Accompagner l'enseignement de l'informatique avec Caseine

Astor Bizard, Denis Bouhineau, Nadia Brauner, Hadrien Cambazard, Nicolas Catusse, Mario Cortes Cornax, Aurélie Lagoutte, Pierre Lemaire, Yvan Maillot, Florence Thiard

Accompagner l'enseignement de l'informatique avec Caseine

Astor Bizard, Denis Bouhineau, Nadia Brauner, Hadrien Cambazard, Nicolas Catusse, Mario Cortes Cornax, Aurélie Lagoutte, Pierre Lemaire, Yvan Maillot, Florence Thiard

C'est quoi cette bouteille de lait ?

Caseine avec son logo ci-contre est à la fois une plateforme d'apprentissage en informatique, mathématiques et génie industriel, et une communauté d'enseignants rassemblés autour de l'objectif commun de favoriser l'apprentissage en développant l'autonomie des étudiants et en améliorant la qualité du temps que l'enseignant leur consacre [1]. De sa création à aujourd'hui et au travers de ses évolutions, Caseine s'est toujours appuyée sur 3 piliers :

— des outils pour fournir un retour automatisé sur les productions des apprenants (code informatique, modèles mathématiques [1], ainsi que d'autres objets computationnels comme des circuits logiques, des jeux de test [2]...) ;

— un environnement permettant le suivi pédagogique des étudiants (Caseine est une instance de Moodle) ;

— le partage des pratiques et des contenus entre les enseignants, à travers des outils (espace de partage, visite de cours), des contenus et une communauté vivante d'utilisateurs (événements et outils de communication).

1. Un tour de la plateforme en 3 minutes : https://caseine.org/mod/page/view.php?id=19571.

DOI:10.48556/SIF.1024.20.107 **1024** – Bulletin de la SiF, numéro 20, Novembre 2022, pp. 107–122

La plateforme, hébergée aujourd'hui par le *datacenter* universitaire Grenoblois, est ouverte à tous et en particulier aux établissements d'enseignements supérieurs français. Elle a été utilisée par plus de 9 000 personnes pendant l'année universitaire 2021-2022, provenant essentiellement de 13 universités différentes (dont deux à l'international) et plus de 10 lycées répartis sur l'ensemble du territoire. Certaines institutions ont généralisé son usage (l'université Grenoble Alpes et Grenoble INP, à l'origine du projet, mais aussi l'université Clermont Auvergne), d'autres l'utilisent pour une partie de leurs enseignements (université de Haute Alsace, université de Bordeaux, certains lycées, etc.). Elle héberge également des projets universitaires, qui contribuent volontairement à son financement.

Est-ce différent de Moodle ?

Ni oui, ni non. Caseine est une instance de l'environnement numérique d'apprentissage Moodle, comme il en existe dans de nombreuses universités ; mais à la différence de beaucoup, elle n'est pas exclusivement adossée à une institution spécifique. Via Renater et plus largement la fédération EduGain, toutes les personnes disposant d'un compte universitaire français, en particulier, peuvent s'y connecter. Les enseignants d'établissements publics peuvent demander l'ouverture d'un ou plusieurs espaces de cours, aussi bien pour expérimenter, partager, que pour y accueillir leurs étudiants.

Certains cours sont en accès libre, en tant qu'invité ou sur auto-inscription. Une part de ceux-ci, mis en avant en tant que « cours ouverts » n'ont pas vocation à accueillir directement des promotions d'étudiants : ces espaces servent à l'autoformation (d'initiative individuelle ou en complément d'un cours), mais aussi à donner aux enseignants et enseignantes intéressés une idée des possibilités de la plateforme, les ressources qu'ils contiennent étant placées sous une licence autorisant leur réutilisation (généralement CC-BY-SA). On y trouve aussi bien des cours « clés en main » qui peuvent être dupliqués et utilisés comme base de nouveaux cours (initiation à la programmation en python, recherche opérationnelle, programmation objet...) que des espaces à vocation de démonstration (jeu de piste SQL, méthodes de test logiciel) ou de catalogue de ressources (vitrine pour l'enseignement de l'informatique en lycée). Les autres cours présents sur la plateforme, qui peuvent être en accès libre ou restreint, sont généralement gérés par des enseignants et des enseignantes et destinés à être utilisés dans le cadre de leurs enseignements.

Moodle offre aux enseignants un environnement complet pour assurer le suivi pédagogique des étudiants et suffisamment riche et souple pour scénariser finement leurs cours pour différents usages (cours complet ou ressources de compléments, préparation, entraînement ou approfondissement, en autonomie ou dans le cadre d'un

enseignement hybride...). Et surtout, cet environnement s'appuie sur une large communauté dynamique, et le cœur de la plateforme est largement extensible par des modules (types d'activités, outils de suivi...) développés par des institutions ou utilisateurs et publiés via le marché d'application Moodle. Côté Caseine, selon les besoins exprimés et discutés entre les enseignants, l'équipe sélectionne ceux qui s'avèrent pertinents. Si une fonctionnalité n'existe pas, le développement d'un module ad-hoc peut être envisagé, avec l'objectif de publier sur le Moodle Store les fonctionnalités généralisables et susceptibles de répondre aux besoins d'autres institutions. Les modules suivants ont ainsi été publiés par l'équipe Caseine [2] :

— le bloc « Point de vue », un outil permettant de recueillir sous forme de « réactions » le point de vue des étudiants sur la difficulté des activités proposées ;
— l'activité *Random Activity* — développée pendant le confinement — permettant d'attribuer aléatoirement des activités aux étudiants d'un cours (utile par exemple dans le cadre d'examens) ;
— les questions de programmation « Questions VPL » permettant d'intégrer de courtes questions de programmation dans des quiz Moodle ;
— le comportement de questions *student feedback* permettant de recueillir des commentaires étudiants sur les questions d'un test Moodle.

