

Mise en place et rendu des TME

Ce document décrit la mise en place des TME, le travail sur git, GitLab et Eclipse, ainsi que les étapes à suivre pour le rendu (obligatoire chaque semaine).

Nous prenons exemple sur le TME 1, mais ces directives resteront valides lors des TME suivants.

0.1 Logiciels utilisés

Nous utiliserons les logiciels suivants :

- **Le langage Java.**
Les sujets sont écrits pour la version 17 de Java, qui est la dernière version LTS à ce jour (*Long Term Support*) ; néanmoins, les versions supérieures devraient fonctionner.
Plusieurs versions sont installées à la PPTI (8, 11, 14, 17) ; il conviendra de choisir la bonne (≥ 17), sinon, certains sources fournis en TME et en cours pourraient ne pas fonctionner !
- **Eclipse** : un environnement de développement (IDE) pour Java.
Cet environnement gèrera votre projet localement sur votre compte à la PPTI ou votre ordinateur personnel. Il vous permettra d'éditer les sources, de les compiler et de les exécuter.
- **GitLab** : un gestionnaire de projets logiciel.
Basé sur le gestionnaire de versions distribué git, GitLab centralise votre projet sur un serveur externe. Il vous permet de garder dans un dépôt de sources (*repository* en anglais) un historique des modifications (visibles avec une interface web), de vous synchroniser avec votre binôme, d'alterner le travail entre les ordinateurs de la PPTI et votre ordinateur personnel et de faire le rendu du projet au chargé de TME. Nous y avons ajouté une fonction d'intégration continue : toute propagation des modifications locales (Eclipse) vers le serveur GitLab exécutera automatiquement une batterie de tests pour valider votre nouvelle version.
Nous utilisons ici un serveur GitLab privé, dédié au cours : <https://stl.algo-prog.info> (différent du site public gitlab.com, que nous n'utilisons pas).

0.2 Installation sur un ordinateur personnel

À la PPTI, ces logiciels sont déjà installés, vous n'avez rien à faire (vous pouvez sauter ce paragraphe). Si vous travaillez sur votre ordinateur personnel, vous pouvez les installer gratuitement :

- **Java.** Installez le dernier JDK Java « Ready for use » disponible sur OpenJDK : <https://jdk.java.net>. Alternativement, sur une distribution Linux telle que Debian ou Ubuntu, vous pouvez installer la version OpenJDK disponible dans le gestionnaire de paquets. Assurez-vous dans tous les cas que la version est 17 ou supérieure.
- **Eclipse IDE.** Installez Eclipse IDE, disponible sur <https://www.eclipse.org/downloads>. Durant l'installation, sélectionnez l'option « Eclipse IDE for Java Developers ».
- **EGit.** Ce plugin utile d'Eclipse est disponible sur le « Marketplace » (dans Eclipse, menu « Help », option « Eclipse Marketplace... » ; entrez « egit » dans la zone de recherche et installez « EGit – Git integration for Eclipse »).
- GitLab est installé sur un serveur distant, vous n'avez rien à installer.

0.3 Mise en place des TME

Les TME sont organisés en arcs d'une ou plusieurs séance(s). Chaque arc correspondra à un projet GitLab importé comme projet Eclipse. Nous prenons ici exemple sur le projet **MotCroise**, correspondant aux TME 1 à 3. Pour mettre en place le projet Eclipse et GitLab du TME, vous devez suivre les étapes ci-dessous :

Vérification de la place disponible (quota).

Si une des étapes échoue à la PPTI, cela peut être dû à un manque d'espace sur votre compte. Avant

toutes choses, utilisez la commande « **quota** » dans un terminal pour déterminer si votre compte dispose d'espace disponible. En particulier, si une étoile (*) apparaît, il ne reste plus de place sur le disque ; vous devez absolument libérer de l'espace avant de continuer.

Lancement d'Eclipse (**eclipse-2020**).

Plusieurs versions d'Eclipse cohabitent à la PPTI. Vous devez utiliser la plus récente en tapant, dans un terminal, la commande : « **eclipse-2020 &** ».

Si c'est la première fois que vous utilisez Eclipse, celui-ci vous demande le nom du répertoire *workspace*. Vous pouvez utiliser le nom par défaut `~/eclipse-workspace`, sachant que les projets gérés par git ne seront pas hébergés dans le *workspace*, mais dans un répertoire `~/git` séparé.

Configuration du proxy pour Eclipse (uniquement à la PPTI).

Si vous travaillez à la PPTI, choisissez dans Eclipse le menu « Window > Preferences » puis « General > Network Connections ». Changez « Active Provider » pour qu'il indique « Manual ». Vous pouvez alors modifier les entrées HTTP et HTTPS pour ce *provider* en cliquant dessus puis sur « Edit... ». Pour les deux entrées HTTP et HTTPS, choisissez comme « Host » : `proxy.ufr-info-p6.jussieu.fr` et comme port 3128.

Configuration du proxy du navigateur (uniquement à la PPTI).

