les outils de chat récents se basant sur des technologies assez complexes à implémenter, nous souhaitions absolument disposer d’une image docker afin de l’installer et le maintenir à jour le plus facilement possible.

Deux solutions remplissant ces critères étaient ainsi ressorties en 2017 :

1.3.1 Mattermost

Mattermost [1] est une solution qui date de 2015. Son installation a été simple (image docker) mais son paramétrage s’est avéré assez complexe. De plus, aucun plugin d’authentification CAS n’était disponible. Enfin, l’interface générale était assez lourde et pas très intuitive. Nous avons donc décidé de ne pas l’utiliser.

1.3.2 Rocket.Chat

Rocket.Chat [2] est une solution qui a été diffusée en open-source en 2015 également. Là encore, son installation a été très simple. Les autres avantages de Rocket.Chat ont été les suivants (liste non exhaustive) : interface très simple et très légère, administration très simple, authentification CAS intégrée, branchement sur stockage S3 intégré. La mise à jour est également très simple à faire avec l’image docker.

Pour ces raisons, à l’université de Lorraine, nous avons choisi d’implémenter Rocket.Chat en mars 2020. De plus, comme l’université de Strasbourg utilisait déjà Rocket.Chat pour les personnels avant le confinement, cela nous a conforté dans notre choix.

1.4 Lien entre Rocket.Chat et nos autres applications

Rocket.Chat est accessible depuis sa propre interface web et son client lourd. Mais, en plus, nous avons souhaité lier nos principales applications avec cet outil afin de le rendre plus visible. Vous trouverez, ci-après, un bref aperçu de ce que nous avons mis en place.

1.4.1 Zimbra

L'application la plus utilisée pour communiquer dans notre établissement est Zimbra, notre application de gestion des mails. Il nous a donc semblé important d'intégrer Rocket.Chat dans Zimbra. Ce travail a été facilité par un plugin « zimlet » déjà existant [3]. Nous avons modifié ce plugin afin de faire en sorte que son onglet clignote en cas de nouveau message et pour éviter qu'il n'ouvre automatiquement une session Rocket.Chat à chaque connexion Zimbra.

1.4.2 Espaces collaboratifs

Nous disposons d'une application « maison » permettant à nos usagers de créer un espace collaboratif. Un usager peut demander la création d'un nouvel espace et y inscrire des collaborateurs. Cela a pour effet de créer un partage dédié dans notre application Nextcloud ainsi qu'une liste de diffusion associée et désormais un canal dédié dans Rocket.Chat. Cette fonctionnalité à été possible grâce à l'API PHP *rocket-chat-rest-client [4]* que nous avons récupérée et améliorée, nous permettant de créer/supprimer des canaux, de synchroniser des utilisateurs, etc.

1.4.3 Sites web

Il est de plus en plus courant de voir une petite icône en bas à droite d'un site web, permettant d'ouvrir une fenêtre de conversation. Nous avons décidé d'implémenter cette fonctionnalité avec Rocket.Chat. Ainsi, il est possible d'afficher un pop-up Rocket.Chat branché sur un canal précis, sans avoir la liste de tous les canaux disponibles à gauche. Cela s'opère en ajoutant un paramètre GET *?layout=embedded* à la fin de l'URL du canal. Nous avons développé un script JavaScript permettant d'afficher, au clic de l'utilisateur, une petite fenêtre d'un canal bien spécifique, au sein de n'importe quelle page web.

1.4.4 Moodle

Rocket.Chat ayant été mis en place pour remplacer le chat de Moodle, nous avons rapidement dû implémenter cette fonctionnalité dans l'environnement Moodle. Ne disposant pas d'un plugin dédié, nous avions fait une documentation d'usage expliquant aux enseignants comment créer un canal public, le protéger par un mot de passe, puis créer une activité de type URL dans Moodle afin de « relier » ce canal Rocket.Chat dans leur cours en ligne. Cette fonctionnalité était loin d'être satisfaisante. En effet, quiconque disposant de l'URL et du mot de passe pouvait accéder au canal. Les comptes des étudiants inscrits au canal n'étaient pas synchronisés avec Moodle (ajouts ou suppressions). Il a donc fallu trouver une alternative à cette solution.

C'est ainsi que l'idée du développement d'un plugin Moodle dédié a émergé. Le gros avantage attendu de cette solution était d'accéder à un canal Rocket.Chat dans un cours depuis Moodle, mais également depuis l'interface web (ou tous les outils listés ci-dessus).