Des outils de retour automatique et d'aide à l'évaluation

Un des modules installés sur Caseine occupe une place particulière. Intégré dès les origines du projet, il représente une grosse partie des activités existantes sur la plateforme. Il s'agit du *Virtual Programming Lab* (VPL) (cf.figure 1), développé et maintenu principalement par Juan Carlos Rodriguez-del-Pino, de l'université de Las Palmas (Canaries). Ce module permet de réaliser des activités de programmation complètes sur Moodle, incluant environnement d'édition et d'exécution, ainsi que des outils de retour et d'aide à l'évaluation automatisée. La version présente sur Caseine diffère légèrement de la version canonique publiée sur le marché Moodle : nous y apportons des corrections et des « améliorations » (de notre point de vue !) ; toujours dans le même esprit, nous remontons et proposons l'intégration des fonctionnalités susceptibles d'intéresser un public plus large.

À quoi ressemble une activité VPL du point de vue de l'étudiant ? L'édition de code (éventuellement à partir d'un squelette fourni) s'effectue dans la partie centrale. Le bouton « fusée » permet d'exécuter le programme (en environnement textuel ou graphique selon l'activité). Le bouton « coche » permet d'évaluer le travail (en lançant les tests prévus par l'enseignant). Le rapport d'évaluation (résultats de tests,

2. `https://moodle.org/plugins/?q=caseine`.

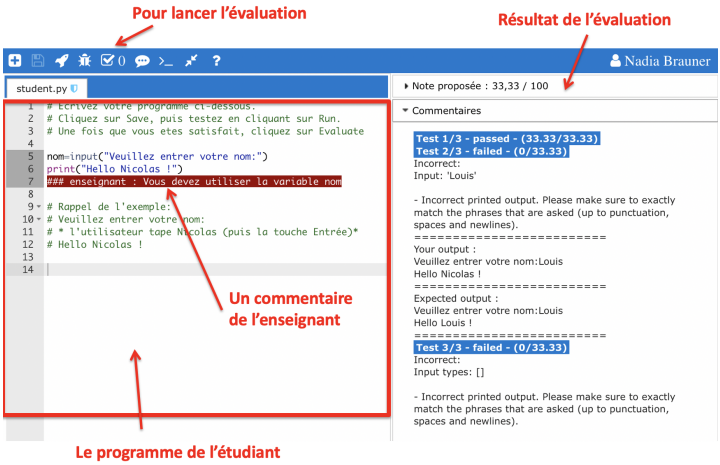


FIGURE 1. Une activité VPL du point de vue de l'étudiant.

commentaires sur les erreurs éventuelles...) apparaît alors dans le volet de droite, associé éventuellement à une note indicative.

L'enseignant, lui, a accès à toutes les soumissions de ses étudiants, ainsi qu'aux rapports d'évaluation et aux notes automatiques attribuées le cas échéant. Il peut évaluer les travaux, les modifier, les commenter, ajuster la note... Il a également accès à une série de statistiques, ainsi qu'à un outil de détection de similarité, permettant de repérer les productions d'étudiants potentiellement trop proches les unes des autres. Remarquons que chaque action « enregistrer », « exécuter » et « évaluer » de l'étudiant donne lieu à une soumission, visible par l'enseignant. Cela présente des avantages, permettant par exemple une aide « en direct » de l'enseignant (qui peut par exemple ajouter des commentaires spéciaux dans le code, que l'étudiant retrouvera dans son interface).

Comment les résultats d'évaluations automatiques sont-ils produits ? Le principe de fonctionnement du VPL est au fond très simple : lorsque l'étudiant lance une action « évaluer », l'ensemble des fichiers fournis par l'étudiant ainsi que certains fichiers fournis par l'enseignant sont envoyés sur le serveur d'exécution (un environnement GNU/Linux classique). La séquence d'évaluation (compilation, exécution des tests...) s'exécute alors dans cet environnement, produisant une sortie formatée, à partir de laquelle seront extraits le rapport d'évaluation à présenter à l'étudiant et la note obtenue. Même principe général pour une action « exécuter », sauf que la sortie produite est alors directement affichée dans la console du VPL.

Comment sont définies ces séquences d'exécution et d'évaluation ? Si l'enseignant n'a pas de besoin particulier et qu'il ne précise rien, le VPL propose un comportement par défaut. Pour l'exécution, il s'agira de compiler les sources et exécuter le premier fichier soumis par l'étudiant, et pour l'évaluation, d'un mécanisme type « boîte noire » : la sortie standard du programme étudiant est comparée avec une sortie attendue, pour une série d'entrées spécifiées par l'enseignant. Celui-ci doit alors simplement décrire une série de cas-tests (entrée, sortie attendue, message en cas d'échec...).

Si ce fonctionnement par défaut a pour avantage la simplicité et une relative indifférence vis-à-vis du langage de programmation, il peut en revanche ne pas suffire à l'usage envisagé (évaluations fonctionnelles, tests unitaires, évaluation de la complexité...). Il est alors possible d'écrire ses propres scripts (bash ou autre) et programmes d'exécution et d'évaluation : un script, un environnement complet ; la seule limite est l'imagination, ou presque ; on peut d'ailleurs tout à fait envisager des activités allant au-delà de la programmation au sens strict (calcul formel, statistiques...).

