Ressources logicielles libres et open source dans l'enseignement de l'informatique en MP2I et MPI

Aslı Grimaud
Lycée Guy Mollet, Arras

L'enseignement de l'informatique en CPGE MP2I et MPI repose sur une formation exigeante, à la fois théorique et pratique, visant à transmettre une compréhension rigoureuse des fondements scientifiques de l'informatique et de ses implications techniques et sociétales. Dans ce contexte, l'usage de logiciels libres et *open source* constitue un levier pédagogique essentiel. Cet article présente trois outils emblématiques de cette approche : le système d'exploitation GNU/Linux, l'environnement de gestion de versions Git accompagné de GitLab, et l'éditeur de texte collaboratif HedgeDoc. En plus de leurs qualités techniques, ces outils incarnent une philosophie fondée sur l'ouverture, la transparence, le partage des connaissances et la maîtrise de l'environnement numérique. En plaçant l'informatique libre au cœur de la formation, on offre aux étudiantes et aux étudiants un environnement cohérent et éthique.

Les filières MP2I (mathématiques, physique, informatique et ingénierie) et MPI (mathématiques, physique, informatique), introduites en 2021 dans les classes préparatoires aux grandes écoles (CPGE), constituent une avancée majeure dans l'intégration de l'enseignement de l'informatique au sein des formations scientifiques postbac. Elles répondent à un besoin croissant de former des étudiants à la science informatique, non seulement comme un ensemble de compétences techniques, mais aussi comme une discipline scientifique à part entière. L'objectif de cet enseignement ne se limite pas à

DOI:10.48556/SIF.1024.26.23

l'acquisition de compétences techniques : il vise à former des personnes informées, capables de comprendre et de maîtriser les techniques numériques tout en mesurant les enjeux. Les filières MP2I et MPI préparent ainsi les étudiants à devenir les ingénieurs, les enseignants, les chercheurs de demain, mais aussi, plus largement, des citoyens capables de gouverner leur vie professionnelle et personnelle en s'appuyant sur une démarche scientifique rigoureuse et une conscience éclairée des transformations induites par l'informatique dans la société contemporaine [4].

Dans ce contexte, l'utilisation de ressources logicielles libres et *open source*[1] prend une importance stratégique. Un logiciel est dit « libre » s'il garantit à ses utilisateurs un certain nombre de libertés : liberté d'utiliser, d'étudier, de modifier et de redistribuer. Ces libertés s'accompagnent souvent d'une transparence du code source et d'une dynamique communautaire forte. Ce modèle s'oppose au logiciel propriétaire, dont le fonctionnement est opaque, figé, et généralement soumis à des conditions de licence restrictives. Les logiciels *open source*, tout en suivant largement les principes du logiciel libre, insistent davantage sur l'accès au code source et la collaboration technique ouverte.

Dans l'enseignement supérieur, les outils libres sont utilisés de longue date pour l'enseignement de la science informatique. Les CPGE, toutefois, relèvent administrativement du secondaire et héritent de l'infrastructure informatique des lycées, souvent conçue pour d'autres usages pédagogiques et dépourvue d'équipe technique dédiée à la gestion de systèmes complexes. Cette configuration rend l'adoption de solutions logicielles libres plus difficile, en particulier dans les filières MP2I et MPI, où l'informatique est l'une des disciplines principales. La mise en place d'un environnement cohérent et libre repose donc encore largement sur des initiatives locales, gérées au cas par cas.

L'utilisation de logiciels libres et *open source* en classe, notamment en informatique, présente plusieurs avantages fondamentaux :

– sur le plan pédagogique, ces logiciels encouragent une posture active et exploratoire. Les étudiants ne se contentent pas d'utiliser un logiciel : ils en découvrent le fonctionnement interne, les principes de conception, et la manière dont des projets logiciels réels sont pensés, développés, documentés et maintenus. Cette immersion progressive dans l'écosystème du libre favorise une compréhension de la logique collaborative, de la traçabilité des modifications, et de la transparence, autant d'éléments structurants pour une culture informatique solide ;
– sur le plan technique, ces outils permettent aux étudiants d'explorer concrètement les mécanismes internes des logiciels, bien au-delà d'une