2 Une collaboration inter-établissements

2.1 Un besoin commun

Comme dit précédemment, nous avons eu besoin de mettre en œuvre une interaction assez forte entre les cours en ligne de notre plateforme Moodle et Rocket.Chat. Les besoins principaux étaient la création/suppression des canaux Rocket.Chat ainsi que le maintien de la synchronisation des inscriptions entre les deux plateformes. Nous savions que plusieurs établissements avaient choisi de mettre en place Rocket.Chat lors du printemps 2020 et qu'ils disposaient également d'une plateforme Moodle.

Deux solutions s'offraient à nous pour développer ce plugin :

En interne

Bien que nous ne soyons pas des développeurs de métier, nos compétences en développement nous permettent de développer un plugin en interne. Il n'aurait pas été fait dans les règles de l'art de ce qu'exige Moodle mais nous serions parvenus à développer quelque chose de fonctionnel dans un délai relativement court.

En collaboration

Nous n'avions jamais initié de travail collaboratif, à partir de zéro, entre plusieurs établissements, mais il nous a semblé intéressant d'étudier la question. En effet, nous nous sommes dit que parmi les établissements français demandeurs de ce plugin, d'autres profils d'usagers pouvaient apporter leurs expériences, notamment certains développeurs (de métier cette fois) pourraient être intéressés par l'initiative.

Pour résumer, nous avions le choix entre un développement rapide mais ne respectant pas les exigences et la rigueur demandées par Moodle, ou un développement plus long mais plus respectueux de ces dernières. Nous avons opté pour la deuxième solution.

2.2 Mise en œuvre de la collaboration

Recenser

La première étape a donc été de recenser les personnes intéressées par le sujet. Nous avons souhaité réunir un maximum de monde, qu'ils participent ou non au développement final. Pour ce faire, nous avons lancé un appel sur le forum Moodle francophone le 16 juin 2020 [5]. Suite à cela, nous avons organisé une première visioconférence regroupant environ 40 personnes. Cette réunion a permis de nous mettre d'accord sur les grandes lignes du projet. À partir de là, des personnes se sont portées volontaire pour participer activement au développement du projet, qu'il s'agisse de développer, d'apporter leur expertise technique ou pédagogique, ou encore de remonter les bugs des implémentations initiales.

Organiser le développement

Une fois que nous avons eu la liste des personnes volontaires pour travailler à l'élaboration de ce plugin, nous avons mis en place un rythme de travail d'une réunion toutes les deux semaines sur le canal dédié sur le Rocket.Chat ESUP-Portail [6]. Les premières réunions nous ont permis de nous mettre d'accord sur les fonctionnalités à implémenter et à prioriser. Les réunions suivantes nous ont permis de faire un point d'étape sur l'avancement du développement, la remontée des bugs, etc.

Développer ensemble

Nous avions besoin d'une plateforme commune pour stocker le code développé, remonter les bugs, prioriser les fonctionnalités, etc. ESUP-Portail dispose d'une organisation dans Github [7], c'est donc tout naturellement que nous nous sommes tournés vers cet outil. Plus particulièrement pour les fonctionnalités suivantes :

- hébergement de code ;
- gestion des tickets ;
- gestion du projet (kanban) ;
- wiki.

Cette organisation et ces outils de travail collaboratifs nous ont permis de mener à bien le développement de ce plugin, qui est désormais listé comme plugin officiel Moodle [8] et utilisé, en 2021, sur 94 sites

3 Fonctionnalités

De façon succincte, le plugin Moodle Rocket.Chat permet de créer, dans un cours Moodle, une (ou plusieurs) activité renvoyant vers un salon privé Rocket.Chat associé. Le plugin permet l'inscription des membres du cours en fonction de leur rôle dans le cours, à la création du module mais aussi lors des modifications et inscriptions/désinscriptions des utilisateurs. L'accès au canal Rocket.Chat associé se fait via Moodle mais est aussi possible en accès externe via un client lourd ou l'interface web de Rocket.Chat. D'autres fonctionnalités avancées sont disponibles (voir Fonctionnalités du plugin).

4 Influence de la collaboration sur les fonctionnalités du plugin

Le développement de plugin ayant été fait via une collaboration inter-établissement, il a induit un certain nombre de fonctionnalités mais aussi de contraintes techniques.