Ce dernier mode d'utilisation conviendra particulièrement, au moins dans un premier temps, aux enseignants rodés à l'évaluation semi-automatique, qui ont peut-être déjà leur propre « moulinette » (qu'ils souhaiteraient rendre plus accessible aux étudiants ou à leurs collègues), où à ceux qui ont une idée précise de ce qu'ils souhaitent faire et de la façon de le réaliser. Il demande toutefois un investissement conséquent en temps et en énergie. Le VPL prévoit heureusement un mécanisme d'héritage entre activités, permettant de partager les fichiers qui les composent (et en particulier les scripts d'évaluation). Ce mécanisme est mis à profit sur Caseine, en permettant d'utiliser directement et facilement des scripts d'évaluation développés par l'équipe ou la communauté et mis à disposition de tous (il suffit de choisir le mécanisme à utiliser lors de la création de l'activité, parmi les bases disponibles). Par exemple, on peut facilement créer des activités en Java ou Python en utilisant un framework de test unitaire (JUnit, python-unittest) pour spécifier l'évaluation, évaluer empiriquement la complexité des programmes étudiants, évaluer des requêtes SQL, ou encore des programmes étudiants modifiant l'environnement...

L'interface web de la figure 1 à l'avantage d'être simple, facilement accessible par les étudiants, sans installation nécessaire, et de les plonger directement au cœur de l'exercice. En contrepartie, elle déconnecte l'élève de l'environnement de développement et d'exécution sous-jacent à la programmation elle-même, et ne lui permet donc pas d'acquérir une pleine autonomie ni la maîtrise de ses outils. Elle est également peu commode pour la gestion de projets conséquents, et à l'autre extrémité du spectre, peut s'avérer trop bruyante pour des exercices « courts » (du moins dans leur expression). Il est bon tout d'abord de rappeler que le VPL est un outil d'aide à l'enseignement, avec ses limites, et a vocation à s'intégrer dans une scénarisation complète, mêlé à d'autres contenus, activités et modalités d'interactions. Toutefois, dans le cadre du projet Caseine, des interfaces alternatives ont été conçues qui peuvent

en élargir le périmètre d'usage. D'un côté, le module « questions VPL » permet de proposer une interface encore plus légère et épurée à une activité VPL, sous la forme d'une question de quiz Moodle. Cette forme est particulièrement adaptée aux exercices à réponses concises (quelques lignes de code, une requête SQL, une fonction mathématique...) ou qui s'intègrent dans un tout (un quiz Moodle). De l'autre, des plugins pour différents IDE (Eclipse, VSCode, IntelliJ...) permettent à l'étudiant de travailler (ainsi que compiler, exécuter...) en local dans un environnement de développement complet, et de soumettre et récupérer le rapport d'évaluation depuis leur IDE (voir section théorie des graphes pour une description plus détaillée et un exemple de contexte d'utilisation).

Une communauté d'enseignants et un espace de partage et d'expérimentation !

Un objectif majeur de Caseine dès sa création était de permettre le partage et la mutualisation, au sein d'une équipe pédagogique d'abord, puis plus largement d'une communauté d'enseignants, de ressources, de contenus mais aussi de bonnes pratiques et d'outils techniques. La plateforme se prête, de façon perméable, à toute une gamme d'usages s'inscrivant dans cet esprit de mutualisation, du terrain de jeu (mais pas seulement) individuel à l'expérimentation collective, à l'utilisation plus routinière de ressources éprouvées.

Par exemple, dans ses espaces de cours, un enseignant peut élaborer et essayer des outils d'évaluation qui conviennent à ses besoins, affiner ces outils et rôder des activités au fil des promotions et des retours étudiants, puis une fois ceux-ci arrivés à une maturité qu'il jugera satisfaisante, les partager avec la communauté : c'est en particulier ainsi qu'ont été élaborés certains mécanismes d'évaluation pour le VPL aujourd'hui disponibles pour tous ou les outils permettant l'utilisation de Caseine depuis différents IDE.

Dans tous les cas, l'appui de l'équipe Caseine peut également être sollicité pour, selon les cas, initier, accompagner ou stabiliser un développement, voire le prendre en charge entièrement.

Pour ce qui est des contenus eux-mêmes, deux modalités coexistent actuellement : le partage individuel d'activités (éléments du cours), ou la proposition d'un cours *ouvert*. Il est également possible de rendre son cours *visitable*, c'est à dire d'ouvrir un accès partiel aux enseignants de la communauté pour qu'ils puissent consulter (et récupérer) les activités partagées directement dans leur contexte (cf. figure 2). Précisons que dans tous les cas, la démarche repose sur une décision volontaire de l'enseignant auteur : rien ne l'oblige à ouvrir ses contenus au partage.

Ce partage de ressources constitue un attrait important pour les enseignants, d'autant que le coût d'entrée technique et l'investissement en temps nécessaire pour exploiter pleinement les possibilités de la plateforme et produire des activités est

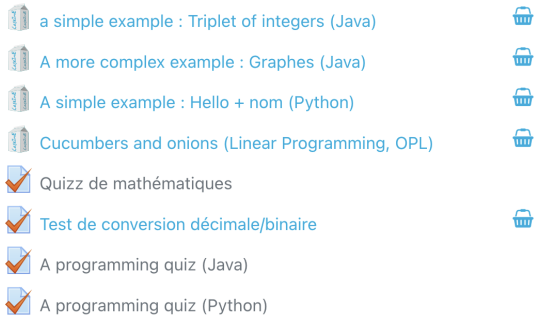


FIGURE 2. Exemple d'une section de cours visitable : les activités partagées peuvent être ajoutées au panier du visiteur qui peut ensuite les insérer dans son cours.

loin d'être négligeable. D'un côté, les nouveaux arrivants peuvent s'appuyer sur des contenus existants pour se mettre le pied à l'étrier, que ce soit en les intégrant dans leurs enseignements, en les utilisant comme base pour construire leur espace ou simplement comme exemple pour adapter leurs propres ressources. Pour les utilisateurs plus expérimentés, au-delà du gain de temps apporté par le réemploi de ressources, la perspective du partage ajoute un sens supplémentaire aux efforts de conception (on ne travaille plus uniquement pour soi et ses élèves, mais si on le souhaite, pour le bénéfice de la communauté) et conduit à terme à une amélioration de la qualité des ressources produites, via les relectures et retours des enseignants intéressés.