1. Selon les expressions consacrées « *Free and open-source software (F/OSS, FOSS)* » ou « *free/libre/open-source software (FLOSS)* ».

simple utilisation en surface. Ils peuvent commencer par personnaliser des éléments, comprendre l'architecture des fichiers, manipuler des configurations, et ainsi mieux appréhender la structure d'un système informatique. Cette approche favorise également la compréhension de la gestion des fichiers, des permissions, et plus largement de l'organisation d'un ordinateur, souvent peu visible dans les environnements propriétaires. Elle ouvre aussi la voie à des notions clés telles que la sécurité informatique ;

– sur le plan éthique, les logiciels libres incarnent des valeurs essentielles pour toute formation à l'informatique : liberté d'usage, partage des connaissances, autonomie technologique et responsabilité individuelle et collective. Ils permettent d'initier les étudiants à des réflexions sur la souveraineté numérique, l'accessibilité du savoir, ou encore le rapport entre technologie et pouvoir. En découvrant qu'un logiciel peut être conçu, amélioré, partagé et utilisé librement par une communauté, les étudiants sont sensibilisés à une vision ouverte et émancipatrice de la technologie, en opposition à une logique de consommation fermée.

Cette orientation est d'ailleurs encouragée par les instances officielles. Le rapport n° 22-23-006A de l'IGÉSR [3], «*La préparation aux formations et aux métiers du numérique et de l'informatique…* », recommande explicitement de prendre en compte les besoins pédagogiques spécifiques à l'enseignement de l'informatique dans les politiques d'équipement des établissements (recommandation, section 2.4.4). Il souligne notamment que ces besoins incluent l'accès à des environnements logiciels adaptés, tels que les systèmes d'exploitation libres.

Dans cet article, on présente trois exemples concrets d'utilisation de ressources libres et *open source* en CPGE MP2I et MPI : le système d'exploitation GNU/Linux, l'environnement de gestion de version Git (et GitLab), et enfin l'éditeur collaboratif HedgeDoc, utilisé pour la prise de notes et la collaboration en classe. Ces outils, au-delà de leur efficacité technique, s'inscrivent dans une démarche pédagogique cohérente avec les objectifs de ces filières.

La mise en place d'un environnement informatique libre et cohérent au sein d'un établissement scolaire ne va cependant pas sans difficultés. Installer, configurer et maintenir des outils libres tels qu'un système GNU/Linux, un service de gestion de versions ou un éditeur collaboratif nécessite des compétences techniques avancées, que ce soit du côté des enseignants ou du côté des personnels techniques. La gestion d'une salle informatique avec un réseau local dédié, une connectivité stable, et parfois même des serveurs internes (physiques ou virtuels) adaptés aux besoins pédagogiques, requiert un investissement conséquent en temps et en savoir-faire. Il s'agit d'un véritable travail d'administration système et réseau, bien souvent

assumé de manière artisanale par des enseignants passionnés mais insuffisamment accompagnés.

À cela s'ajoutent des contraintes institutionnelles et techniques, telles que les restrictions imposées par certains pare-feux ou politiques de sécurité réseau, qui peuvent limiter l'accès aux ressources nécessaires pour l'enseignement de l'informatique. Ces freins techniques, lorsqu'ils ne sont pas anticipés ou levés à temps, ralentissent la mise en œuvre pédagogique des outils libres et réduisent l'ambition initiale des programmes. Dans ce contexte, l'utilisation de ressources en ligne reposant sur des serveurs externes, parfois opaques sur le traitement des données, tend à se généraliser, souvent par facilité, au détriment d'un enseignement maîtrisé, et conforme aux exigences pédagogiques de la discipline. Ce constat appelle à une reconnaissance structurelle forte de ces besoins spécifiques, et à une meilleure articulation entre les exigences de sécurité informatique et les impératifs pédagogiques, afin de garantir des conditions d'enseignement à la hauteur des enjeux de la formation en informatique.