4.1 Au niveau fonctionnel

Les usages de Moodle diffèrent selon les établissements. Il en résulte des besoins différents quant aux fonctionnalités nécessaires. Les méthodes d'inscriptions aux cours Moodle de l'université de Strasbourg, ainsi que leur intégration au système d'information, ont par exemple induit des développements liés à un besoin d'exécution asynchrone en cas d'inscriptions massives.

L'anticipation des besoins de la communauté Moodle pour des entités différentes de la nôtre, comme les collèges, lycées et établissements privés, a également induit son lot de fonctionnalités supplémentaires (prise en compte de méthodes d'inscriptions différentes, utilisation du module pour tous les usagers d'une plate-forme, hébergement mutualisé).

La diversité des métiers et des points de vue au sein de l'équipe projet a permis d'évaluer de façon plus complète les besoins suivants :

- les technico-fonctionnels ont apporté une vision au plus proche des besoins et de l'expérience utilisateur ;
- les administrateurs systèmes, par leur vision plus proche de l'exploitation, nous ont permis de proposer des outils d'administration, des scripts, des automatismes et de considérer ce projet avec une vision plus orientée vers les performances ;
- enfin, les développeurs, de par leurs connaissances du développement Moodle, ont permis d'écarter ce qui n'était pas faisable, de miser sur la ré-utilisabilité ainsi que sur le paramétrage du plugin.

Nous avons fait le choix de rendre paramétrables les comportements possibles pour implémenter un maximum de possibilités. Ces différents référentiels ont permis bien souvent d'éviter des débordements et fausses routes.

Cette synergie entre les différents métiers a été un réel moteur et une véritable force tout au long de ce projet en proposant un large éventail de fonctionnalités et de possibilités.

L'actualité du sujet traité a permis de mobiliser fortement l'ensemble des personnes impliquées sur le projet et de libérer de leur temps pour participer au projet. Cette implication a permis des itérations fréquentes qui ont évité un essoufflement du projet, risque lors de projet intégrant des acteurs ayant des degrés de disponibilité différents.

4.2 Au niveau technique

Les choix techniques ont été fortement influencés par des questions d'architecture, de performances architecturales ainsi que d'un besoin important de performances et de qualité.

4.2.1 Diverses architectures et méthodes de déploiement

Il a fallu prendre en considération les différences d'intégration à nos systèmes d'information lors du développement de ce plugin comme, par exemple, la gestion des identifiants Moodle différents de ceux de Rocket.Chat (voir Hook de correspondances d'identifiants).

Une autre considération architecturale vient de la prise en compte des questions de déploiement. Certains sont en mesure de déployer le code directement sur leurs serveurs, qu'ils soient au sein de leurs entités ou chez des hébergeurs. D'autres dépendent d'hébergeurs Moodle qui déploient le code officiellement publié chez Moodle. Dans ce cas, le choix du respect du framework de code Moodle s'impose. Cependant l'application de patchs de code Moodle devient dans ce cas difficile et un fonctionnement du plugin sans patch de code, en mode dégradé, a été pris en charge.

Nous avons soumis nos propositions de patchs à Moodle HQ (Headquarters)[1] par le biais de tickets (voir [9] et [10]) pour qu'ils soient intégrés au cœur Moodle.

À noter que ces patchs correspondent à des points techniques non pris en charge par Moodle et sont les moins intrusifs et impactants possibles.

4.2.2 Performances

La plupart d'entre nous ont des Moodle dits « de masse », c'est-à-dire contenant un grand nombre d'utilisateurs distribués ou non en cohortes (sorte de groupes Moodle) parfois très nombreuses et volumineuses. À l'inscription des utilisateurs aux salons Rocket.Chat rattachés, on observait initialement une attente problématique de la finalisation des pages de création de salons ou d'inscriptions d'utilisateurs. Ce problème a été réglé en mettant en place un système d'inscriptions asynchrones (voir Implémentation du lancement des tâches d'inscription arrière-plan).

Nous avons fait en sorte de laisser toute latitude à l'administrateur d'opter pour un comportement asynchrone ou non.