À la PPTI, vous devez configurer le proxy de votre navigateur. Choisissez, pour les protocoles HTTP et HTTPS, le serveur `proxy.ufr-info-p6.jussieu.fr` et le numéro de port 3128.

Configuration de proxy pour git (uniquement à la PPTI).

À la PPTI, afin d'éviter les erreurs d'authentification avec le serveur git, vous devez ouvrir un terminal et y taper les commandes suivantes :

```
git config --global http.sslVerify false
git config --global http.proxy http://proxy.ufr-info-p6.jussieu.fr:3128
git config --global https.proxy https://proxy.ufr-info-p6.jussieu.fr:3128
```

(les deux dernières commandes ne sont utiles que si vous utilisez git en ligne de commande, tandis que la première est aussi nécessaire pour travailler sur Eclipse).

Connexion au serveur GitLab.

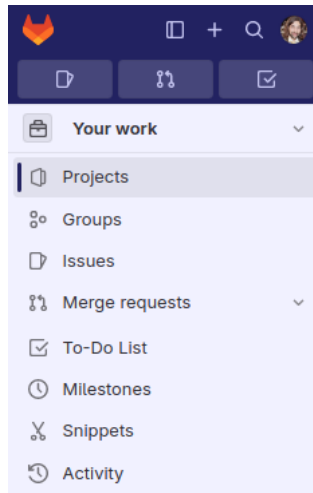
Le serveur GitLab vous a envoyé deux emails. Le premier a pour titre « *Account was created for you* » et contient un lien « *Click here to set your password* » que vous devez cliquer pour choisir votre mot de passe et activer votre compte. Le deuxième email indique que vous avez été ajouté au groupe du cours.

Les emails sont envoyés sur **votre compte email universitaire** `@etu.sorbonne-universite.fr` (généralement redirigé vers l'adresse que vous avez renseignée lors de votre inscription). Si vous ne trouvez pas cet email, pensez à le chercher dans votre boîte spam et sur le webmail étudiant. Si vous utilisez Gmail, l'email est souvent répertorié comme spam et les liens de connexion sont désactivés ; il est alors nécessaire d'indiquer explicitement à Gmail que l'email n'est pas dangereux pour rendre le lien « *Click here...* » cliquable. Si le message est trop vieux, le lien aura expiré. Vous pouvez demander l'envoi d'un nouvel email en cliquant sur « *Forgot your password?* ».

Loguez-vous sur le serveur <https://stl.algo-prog.info>.

- votre *username*, utilisé pour la connexion, est votre **numéro d'étudiant** ;
- votre *email* est celui de l'université (`@etu.sorbonne-universite.fr`) ;
- vous devrez choisir un mot de passe (fort et distinct de vos autres mots de passe, pour plus de sécurité) lors de votre première connexion.

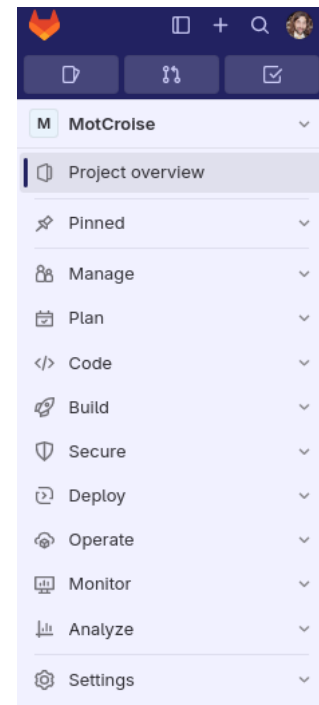
Vous devriez voir le menu « Your work » sur la gauche de la page, qui ressemble à l'image (a) de gauche ci-dessous :



(a) Menu « Your work » ouvert.

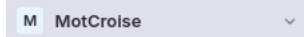
□ Your work > Projects

(b) Menu fermé.



(c) Menu de projet ouvert.

L'option « Projects », sélectionnée ici, permet d'accéder à liste de vos projets, à droite du menu. Si le menu n'apparaît pas, cliquez sur l'icône □ pour le faire apparaître (image (b)). Après avoir sélectionné un projet dans la liste (par exemple « MotCroise »), le menu de gauche change pour refléter les options du projet, comme montré sur l'image de droite (c). L'option « Project overview », sélectionnée ici, permet de naviguer dans les sources du projet. Vous pouvez revenir au menu « Your work », par exemple pour sélectionner un nouveau projet, en cliquant sur le titre du menu



ce qui fait ouvrir un menu déroulant où « Your work » apparaît.

Fork du squelette de projet sous GitLab.

Vous êtes automatiquement membre du groupe P0BJ-XXXX, où XXXX indique le semestre en cours (par exemple P0BJ-2023oct, pour 2023–2024). Ce groupe contient déjà un projet GitLab **MotCroise**, auquel vous pouvez accéder en lecture seule. Le projet est un squelette qui fournit quelques sources (exemples, tests, interfaces, etc.). Vous allez le compléter durant les TME 1 à 3. Les TME suivants seront réalisés dans d'autres projets, mais le principe de mise en place et de rendu sera le même.