Système d'exploitation GNU/Linux : outil d'apprentissage au service de la compréhension informatique

Le système GNU/Linux est emblématique de la philosophie du logiciel libre. Le projet GNU, initié par Richard Stallman en 1983 [5], visait à créer un système d'exploitation entièrement libre, composé d'outils, de bibliothèques et de programmes conformes aux principes de liberté logicielle. Il manquait toutefois un noyau fonctionnel pour que le système soit complet. C'est en 1991 que Linus Torvalds publie la première version du noyau Linux, qu'il place rapidement sous une licence libre [6]. La combinaison du système GNU et du noyau Linux a donné naissance à ce qu'on appelle aujourd'hui GNU/Linux, utilisé dans une multitude de contextes : serveurs, supercalculateurs, smartphones (*via* Android), objets connectés, etc.

Installation native

Utiliser un système GNU/Linux installé en natif constitue la solution la plus cohérente et la plus formatrice pour l'enseignement de l'informatique en CPGE. Cet environnement offre les outils pour une maîtrise complète du système, sans surcouche ni abstraction, ce qui permet aux étudiants de comprendre en profondeur la structure, le fonctionnement et la logique interne d'un système Unix. La gestion du système de fichiers, des utilisateurs, des processus, des périphériques, des droits d'accès ou encore des services en arrière-plan devient visible, manipulable et compréhensible.

C'est aussi une manière de se familiariser avec un écosystème largement répandu dans les domaines de la recherche, de l'ingénierie et de l'administration système.

Ce système respecte la norme POSIX, ce qui garantit une compatibilité avec d'autres systèmes Unix ou de type Unix (comme macOS ou BSD), ce qui en fait un socle pédagogique stable et standardisé. Des distributions comme Ubuntu sont un compromis particulièrement pertinent pour l'enseignement : une interface graphique conviviale, des dépôts de logiciels bien fournis, une large documentation, tout en conservant la puissance et la flexibilité du système Linux sous-jacent. De plus, l'installation d'Ubuntu sur un ordinateur personnel est une étape accessible et bien documentée, ne nécessitant pas de matériel particulier ni de licence payante. Cela permet aux étudiants de travailler dans un environnement homogène, à la maison comme en salle de cours, sans surcoût et sans dépendance à des solutions propriétaires.

Par contraste, une machine virtuelle, bien qu'utilisable dans certains contextes contraints, limite nécessairement l'immersion dans le système. D'abord, les performances sont dégradées : exécuter un système GNU/Linux dans une machine virtuelle implique une surconsommation de ressources (RAM, processeur, disque), ce qui peut conduire à des ralentissements notables, en particulier sur les machines les moins récentes. Ensuite, l'accès au matériel est souvent restreint : la gestion des périphériques réseau, des ports USB ou des fonctionnalités graphiques avancées peut s'avérer partielle, voire impossible à configurer de façon optimale. Enfin, l'environnement virtualisé introduit une couche d'abstraction qui masque de nombreux comportements du système. L'accès aux processus de fond, la configuration réseau réelle ou encore la manipulation directe du système de fichiers sont simplifiés ou artificiellement encapsulés. Cela empêche une appropriation concrète du fonctionnement global d'un système Unix et limite la compréhension fine des interactions entre les différents composants logiciels.

La machine virtuelle reste certes un compromis acceptable dans des environnements où l'on ne peut pas installer Linux directement, par exemple dans le cas d'un matériel imposé, d'un système de parc figé ou d'un besoin de compatibilité temporaire avec certains outils propriétaires. En revanche, elle ne peut en aucun cas remplacer l'expérience complète, directe et authentique qu'offre une installation native. À ce titre, l'installation en dur d'un système GNU/Linux reste la solution la plus cohérente pour former des étudiants à la compréhension du système, à sa maîtrise, et à sa philosophie.

Lorsque des contraintes institutionnelles imposent le maintien d'un système Windows, une solution viable consiste à mettre en place un double démarrage (*dual boot*). Cette configuration permet d'avoir un système GNU/Linux pleinement fonctionnel, sans renoncer à l'accès ponctuel à un

autre environnement. Toutefois, dès que cela est possible, il est préférable d'opter pour une installation Linux exclusive, plus simple à maintenir, plus fluide à utiliser, et surtout plus cohérente pédagogiquement. Elle développe une compréhension authentique de l'informatique, en permettant la manipulation d'un système libre, structuré, documenté, et non un environnement contraint par des choix technologiques extérieurs.