1 Le projet du Moodle est actuellement coordonné par le Moodle HQ, une entreprise australienne de 50 développeurs financée par un réseau de 84 partenaires Moodle, entreprises de services du Moodle du monde entier. Le développement du Moodle est également soutenu par le travail de programmeurs de logiciels libres

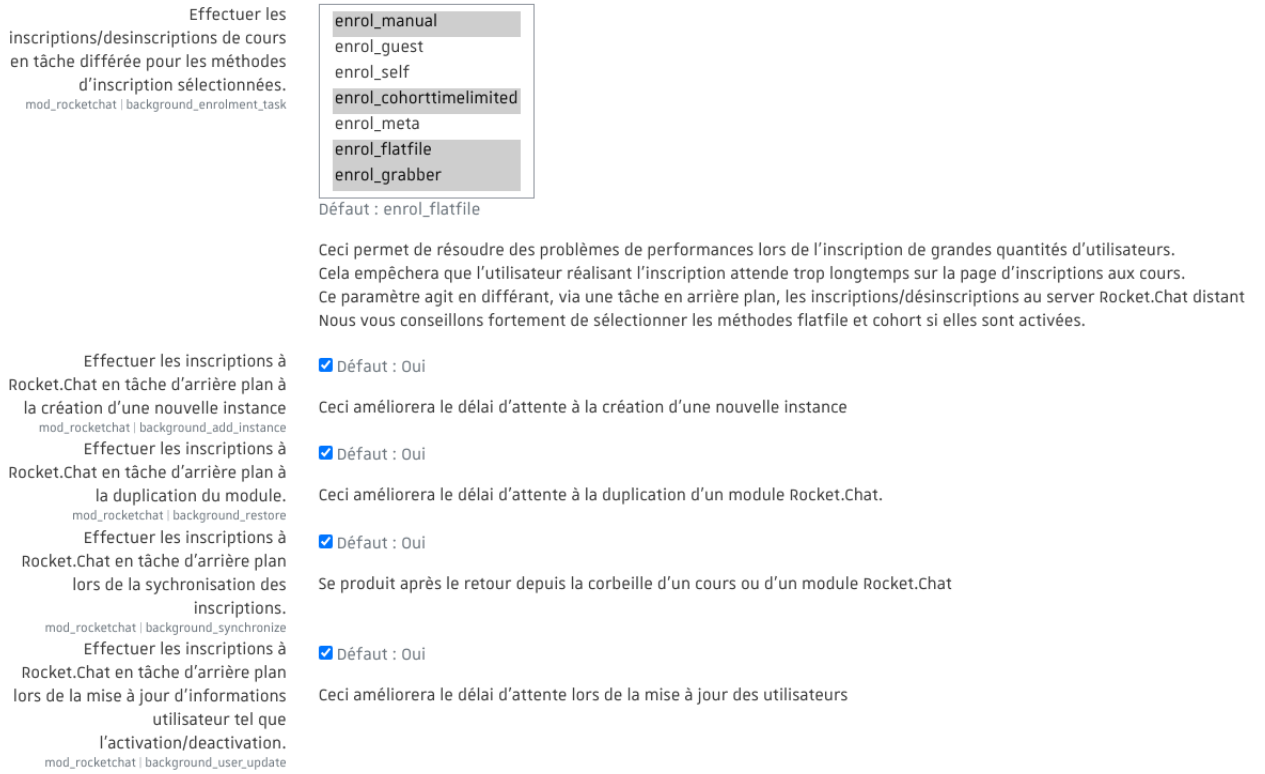


Figure 1: Paramétrage des tâches différées

4.2.3 Qualité

Travailler pour les autres, pour la communauté, nous a obligé à faire le choix d'investir dans la qualité. De plus Moodle exige un niveau élevé de qualité avant d'approuver un plugin. La publication officielle des plugins Moodle qui en résulte est, chez la plupart d'entre nous, un prérequis à leur installation. L'approbation de cette publication passe par une validation automatique du code par différents bots. Pour passer cette étape, il est nécessaire de respecter le *code styling* et le framework Moodle et d'éviter un certain nombres d'erreurs qui pourraient mettre en péril la sécurité de Moodle. Le code passe ensuite par une relecture et une revue de code par des membres de Moodle Quality Assurance testing (ensemble de testeurs désignés par Moodle HQ), ce qui permet alors de proposer un code plus sûr et plus performant.

À noter que respecter ces points permettra par la suite à tout développeur Moodle, HQ ou non, de relire facilement le code du plugin, de corriger et participer à son développement.