Toutefois, la mise en œuvre pratique est encore largement perfectible et en mutation ; elle est au centre de réflexions en cours pour, entre autres, affiner les outils, rendre leurs usages plus fluides, identifier la (ou plus sûrement les) granularité adéquate pour les ressources partagées, faciliter les retours et la collaboration.

Enfin, partager, c'est avant tout échanger, au-delà des outils et des modalités pratiques. Caseine est aujourd'hui aussi bien une plateforme qu'une communauté active d'enseignants qui partagent au sujet de leurs disciplines et leurs pratiques, au-delà du cadre de leurs équipe pédagogiques et institutions respectives.

La crise sanitaire de 2020 a marqué un tournant dans la fédération de cette communauté. En posant une série de défis techniques et pédagogiques (enseignement hybride, asynchrone, maintien de l'engagement des étudiants, évaluations à distance...) qui ont fortement sollicité la plateforme, les confinements et leurs suites ont été un déclencheur de discussions plus larges autour des usages numériques pour la pédagogie. Le chat (Mattermost) de la communauté Caseine, déployé à cette occasion pour leur offrir un espace, permet aujourd'hui d'échanger tant avec l'équipe qu'avec la communauté. D'un côté très pratique, il donne accès à un support technique collaboratif (de la part des ingénieurs Caseine mais aussi des utilisateurs experts), favorise

l'identification des besoins, le partage de compétences et l'émergence des nouvelles fonctionnalités ; mais il héberge également des discussions plus générales autour de la pédagogie, ainsi que des espaces de travail d'équipes pédagogique ou de projets hébergés sur la plateforme via des canaux thématiques publics ou privés (Ocaml, C, pédagogie, SQL...). Des webinaires « demi-heures Caseine », également mis en place au printemps 2020 sont aujourd'hui un rendez-vous semi-régulier, par visio-conférence, constitué d'un exposé suivi d'une discussion sur un sujet lié à Caseine... De près ou de loin ; il peut s'agir aussi bien d'une présentation d'une nouvelle fonctionnalité par l'équipe (« Le partage pour les nuls ou rendre son cours visitable »), d'un outil développé par un membre de la communauté (« Le plugin de conception C/C++ »), d'enseignant décrivant leurs usages (« Ludification sur caseine »), de sujets pédagogiques plus larges (« QCM augmentés », « évaluation par les pairs »)... Ces échanges sont enregistrées et rendus disponibles pour les enseignants, servant ainsi de base de connaissances.

Un commun numérique de fait

En fin de compte, depuis sa genèse et avant même la loi pour une république numérique en 2016, Caseine a inscrit la mutualisation — des ressources, des outils, des pratiques — au cœur de ses activités, ce qui améliore la qualité et l'efficacité tout en ajoutant du sens et du plaisir au travail fait. De ce fait, Caseine, en tant que projet, plateforme et communauté, s'inscrit naturellement dans la dynamique des communs numériques tels qu'ils sont maintenant définis et recommandés dans des textes officiels [3] [4]. Ainsi, le deuxième plan national de 2021 pour la science ouverte [5] stipule d'« *engager les acteurs de l'enseignement supérieur et de la recherche dans un travail commun sur les ressources éducatives libres pour les rendre plus visibles, mieux les partager et favoriser leur réutilisation* » ; c'est ce que s'efforce de faire Caseine depuis 7 ans.

Exemples d'utilisation de Caseine

Plus de 200 cours d'informatique ont été déposés sur la plateforme. La plupart sont de niveau universitaire, mais la plateforme accueille aussi des enseignements de niveau lycée (NSI, algorithmique). Issus des cours universitaires institutionnels, ils remplacent les sites institutionnels, tiennent lieu de site officiel pour des enseignements ou sont adossés à ces sites institutionnels. Le choix de cette plateforme externe peut étonner dans le cas où les plateformes institutionnelles existent, alors même que

3. https://labo.societenumerique.gouv.fr/2019/10/16/les-communs-numeriques-un-modele-innovant-de-developpement-des-ressources-numeriques.

4. https://communs.societenumerique.gouv.fr.

5. https://www.enseignementsup-recherche.gouv.fr/sites/default/files/2021-09/2e-plan-national-pour-la-science-ouverte-12968.pdf.

la multiplication des plateformes est régulièrement critiquée par les étudiants. Plusieurs explications de ce choix d'une plateforme externe à l'institution, stricto sensu, peuvent être données au vu des cours concernés.

Une part significative des cours disponibles sur Caseine sont en libre accès : soit ouverts aux anonymes, soit accessibles en auto-inscription, ce qui est plutôt l'exception sur les plateformes institutionnelles. Ce peut être une explication du choix de Caseine : la mise à disposition de contenus pour un plus grand nombre d'étudiants, le choix pour une plateforme plus orientée vers le partage entre enseignants, pour bénéficier des outils spécifiques mis en place pour favoriser l'ouverture et les échanges. Toutefois, l'ouverture et le partage ont des limites, les raisons du choix de Caseine ne peuvent s'y limiter.