Maîtrise des droits d'accès et apprentissage structuré

Un système GNU/Linux repose sur une gestion rigoureuse des droits d'accès, ce qui en fait un support idéal pour apprendre les principes fondamentaux de la sécurité informatique. L'usage du compte *root* (superutilisateur) permet à l'enseignant de contrôler les installations et les configurations sensibles, tout en laissant aux étudiants la possibilité d'utiliser des commandes, comme `apt-get`, pour demander, ou proposer des installations, dans un cadre maîtrisé. Ce fonctionnement reflète en réalité celui que l'on retrouve dans les environnements professionnels : la séparation des rôles et la traçabilité des interventions.

Richesse des logiciels natifs

L'un des atouts majeurs d'un système d'exploitation libre est la quantité et la qualité des logiciels disponibles directement depuis les dépôts. Sans installation complexe ni conditions de licence restrictives, on peut accéder à une large panoplie d'outils essentiels à l'apprentissage de l'informatique :
– des compilateurs (comme `gcc`, `g++`) ;
– des outils d'automatisation de la compilation (`make`, `cmake`) ;
– des éditeurs puissants (Vim, Emacs, VSCodium) ;
– des interpréteurs (`python`, `bash`, `ocaml`, etc.) ;
– des outils de gestion de versions comme `git`, devenus incontournables dans le développement logiciel moderne ;
– des environnements de développement, des débogueurs, des outils réseau, etc.

Cette richesse est aussi un levier pédagogique : on peut choisir d'introduire progressivement ces outils, les comparer, ou inviter les étudiants à personnaliser leur environnement, ce qu'ils font souvent spontanément, notamment avec des éditeurs comme Emacs, dont la souplesse d'usage et la possibilité de configurer finement les raccourcis, les couleurs, ou les extensions, encouragent une appropriation active de l'environnement de travail.

Git et GitLab : une gestion de versions moderne

Git est un système de gestion de versions décentralisé, conçu en 2005 par Linus Torvalds, le créateur du noyau Linux, pour répondre aux besoins du

développement du noyau. Contrairement à un simple espace de dépôt ou de sauvegarde de fichiers, Git permet de suivre l'évolution d'un projet dans le temps, de revenir à une version antérieure, de comparer des modifications, de travailler à plusieurs en parallèle, et de fusionner des contributions. Ce fonctionnement est devenu la norme dans l'ensemble des projets logiciels modernes, qu'ils soient personnels, collaboratifs, académiques ou industriels.

Pour accompagner l'usage de Git, on peut s'appuyer sur GitLab, une plateforme libre qui permet d'héberger des projets versionnés, de les visualiser, de gérer les droits d'accès, les branches, les demandes de fusion, et même d'automatiser certaines tâches (tests, déploiement, etc.). GitLab peut être installé localement dans un établissement sur un serveur dédié, ou de manière plus souple dans un conteneur *via* Docker, ce qui facilite la maintenance, la portabilité et l'isolation du service. Cette approche par usage de conteneurs permet de disposer d'un environnement entièrement personnalisable, hébergé en interne ou sur un serveur distant, selon les besoins pédagogiques.

Gestion du travail informatique

L'usage de Git n'est pas qu'un outil pratique : il permet une structuration rigoureuse du travail informatique, tant pour les étudiants que pour les enseignantes et les enseignants [1]. Il met en place une logique de versions où chaque modification est enregistrée, contextualisée, datée, et réversible. Les étudiants découvrent ainsi que le développement logiciel ne se résume pas à une série de fichiers finalisés, mais à un processus d'itération, de documentation et de collaboration.

L'apprentissage de Git peut se faire de manière progressive. Dans un premier temps, les étudiants peuvent gérer leurs propres projets en local, en suivant l'évolution de leur travail, en créant des *commits* explicites, en testant des modifications sans crainte de perte. Dans un second temps, ils expérimentent le travail collaboratif, en partageant leurs projets sur GitLab, en gérant des branches, en résolvant des conflits et en découvrant les bénéfices d'un historique partagé.