La qualité passe également par l'investissement dans des codes unitaires et fonctionnels. Bien qu'optionnels dans l'approbation du plugin, ils sont un plus dans le process, mais ils ont surtout un apport interne dans le cycle de vie du plugin. Toute nouvelle fonctionnalité peut ainsi être validée en évitant les risques de régression. Ces tests facilitent aussi la participation de développeurs externes. Il en va de même pour tout changement dans la version du logiciel tiers Rocket.Chat.

L'investissement de départ peut paraître coûteux, mais proposer un code stable dont on n'aura pas besoin de corriger les bugs dans l'urgence, est un vrai gain de temps.

Dernier point de qualité, nous avons respecté le Règlement Général sur la Protection des Données (RGPD). Optionnel à l'origine dans le process d'approbation des plugins, il est depuis peu devenu obligatoire. Moodle propose dans son framework de quoi exposer, exporter et supprimer ou anonymiser les données. La simple implémentation des classes adéquates permet de mettre en place le RGPD dans un plugin.

4.3 Apports de cette collaboration

Notre collaboration a fortement influencé les fonctionnalités et les aspects techniques du plugin Moodle lui procurant une grande richesse, mais ces choix ont induit des contraintes de temps, de travaux supplémentaires.

Cependant d'autres apports bénéfiques ont pu être observés :

- au sein des établissements participants au projet nous avons pu proposer un plugin complet à leurs usagers incluant des fonctionnalités auxquelles nous n'aurions pas pensé si nous avions développé le plugin individuellement (par exemple : outils systèmes, usages différents, etc.) ;
- en présentant régulièrement au groupe de travail l'avancement de nos travaux, nous avons pu bénéficier des suggestions du groupe ESUP-Portail. Des fonctionnalités telles que la prise en compte des identifiants différents entre Moodle et Rocket.Chat illustrent bien ce fait ;
- le groupe ainsi que la communication plus large avec la communauté Moodle via les forums entre autres, nous a aussi permis de bénéficier d'un lot de testeurs variés, nombreux et du coup très pertinent. Notre plugin a ainsi acquis en stabilité et robustesse.

Ce projet nous a aussi beaucoup apporté d'une façon plus générale au quotidien dans nos métiers. Il nous a permis de découvrir d'autres visions – administrateur système, expériences utilisateurs diverses et possibilités techniques – qui nous seront utiles lors de futurs développements Moodle mais aussi dans le cadre d'autres projets.

Nous avons donc beaucoup appris fonctionnellement et techniquement mais surtout les uns des autres.

5 Conclusion

Lors de la crise sanitaire, Rocket.Chat s'est avéré être une bonne solution : cet outil est simple à gérer et agréable à utiliser. Son API complète nous a permis de l'intégrer dans plusieurs de nos applications, multipliant ainsi sa visibilité et apportant un lien supplémentaire entre ces différentes applications.

La collaboration que nous avons mise en place pour le développement d'un plugin Moodle a été bénéfique à différents niveaux. Cela nous a, en effet, permis de réunir plusieurs établissements pour un travail commun. Ce travail a été fructueux grâce à la bonne volonté de ses participants et aux outils de travail collaboratifs à notre disposition.

Un problème est cependant apparu au mois de juillet 2021 : Rocket.Chat a changé son modèle économique. Là où, auparavant, toutes les fonctionnalités étaient intégrées gratuitement dans Rocket.Chat, il y a désormais une version « enterprise » en plus d'une version communautaire. Certaines fonctionnalités autrefois gratuites sont devenues payantes. Parmi ces fonctionnalités, il y a par exemple le *provisionning* LDAP utilisé dans certains établissements. Au vu du nombre d'étudiants, le coût de revient de la solution « enterprise » est beaucoup trop élevé pour les établissements d'enseignement supérieur français. Des négociations sont en cours avec Rocket.Chat, mais certains établissements se sont déjà tournés vers d'autres outils. Ce précédent nous force à nous poser la question de la pérennité de la gratuité des fonctionnalités utilisées actuellement. C'est la raison pour laquelle nous sommes en train d'étudier la faisabilité de l'adaptation du plugin existant pour le brancher sur d'autres applications de chat, comme Mattermost ou Element/Matrix.

6 ANNEXES

Fonctionnalités du plugin

Fonctionnalités principales :

- module d'activité ;
- création d'un canal/groupe/salon privé Rocket.Chat par instance d'activité ;
- inscription des membres du cours au canal associé ;
- accès au canal depuis Moodle ou directement via Rocket.Chat.