Les cours disponibles sur Caseine sont rarement des cours « simples » (polycopiés et fiches d'exercices déposés sous forme de fichiers pdf, forum et une zone de rendu). Quasiment tous les cours d'informatique contiennent des activités de type VPL : avec en moyenne 16 VPL par cours pour l'ensemble des cours qui contiennent au moins un VPL, et 25 VPL pour les cours proposant 4 VPL ou plus. Le nombre de quiz est également significatif (bien que moins important que le nombre de VPL). Les cours proposés utilisent donc de manière significative des contenus orientés vers la mise en activité effective des étudiants sur la plateforme avec des retours automatiques possibles de leur production. D'autres modules sont également utilisés pour mettre en valeur les contenus et soutenir la motivation et l'activité des étudiants, mais les activités VPL dominent. La présence du module VPL, sa qualité intrinsèque, sa malléabilité ainsi que le support technique, pédagogique et à l'exploitation offert par la plateforme expliquent une grande partie des choix pour Caseine au lieu des plateformes institutionnelles. Ces dernières sont rarement aussi réactives et à jour des plugins de programmation (qui n'intéressent, par ailleurs, pas forcément beaucoup les autres disciplines, hors l'informatique) et des autres modules.

L'ouverture et le partage, le module VPL et le support pour les activités de programmation entraînent une présence massive d'auteurs actifs et ouverts, informaticiens pour la plupart, mais pas forcément tous intéressés par l'enseignement de la programmation, stricto-sensu. Cependant, la présence forte de VPL n'oriente pas nécessairement l'ensemble des activités vers la programmation. Des cours sur les couches basses de l'informatique (représentation de l'information, circuits logiques, langage machine, système), sur les données (SQL, SGBDR, conception de BD) et d'autres cours qui dépassent le noyau usuel de la programmation (Python, C, Java) sont également disponibles (langages exotiques, analyse syntaxique, test logiciel, complexité algorithmique, sécurité informatique, etc.). Dans la suite, voici quelques exemples d'usages variés du VPL et de leur intégration à Caseine.

Retour automatisé pour les requêtes SQL

La question revient régulièrement dans la communauté. Une première preuve de

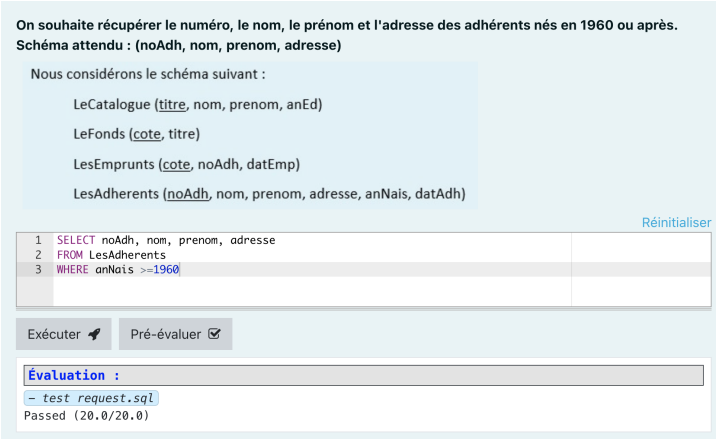


FIGURE 3. Évaluation de requêtes SQL.

concept et expérience à l'université Clermont Auvergne, un affinage du cahier des charges et un approfondissement de la réflexion par l'équipe pédagogique de l'université de Grenoble puis un développement soutenu par l'équipe Caseine aboutissent à un outil utilisé aujourd'hui par 2000 étudiants dans les deux universités, plusieurs lycées, un organisme de formation, ainsi qu'un cours « ouvert » en autoformation. L'idée initiale reprend l'esprit d'évaluation « boîte noire » du VPL : sans chercher à effectuer une analyse syntaxique approfondie de la requête soumise par l'étudiant, on se contente de l'envoyer au moteur de base de données (SQLite, serveur Oracle...) et de comparer les résultats obtenus avec ceux d'une requête de référence proposée par l'enseignant, sur une base de données idéalement suffisamment grande et variée (cf. figure 3). Ce choix à l'avantage de la simplicité et de l'indépendance vis-à-vis du dialecte, et l'inconvénient de la simplicité d'un retour « juste ou faux ». Pour détecter des erreurs courantes ou prédictives et fournir à l'étudiant un retour ciblé et des conseils de remédiation appropriés, on propose alors à l'enseignant de fournir non pas une mais plusieurs requêtes de référence, dont des requêtes incorrectes associées à un message personnalisé : la requête fournie par l'étudiant est alors évaluée en regard de chacune d'entres elles, jusqu'à trouver une correspondance, et c'est le retour (et la pénalité éventuelle) associé qui est appliqué. Notons qu'il n'est, en général, pas possible avec cette approche de distinguer deux requêtes donnant les mêmes résultats (même si des options permettent d'affiner, selon la présence ou l'absence de certains mots) : évaluer le style ou la performance reste la prérogative de l'enseignant. Cet outil a été utilisé à Grenoble pour transformer les travaux pratiques sur les requêtes SQL dans les enseignements de Licence en utilisant d'abord des VPL, puis des quiz Moodle avec questions VPL, interface qui se prête particulièrement