Organisation claire et gain de temps pour tous

L'utilisation conjointe de Git et GitLab permet de mettre en place une organisation du travail efficace, lisible et reproductible, tant pour les étudiants que pour les enseignantes et les enseignants. Chaque personne dispose d'un dépôt personnel ou collaboratif, accessible depuis n'importe quel poste connecté à Internet. Fini les échanges de fichiers par clé USB ou par e-mail, ou les erreurs de synchronisation : chaque projet est centralisé, versionné et consultable à tout moment. Les pertes de données sont évitées, les erreurs peuvent être retracées, et les différents jalons du projet sont conservés.

Du côté enseignant, GitLab offre de nombreux avantages pédagogiques et logistiques. Il est par exemple très facile de fournir un squelette de projet à l'ensemble des étudiants. Ce dépôt de base peut contenir une arborescence de fichiers, des indications, des exemples ou des contraintes techniques. Chaque étudiant ou binôme peut ensuite bifurquer ce dépôt de référence (*via* une opération de *fork*), créant ainsi son propre espace de travail personnel, tout en conservant un lien clair avec le projet initial. L'enseignante ou l'enseignant peut ainsi structurer le travail en amont, tout en conservant une vision globale de l'avancement et des choix techniques de chacun.

De plus, pour les rendus, il suffit de cloner les dépôts étudiants pour accéder à leur dernière version à une date donnée, sans avoir à gérer des fichiers transmis hors délai ou dans des formats inattendus. L'environnement de développement est reproductible, ce qui facilite l'évaluation, la relecture, la comparaison, ou même la démonstration d'un projet en classe. GitLab permet également de suivre l'activité individuelle de chaque étudiant dans un projet collaboratif (nombre de commits, type de modifications, etc.), offrant ainsi un support objectif d'évaluation continue.

Enfin, l'usage de Git incite naturellement à privilégier des fichiers texte simples : code source, fichiers de configuration, documentation, README, etc. Cette approche encourage la clarté, la sobriété, la compatibilité entre systèmes et prépare aux bonnes pratiques du développement logiciel. Le format Markdown, très utilisé pour la documentation dans les dépôts, s'intègre parfaitement dans cette logique et sera développé dans la section suivante.

Démarche pédagogique active et professionnelle

Git habitue les étudiants à une méthodologie professionnelle de développement : gestion du temps, clarté des modifications, documentation intégrée, et respect d'un environnement partagé. Il prépare aux exigences des projets en école d'ingénieurs, en recherche ou en entreprise, tout en posant les bases d'un travail reproductible, traçable et rigoureux.

Du point de vue pédagogique, l'introduction de Git dès les premiers mois de la formation en MP2I et MPI permet d'instaurer une culture du code claire et structurée, dans laquelle chaque étudiant développe des compétences transversales : lecture de l'historique, relecture de code, structuration des *commits*, interactions dans un espace partagé.

HedgeDoc : un environnement de prise de notes collaboratif et structurant

HedgeDoc est un éditeur de texte collaboratif en ligne, conçu pour permettre à plusieurs utilisateurs d'écrire simultanément dans un document au format

Markdown. Il permet une visualisation instantanée du rendu final en parallèle du code source. Cette interaction directe entre le contenu brut (Markdown) et sa mise en forme constitue un atout pédagogique fort : elle rend les mécanismes de structuration du texte immédiatement visibles, tout en permettant de rester concentré sur l'essentiel, le contenu, et non l'interface.

Dans le cadre de l'enseignement en MP2I et MPI, HedgeDoc peut être installé sur un serveur accessible à l'ensemble des étudiants, selon un principe similaire à GitLab. Une installation par conteneur Docker permet de déployer facilement le service sur un serveur, tout en assurant une accessibilité continue depuis la salle de classe comme depuis un poste personnel. Cette installation autonome permet de se détacher de solutions propriétaires externes et d'offrir un environnement maîtrisé, sécurisé, et conforme aux principes du libre.