Inscription et synchronisation des utilisateurs du cours dans le canal associé en fonction du rôle dans le cours :

- prise en charge de l'attribution de rôle au niveau de l'activité ;
- possibilité de créer le compte utilisateur sur Rocket.Chat ;
- possibilité d'utiliser une fonction de correspondance entre les *username* de moodle et ceux de Rocket.Chat.

Gestion de l'affichage :

- gestion de l'affichage conventionne ;
- choix du mode *embedded*.

Mise en corbeille, restauration et suppression définitive :

- en corbeille : le canal passe en *read-only* ;
- restauration : le canal repasse en écriture ;
- suppression : le canal est supprimé.

Restauration de l'activité :

- sauvegarde/restauration de l'activité : nouveau canal côté Rocket.Chat ;
- duplication de l'activité : nouveau canal côté Rocket.Chat.

Rétention des messages :

- surcharge du *maxage* (temps de conservation des messages) ;
- lancement des tâches d'inscription arrière-plan ;

Possibilité de lancer les inscriptions et les synchronisations d'inscriptions en arrière plan (ad_hoc task) :

- au niveau des inscriptions au cours ;
- choix des méthodes concernées ;
- au niveau de la corbeille ;
- au niveau de la restauration ;
- l'utilisateur courant est toujours inscrit en synchrone ;

Outils d’administrations :

- visualisation des données Rocket.Chat des salons créés depuis Moodle ;
- synchronisation des inscriptions Moodle/Rocket.Chat ;
- recréation d’un canal absent.

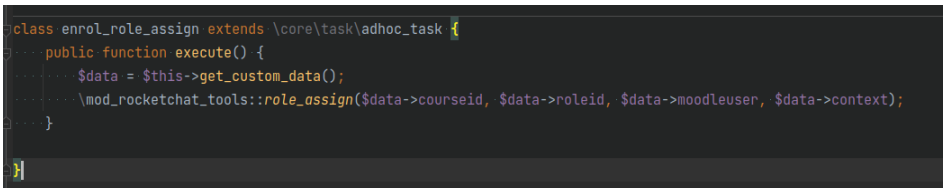
Implémentation du lancement des tâches d'inscription arrière-plan

Lors de l'inscription en masse d'utilisateurs au cours, une demande d'inscription en masse à Rocket.Chat est à effectuer. Rocket.Chat propose, dans son API de webservices, uniquement des inscriptions unitaires. Lancer en boucle un grand nombre d'inscriptions au sein de plugin Moodle s'avère long et engendre un temps d'attente déstabilisant pour l'utilisateur.

Par défaut PHP ne gère le multi-threading que par l'installation d'extensions et l'emploi d'une librairie PHP le permettant (recompilation nécessaire). L'emploi de queues/messages pourrait aussi être une solution mais nécessiterait l'emploi de bibliothèques supplémentaires et pourrait s'avérer un peu trop lourde dans notre cas. Comme la solution devait s'adresser aussi à des entités hébergeant leur Moodle, nous avons écarté l'emploi de telles solutions.

Moodle propose un sorte de micro tâches qui permettent de palier cela : les adhoc tasks ([9] et). En encapsulant les inscriptions atomiques dans des objets adhoc_task, nous déléguons ainsi leur exécution en asynchrone au système de tâches/cron Moodle. L'appel lancé, la tâche créée et stockée en base de données pour exécution ultérieure au lancement des cron Moodle.

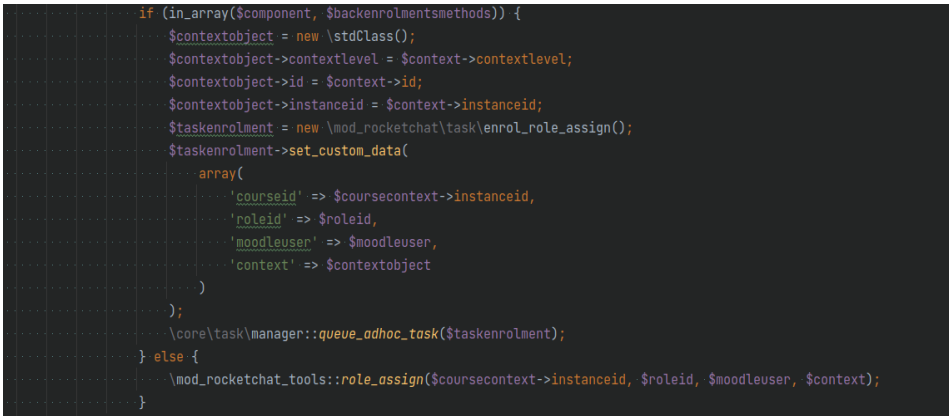
Le process en cours s'exécute alors beaucoup plus rapidement. Pour sa part, l'enseignant est tout de même inscrit immédiatement pour pouvoir tester dès la création de l'instance.



```php
class enrol_role_assign extends \core\task\adhoc_task {
    public function execute() {
        $data = $this->get_custom_data();
        \mod_rocketchat_tools::role_assign($data->courseid, $data->roleid, $data->moodleuser, $data->context);
    }

}
```