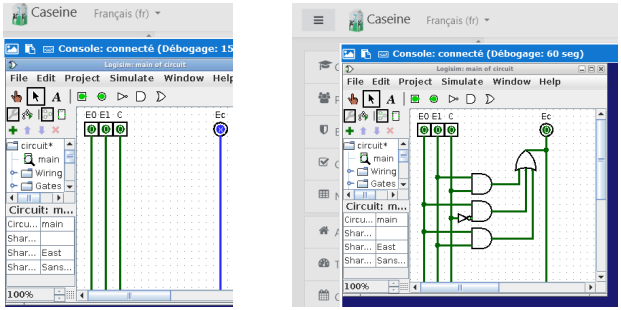
bien à cet usage (succession de petites questions à réponse courte), avec des effets positifs tant sur les suivis de TP que sur la correction de travaux notés, ainsi que sur l'intégration des ATER moins expérimentés dans l'équipe pédagogique. Pour les étudiants, comme pour les enseignants de passage, l'interface des questions VPL permet de masquer la complexité de l'environnement utilisé (Oracle), contourner les difficultés de connexion, et leur permet de démarrer rapidement en autonomie. Le suivi en temps réel de l'avancée des exercices pendant les TP s'est révélé utile (et pendant les périodes de confinement, indispensable) pour faire un support et des interventions personnalisées aux étudiants. Les retours automatisés (qui peuvent être adaptés en temps réel pendant la séance) favorisent l'autonomie des étudiants en TP, en débroussaillant les erreurs les plus classiques ; ce qui permet aux enseignants de se concentrer sur les difficultés de fond et le cœur de l'enseignement, ainsi qu'en facilitant la proposition de TP optionnels. Dans le cas des TP notés, le temps de correction a été réduit, les enseignants pouvant s'appuyer sur le retour automatique fourni dans l'interface de correction et y ajouter leur retour.

Circuits logiques

L'enseignement des circuits logiques (combinatoires, séquentiels, mémoires) pose de nombreux problèmes techniques (surtout) et pédagogiques. L'attrait de cette discipline vient de son aspect graphique, mais avant d'en faire un avantage, il faut pouvoir en surmonter les contraintes techniques : intégrer un outil graphique dans Moodle et VPL et adapter le processus de validation aux exercices de dessin de circuits.

Côté technique, le fonctionnement standard de Moodle/VPL repose sur les technologies web pour l'édition des programmes, l'intégration dans Moodle et sur un serveur Linux isolé (jail-server) pour l'évaluation des programmes et la mise au point (une interface Linux graphique est alors prévue). Deux pistes de solutions et deux technologies sont donc possibles pour l'intégration d'un outil graphique. En fait, les deux technologies vont être nécessaires. Parmi les logiciels de dessin de circuits logiques, c'est Logisim [6] qui a été choisi pour son ergonomie et ses fonctionnalités. Même si les sources de Logisim sont disponibles, l'intégration dans Caseine n'est pas passée par une adaptation du logiciel lui-même. En effet, l'exécution de Logisim sur le serveur Linux de VPL est facile. Fournir un circuit comme point de départ et permettre l'édition et la sauvegarde de ce circuit par l'utilisateur est aussi immédiat. Il reste à faire le lien entre Linux et Moodle. Le lien entre les fichiers de l'utilisateur sous Moodle et le serveur Linux isolé est prévu dans un seul sens par VPL (Moodle $\rightarrow$ Linux). Pour fournir un lien dans l'autre sens, un démon a été ajouté au serveur Linux pour scruter les changements de fichier utilisateur et envoyer via les services web Moodle/VPL prévus pour des soumissions à distance, les productions de l'étudiant. L'étudiant peut donc travailler avec Moodle/Linux/Logisim.

6. `http://www.cburch.com/logisim/`.



(a) (b)

FIGURE 4. Le circuit fourni initialement avec les entrées/sorties (a) et produit par l'utilisateur pour un multiplexeur (b).

Côté pédagogique, il faut aussi prévoir le test des circuits produits. À cet effet, Logisim a prévu une exécution des circuits à partir d'entrées fournies avec la production d'une trace des valeurs temporelles des sorties (afin de vérifier ces sorties). Le test prévu par VPL est similaire (tests basés sur un fichier d'entrées/sorties). Pour lier ces deux mécanismes de test, des transpositions sont effectuées des entrées VPL vers des fichiers d'entrée Logisim, puis après exécution de Logisim, des fichiers de sortie Logisim vers des sorties au format VPL. Pour aligner les entrées/sorties, la solution choisie fournit à l'étudiant un circuit initial ayant des entrées/sorties nommées suivant un ordre alphabétique spécifique. Remarque : pour les circuits séquentiels, c'est un peu plus compliqué, il faut ajouter ou prévoir un mécanisme (i.e. une partie de circuit supplémentaire) pour séquencer les entrées et mémoriser les sorties séquentielles.

L'intégration de Logisim à Caseine a mis en évidence plusieurs éléments. Le dessin de circuits logiques repose sur une application Logisim qui peut être transformée en activité (au sens de Moodle), sans changer l'application, à partir du moment où cette application peut s'apparenter à une boîte noire ayant des entrées et des sorties. Sur la base de ces entrées et sorties, une intégration à Caseine est possible, pour d'autres activités réductibles à une boîte noire similaire, en répétant le schéma suivi par l'intégration de Logisim par ajout de transpositions et d'un démon de communication. On peut ainsi songer à l'intégration d'activités de dessin d'automates formels, ou d'autres activités reposant sur des applications tierces graphiques ou non.