Prise des notes autrement

L'un des usages les plus pertinents de HedgeDoc est la prise de notes collaborative [2], notamment en cours, en travaux dirigés ou en travaux pratiques. Le programme d'informatique en MP2I s'y prête particulièrement bien, même si cela nécessite un travail préalable d'organisation et de préparation de la part des enseignantes et des enseignants. Lorsqu'elle est possible structurellement, cette pratique apporte de nombreux bénéfices pédagogiques. Les étudiants peuvent travailler en binôme ou en petit groupe, synchroniser leurs apports, se corriger mutuellement, construire une synthèse commune. Cette dynamique de collaboration active renforce l'engagement en classe et développe des compétences précieuses : écoute, clarté, esprit de synthèse, précision de l'écriture.

D'un point de vue plus technique et pragmatique, l'utilisation régulière de HedgeDoc permet aux étudiants de développer leur aisance au clavier, une compétence encore inégalement acquise à l'entrée en CPGE. Écrire sans chercher les touches, structurer un texte à la volée, insérer du code ou des formules, reformuler un contenu entendu ou compris : autant de gestes qui s'affinent avec la pratique, et que l'outil favorise.

Markdown et Latex : pour une écriture simple, sobre et efficace

L'un des principes fondateurs de HedgeDoc est son utilisation exclusive du format Markdown, un langage de balisage léger conçu pour structurer un texte sans interface graphique lourde. Au lieu de chercher des options dans une barre d'outils, les étudiants apprennent à utiliser des symboles simples ("#" pour les titres, "∗" pour les listes, etc.), ce qui renforce la maîtrise du

contenu et de sa structure. Cette pratique sensibilise également à l'intérêt des fichiers de texte simples et portables, au détriment des formats binaires fermés.

Markdown peut être enrichi de manière fluide par des éléments Latex, permettant d'intégrer des formules mathématiques directement dans le document. Cela ouvre la voie à une écriture scientifique lisible, propre et exportable, sans avoir recours à des traitements de texte complexes.

Outil à usage pédagogique personnel

Les étudiants prennent généralement en main HedgeDoc très rapidement, grâce à son interface claire et à l'édition en temps réel. Ils l'adoptent aussi bien comme outil de travail collaboratif que comme support personnel pour organiser leur apprentissage. Au fil des séances, il devient un véritable espace de production et de structuration des connaissances.

Ils y construisent leur propre documentation : synthèses de cours, réponses à des exercices, rappels de syntaxe, résumés de méthodes, bibliothèques de commandes utiles, etc. Le caractère textuel et brut de Markdown encourage une écriture sobre, lisible, et facilement versionnable (notamment en lien avec Git). Ils apprennent à organiser leurs notes de manière hiérarchisée, à distinguer l'essentiel de l'accessoire, et à enrichir progressivement un document qu'ils peuvent réutiliser ou adapter dans d'autres contextes.

Par ailleurs, HedgeDoc peut être utilisé pour préparer des exposés, servir de journal de bord de projet, ou encore pour les TIPEs. Les étudiants peuvent y inclure des explications, des fragments de code, des erreurs rencontrées, ou des pistes de réflexion ou construire des fiches récapitulatives. Ces pratiques favorisent non seulement l'autonomie, mais aussi le développement de compétences rédactionnelles et de métacognition.

Conclusion

L'utilisation conjointe de GNU/Linux, GitLab et HedgeDoc dans le cadre de l'enseignement de l'informatique en MP2I et MPI constitue une démarche cohérente, à la fois sur le plan pédagogique, technique et éthique. Elle transforme concrètement les pratiques pédagogiques en structurant des séquences d'apprentissage autour de projets techniques authentiques : versionner un code source avec Git, rédiger une documentation collaborative en Markdown, automatiser des tests, ou encore configurer un environnement Linux complet. Ces activités, loin d'être accessoires, deviennent le support même des apprentissages. Les étudiants développent ainsi des compétences transversales : organisation de projet, compréhension de l'historique des erreurs, expérimentation autonome, appropriation de la documentation technique. Cette approche rend visibles les étapes du raisonnement, invite à documenter ses

choix, à structurer ses méthodes de travail, et à entrer dans une culture informatique ancrée dans la pratique réelle. Elle fait émerger une posture active et critique, qui ne sépare pas l'outil de la pensée qu'il permet. Ces outils libres proposent ainsi un environnement d'apprentissage ouvert, structuré et rigoureux, qui permet aux étudiants de se former en profondeur aux pratiques réelles de l'informatique.