Figure 2: Définition d'une adhoc_task



```php
if (in_array($component, $backenrolmentsmethods)) {
    $contextobject = new \stdClass();
    $contextobject->contextlevel = $context->contextlevel;
    $contextobject->id = $context->id;
    $contextobject->instanceid = $context->instanceid;
    $taskenrolment = new \mod_rocketchat\task\enrol_role_assign();
    $taskenrolment->set_custom_data(
        array(
            'courseid' => $coursecontext->instanceid,
            'roleid' => $roleid,
            'moodleuser' => $moodleuser,
            'context' => $contextobject
        )
    );
    \core\task\manager::queue_adhoc_task($taskenrolment);
} else {
    \mod_rocketchat_tools::role_assign($coursecontext->instanceid, $roleid, $moodleuser, $context);
}
```

Figure 3: instanciation de la tâche

Hook de correspondances d'identifiants

Pour gérer l'éventuelle différence d'identifiants entre Moodle et Rocket.Chat, nous avons opté pour le développement d'un système de hook. Celui-ci permet à un établissement d'ajouter, dans un fichier, une fonction de transformation des identifiants Moodle vers ceux de Rocket.Chat et après détection par Moodle, d'activer ce hook via l'interface d'administration du plugin.

```php
function moodle_username_to_rocketchat($moodleusername) {
    return preg_replace(
        '/[^0-9a-zA-Z-_.]/', '__',
        preg_replace('/@univ-paris1[.]fr$/', '', $moodleusername));
}
```

Figure 4: Exemple de code

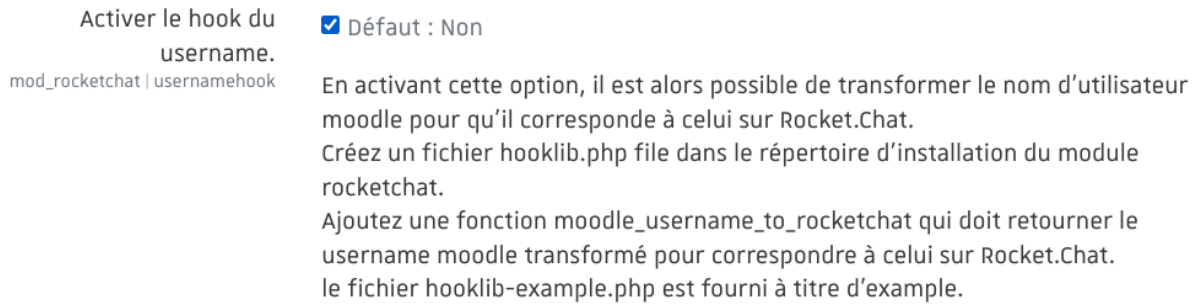


Figure 5: Paramétrage via l'interface d'administration Moodle

```php
public static function rocketchat_username($moodleusername) {
    global $CFG;
    $hook = get_config( plugin: 'mod_rocketchat', name: 'usernamehook');
    if ($hook) {
        require_once($CFG->dirroot.'/mod/rocketchat/hooklib.php');
        return moodle_username_to_rocketchat($moodleusername);
    }
    return $moodleusername;
}
```

Figure 6: Appel au hook

Références

[1] Site web de Mattermost
[2] Site web de Rocket.Chat
[3] zimlet Rocket.Chat
[4] API PHP rocket-chat-rest-client
[5] Forum Moodle francophone le 16 juin 2020
[6] Canal dédié sur le Rocket.Chat ESUP-Portail
[7] ESUP-Portail dans Github
[8] Page officielle du plugin Rocket.Chat sur moodle.org
[9] Patchs Moodle corbeille pour la restauration
[10] Patch corrigeant un bug d'inscriptions/désinscriptions
[11] Adhoc tasks