Pour les usages, le travail d'intégration et de mise en place d'activités de dessin de circuits a été accéléré par les conditions sanitaires de confinement. Cependant, la première année (2020-2021), le nombre d'activités VPL et l'intégration n'étaient

pas aussi avancée qu'aujourd'hui (juin 2022) ; mais les usages étaient plus importants (pour suppléer les présentiels manquants). En 2020-2021, les dessins de circuits faisaient partie des rendus du cours de L3 et ont été associés à la rédaction d'explications et à des calculs de complexité des circuits. Le contrôle de ces rédactions a été effectué par l'enseignant « à la main » et avec, hors plateforme, quelques outils en sus pour contrôler des aspects syntaxiques simples sur les circuits (vérification de l'absence d'utilisation inadaptée de la bibliothèque Logisim). À ce jour, le cours comporte 32 activités VPL, réparties en quatre groupes (premiers circuits, circuits combinatoires, automates et circuits à flots de données) ainsi que quelques vidéos. Plus d'une centaine d'étudiants de licence en ont profité, plusieurs centaines de circuits ont été dessinés. L'organisation en quatre groupes est liée de manière directe à la complexité des exercices. C'est par exemple, l'introduction des bascules ou points mémoires qui permet une complexité conceptuelle des circuits plus grande, mais qui est aussi associée à une complexité technique (le nombre de détails et de configurations possibles augmentent) et une taille de circuit aussi plus grande. En particulier, pour le passage des circuits combinatoires aux circuits séquentiels, le saut est important et l'effort pédagogique pour l'atténuer n'a peut-être pas été fait suffisamment dans ces activités VPL (pour le cours en présentiel, au tableau et avec papier-crayon, certaines séquences annexes rendent le passage aux circuits séquentiels plus progressif).

Une analyse informelle des productions des étudiants montre que l'activité est profitable à l'apprentissage en mesure de la complexité des circuits. Pour les circuits simples, le formalisme imposé de Logisim (plus rigide que celui demandé au niveau d'exercices papier-crayon) entraîne de meilleurs dessins : pas ou moins d'erreurs d'étourderie, qualité formelle des dessins, sans compter la clarté des tracés concrets (évidemment). Il semble qu'une partie des gains en apprentissage (en particulier ceux concernant la forme des dessins) a été conservée jusqu'à l'examen terminal. Cependant, pour les circuits plus complexes, la motivation étudiante et le nombre de solutions baisse avec Logisim, la proportion de situations d'évitement de la difficulté augmente (usage de circuits tout faits), même si l'activité reste motivante avec Logisim (comme l'ont fait remonter nombre d'étudiants). En comparaison avec le papier-crayon, les circuits séquentiels sont souvent incomplets et incorrects mais l'idée du circuit reste accessible.

Théorie des graphes

Les travaux pratiques de théorie des graphes peuvent prendre la forme de projets informatique conséquents qui nécessitent d'aller au-delà de l'interface web du VPL. Nous montrons l'usage du VPL dans le cadre d'un cours de théorie des graphes de licence d'informatique qui est dans la prolongation des cours de programmation et algorithmique. Ce cours partage de nombreuses activités avec les cours de recherche

opérationnelle. Pour un aperçu plus global, voir le cours ouvert et partagé de théorie des graphes [7] et [3] pour un aperçu plus détaillé.

Outre la maîtrise du domaine, ce cours vise, au travers de mises en activité sur machine, une prise de recul sur les structures de données utiles et sur la complexité algorithmique des solutions envisagées. Aussi, les programmes demandés, en Java ou en Python, sont basés sur une bibliothèque de classes fournie pour les implémentations élémentaires et si celle-ci fait quelques centaines de lignes, le travail étudiant ne concerne que les algorithmes de plus haut niveau qui ne font qu'une vingtaine de lignes.

Il s'agit, par exemple au cours d'une séance de TP de 3 heures, d'implémenter l'algorithme de Kruskal pour la recherche des arbres couvrants de poids minimum en utilisant une structure *Union Find* qu'ils doivent également programmer. Lorsqu'ils démarrent ce TP, les étudiants sont déjà familiarisés avec la classe `Graphe` fournie. L'algorithme de Kruskal a été vu en cours et en TD. Le TP doit permettre aux étudiants de consolider leur compréhension de cet algorithme et les forcer à réfléchir aux structures de données nécessaires pour une mise en œuvre effective et efficace. L'intérêt de la mise en pratique sur machine vient en particulier de cette confrontation à un objectif opérationnel qui ne peut être abordé avec autant d'acuité au tableau ou sur papier. L'utilisation d'une bibliothèque et la taille du code final complet permet aussi un travail à la frontière du génie logiciel et justifie l'utilisation d'IDE proposant des outils avancés de validation syntaxique, de documentation contextuelle, de complétion automatique, de débogage, etc. L'enjeu est alors de travailler sur de bonnes pratiques pour des codes qui commencent à devenir importants avec des objets complexes. En outre, certains étudiants utilisaient déjà ces IDE et l'équipe enseignante était favorable à une généralisation de ces usages. Cependant VPL, qui n'est pas un IDE, propose un environnement riche que l'on peut interroger de l'extérieur via des *webservices* ce qui permet les utilisations externes avec les IDE visés, VPL fournissant le lien avec Moodle et le système de validation qui lui est propre. Des *plugins* de connexion à Caseine ont donc été développés pour plusieurs IDE : VSCode/VSCodium, Eclipse, IntelliJ. Ces outils permettent de faire cohabiter l'environnement Moodle et un vrai environnement de développement : récupération (*pull* VPL→IDE) et envoi (*push* IDE→VPL) du code où le VPL permet de déposer les fichiers de l'étudiant et de faire tourner les tests automatiques. La connexion des étudiants à la plateforme est simple et transparente. Elle se fait au tout début de l'activité à partir d'un *token* d'identification Moodle de l'étudiant saisi une fois seulement par celui-ci avec le numéro de l'exercice VPL. Pour l'étudiant, c'est le seul lien avec Moodle, l'ensemble de son travail reste ensuite sur l'IDE ; le couple Moodle/VPL est complètement externalisé, il peut donc profiter et s'habituer aux IDE sans limite. Pour l'enseignant, le travail continue sur Caseine : il a accès aux

7. `https://moodle.caseine.org/course/view.php?id=147`.

codes et résultats des étudiants comme si ceux-ci étaient sur Caseine, l'utilisation de l'IDE est transparente pour l'enseignant.