Au-delà des avantages fonctionnels, cette infrastructure logicielle présente également un intérêt majeur en termes de protection des données personnelles. Contrairement à de nombreuses solutions propriétaires, GitLab et HedgeDoc, lorsqu'ils sont installés sur un serveur local ou administré par l'établissement, stockent l'intégralité des données produites sur ledit serveur, ces dernières ne quittent pas le cadre institutionnel. Du côté du système d'exploitation, GNU/Linux présente également des garanties importantes. Les comptes utilisateurs sont entièrement locaux, sans identification centralisée par des services extérieurs. Ces outils libres ne sont ni soumis à des politiques d'exploitation commerciale, ni exposés aux réglementations extraterritoriales comme le *Cloud Act* américain, qui peut contraindre certaines entreprises à fournir des données à des autorités étrangères. Par exemple, un compte Microsoft lié à des services en ligne, soulève des questions de confidentialité dans un cadre éducatif. L'usage d'outils libres et auto-hébergés assure ainsi une conformité au RGPD et une maîtrise complète de l'environnement numérique, en accord avec les principes de souveraineté numérique et de protection des étudiants.

Cette démarche ne peut toutefois reposer uniquement sur l'enthousiasme individuel des enseignants. Elle suppose un engagement institutionnel fort. Il est donc essentiel que les besoins spécifiques de l'enseignement de l'informatique en CPGE soient pleinement intégrés aux politiques éducatives, tant au niveau académique que ministériel. Cela implique de garantir des moyens techniques adaptés (serveurs, salles équipées, connexions stables), de reconnaître officiellement la nécessité de disposer d'installations Linux en natif dans les salles de travaux pratiques, et de soutenir les équipes pédagogiques par l'affectation de personnels qualifiés en administration système et réseau. Plus largement, une politique publique ambitieuse en faveur du logiciel libre dans l'éducation, accompagnée de dispositifs de formation continue pour les enseignants et les personnels techniques, permettrait d'assurer une diffusion équitable et durable de ces pratiques. En intégrant ces outils dans les pratiques pédagogiques, on ne se contente donc pas d'enseigner l'informatique : on enseigne aussi une manière éthique, responsable et lucide de la pratiquer.

Références

[1] Julio César Cortés Ríos, Kamilla Kopec-Harding, Sukru Eraslan, Christopher Page, Robert Haines, Caroline Jay, et Suzanne Embury. 2019. A Methodology for Using GitLab for Software Engineering Learning Analytics. 3-6. https://doi.org/10.1109/CHASE.2019.00009.

[2] Axel Dürkop. 2022. *Collaborative content creation with HedgeDoc*. (Consulté le 17 juillet 2025). Technical University of Hamburg (TUHH). Consulté à l'adresse http://hdl.handle.net/11420/14416.

[3] IGÉSR. 2024. La préparation aux formations et aux métiers du numérique et de l'informatique : parcours, programmes, pédagogie, mixité des cursus dans les lycées généraux et technologiques et dans les lycées professionnels. (Rapport nº 22-23-006A).

[4] Ministère de l'Éducation nationale de la Jeunesse et des Sports. Programme d'informatique de MP2I et MPI. (Consulté le 17 juillet 2025). Consulté à l'adresse https://www.education.gouv.fr/bo/21/Special1/ESRS2035777A.htm.

[5] Richard M. Stallman. 2002. *Free Software, Free Society: Selected Essays of Richard M. Stallman*. (Consulté le 17 juillet 2025). GNU Press, Boston, MA. Consulté à l'adresse https://www.gnu.org/philosophy/fsfs/rms-essays.pdf.

[6] Linus Torvalds et David Diamond. 2001. *Just for Fun: The Story of an Accidental Revolutionary*. HarperBusiness.