Concernant la validation de l'activité, pour l'exemple Kruskal, l'activité fournit le code de base à l'étudiant et des tests fonctionnels, unitaires, de cas limites afin de permettre à l'étudiant de vérifier de façon incrémentale son avancée dans le travail demandé. En TP, généralement à la demande, l'enseignant peut lire le code, le valider ou le commenter et accompagner l'étudiant depuis Caseine en s'appuyant aussi sur les tests. Même si l'étudiant est incité à écrire quelques tests qui vérifient que l'algorithme fonctionne sur les cas « normaux », il ne lui est pas demandé de tester les cas particuliers même s'il doit les implémenter. Les étudiants voient une partie des tests qui seront utilisés pour évaluer leur travail. Ils ont ainsi accès à des tests correctement écrits dans un langage dédié au test (`JUnit` pour Java ou `Doctest` pour Python).

Un outil de retour automatique sur la complexité temporelle d'un algorithme. Comment réaliser automatiquement un retour à un étudiant sur la complexité temporelle de son algorithme ? Le problème est indécidable (au moins aussi difficile que le problème de la terminaison d'un programme) mais les complexités algorithmiques typiques rencontrées dans un cours d'informatique sont souvent assez limitées et se rencontrent au départ dans des cas académiques très épurés (recherche dans un tableau, recherche dichotomique, tris quadratiques, etc.). L'enjeu pédagogique de ce retour automatique est d'inciter un étudiant à aller au-delà « d'un programme qui fonctionne, qui passe les tests de validité ». Il doit permettre de provoquer des questionnements chez l'étudiant, sur la complexité algorithmique en général et sur celle de son algorithme dans un exercice précis. L'étudiant peut ainsi être sensibilisé à cet aspect et revenir vers l'enseignant.

L'outil développé comme une extension du VPL estime la complexité à partir du temps d'exécution de l'algorithme de l'étudiant sur différentes tailles de données d'entrée. Un ensemble de complexités possibles classiques (1, n, n^2, k^n, $n\log(n)$, 2^n...) sont comparées par régression linéaire ainsi que la complexité attendue et spécifiée par l'enseignant pour l'exercice. Les coefficients de corrélation et leurs intervalles de confiance sont utilisés pour éventuellement identifier une « bonne hypothèse ». En cas de succès de cette étape, le système fait un retour positif ou négatif (complexité meilleure ou moins bonne que celle attendue) à l'étudiant en indiquant la complexité identifiée. En cas d'échec, une régression non linéaire (de type $a+bx^c$ ou $a+be^{cx}$) est réalisée. Le système fait alors un retour positif ou négatif prudent sur la complexité observée en pratique et sa comparaison à celle attendue. Enfin, si la régression non linéaire colle mal aux données de temps CPU, l'analyse de la complexité est déclarée « non-concluante ». Cinq réponses sont ainsi envisagées : positive, négative, positive prudente, négative prudente, analyse non concluante. L'analyse doit être faite en ligne, rapidement et la taille des données à tester varie considérablement d'un exercice à l'autre. La taille maximum pour laquelle la limite de temps

du code étudiant est atteinte est rapidement identifiée afin de générer des tailles intermédiaires bien réparties dans cet intervalle dans le temps restant. Cette approche expérimentale ne permet pas de distinguer un facteur logarithmique dans les complexités (des données de taille exponentielle seraient nécessaires) et cette limite est indiquée dans le retour à l'étudiant. Pour utiliser ce système, l'enseignant doit spécifier une complexité théorique attendue pour un exercice mais aussi une méthode pour générer des jeux de données de taille paramétrée. Ces jeux de données définissent le type de complexité analysée (au pire cas, meilleur cas ou encore en moyenne) et exigent une réflexion soigneuse de la part de l'enseignant. Même si la mise en place pour l'enseignant nécessite quelques ajustements au début, par exemple pour le choix de la bonne famille de cas de tests générée, en pratique, cet outil fonctionne bien et permet vraiment d'accompagner les étudiants vers les objectifs de prise de recul sur les choix d'implémentation et les répercussions sur l'efficacité. Il a été utilisé entre autres dans des cours de recherche opérationnelle ou de théorie des graphes (voir ci-dessus) sur des centaines de codes d'étudiants. En pratique, il a par exemple permis dans ces cours d'identifier des codes exponentiels contre pseudo-polynomiaux en programmation dynamique, d'alerter les étudiants sur des récursivités avec mauvaise gestion de la mémoire, d'attirer l'attention des étudiants sur la complexité de leur structure *Union Find* pour l'algorithme de Kruskal, etc.

Pour rejoindre cette aventure, rendez-vous pour une première visite sur le site de Caseine [8].

Références

[1] Hadrien Cambazard, Nicolas Catusse, Nadia Brauner, and Pierre Lemaire. Teaching OR : automatic evaluation for linear programming modelling. 20(2) :333–345, 2022.

[2] Lydie du Bousquet and Christophe Saint-Marcel. De l'adaptation de caseine pour l'évaluation des tests des étudiants. In *21èmes Journées Approches Formelles dans l'Assistance au Développement de Logiciels (AFADL)*, 2022.

[3] N. Catusse, H. Cambazard, N. Brauner, B. Penz and F. Fontan. Innovative ideas for teaching supports : Application to Graph theory. arXiv :2209.05078, 2022.

8. `https://moodle.caseine.org`.