Pour réaliser le TME et compléter ce projet, vous devez d'abord en faire **une copie sur le serveur GitLab (fork)** dans votre espace de nom. Cette copie sera **privée** à votre binôme et vous. Pour chaque binôme, **un seul élève** fera le *fork* du projet et il y ajoutera son binôme et son chargé de TME comme membres :

- Dans le menu « Your work » choisissez l'option « Projects » et sélectionnez le projet P0BJ-XXXX / MotCroise.
- Cliquez sur le bouton « Forks » en haut à droite. Dans la page « Fork project », ouvrez le menu « Select a namespace » et choisissez votre identifiant (numéro d'étudiant). Cliquez ensuite sur « Fork project ». Ceci crée le projet personnel privé nommé *username* / MotCroise (où *username* est votre numéro d'étudiant). Vous travaillerez désormais dans ce projet, où vous avez les droits d'écriture.
- Vérifiez que vous êtes bien dans ce nouveau projet privé. La ligne tout en haut de la fenêtre, à droite du menu, doit maintenant indiquer *username* > MotCroise au lieu de P0BJ-XXXX /

MotCroise. Si ce n'est pas le cas, revenez dans le menu « Your work », option « Projects » et sélectionnez le projet *username* / **MotCroise**.

- Dans le projet privé, cliquez sur « Manage > Members » dans le menu de gauche. **Invitez votre binôme et votre chargé de TME**, avec le bouton « Invite Members » en haut à droite et donnez-leur le rôle « **Maintainer** » (et pas l'option par défaut, « Guest »).

Si c'est votre binôme qui a créé le projet et vous y a ajouté, vous le trouverez dans votre liste de projets sous le nom *binome* / **MotCroise**, où *binome* est le numéro d'étudiant de votre binôme.

Importation du projet dans Eclipse (clone).

Dans Eclipse, choisissez dans le menu : « File > Import... > Git > Projects from Git » puis « Next ». Dans l'écran suivant, choisissez l'option « Clone URI » puis « Next ».

Sur la page du projet GitLab, cliquez sur le bouton « Clone » à droite, trouvez l'URL sous la mention « Clone with HTTPS ». Elle a la forme : `https://stl.algo-prog.info/forkusername/MotCroise.git` où *forkusername* est le numéro de l'étudiant qui a fait le *fork*. Copiez-là dans le champ « URI » dans Eclipse. Prenez garde à bien importer votre projet de binôme, *forkusername* / **MotCroise**, et pas **POBJ-XXXX** / **MotCroise** ; vous ne pourrez pas travailler dans ce dernier, qui est en lecture seule ! À la PPTI, vous devez utiliser le protocole HTTPS, donc une adresse commençant par `https://`. Une adresse commençant par `git@` utilise le protocole SSH, qui ne fonctionnera pas à la PPTI.

Dans « Authentication » vous devez renseigner vos identifiants GitLab : votre *username* GitLab (i.e., votre numéro d'étudiant) et votre mot de passe associé. Prenez garde à bien entrer vos identifiants de compte GitLab personnel, pas ceux de votre binôme, même si c'est lui qui a fait le *fork* sur GitLab. Le projet GitLab est commun au binôme, mais la copie locale du projet est personnelle. Tout membre du projet GitLab peut importer (cloner) une copie locale.

Le reste de la fenêtre se remplit automatiquement ; cliquez sur « Next ». Les écrans suivants, « Branch Selection » et « Local Destination » sont pré-remplis à des valeurs acceptables ; il suffit de cliquer « Next ». Choisissez « Import existing Eclipse projects » et « Next » puis « Finish ».

Un nouveau projet **MotCroise** apparaît dans le « Package Explorer » d'Eclipse, à gauche. La mention « [motcroise main] » associée au projet indique que le projet est bien géré par git. La copie locale des fichiers est hébergée dans le répertoire `~/git/MotCroise` (et non votre *workspace*).

Clones multiples.

L'importation (clone) d'un projet peut être faite simultanément par plusieurs membres du projet (par l'élève qui a fait le *fork* comme par son binôme) et sur plusieurs ordinateurs simultanément (en salle PPTI, chez vous, sur un portable, etc.). Cela vous permet de facilement travailler en binôme et en hybride à la PPTI et chez vous. La section suivante explique comment synchroniser les différentes versions locales de votre projet et celle hébergée sur le serveur GitLab.

Version de Java.

Plusieurs versions de Java sont installées à la PPTI. Nous vous proposons d'utiliser la plus récente, Java 17 à ce jour. Pour cela, faites un clic droit sur le nom du projet dans la fenêtre « Package Explorer » cliquez sur « Properties » puis, dans l'onglet « Java build path / Libraries » vérifiez la version de la bibliothèque système utilisée (« JRE System Library »). Si ce n'est pas Java 17, sélectionnez la ligne « JRE System Libraries » cliquez sur « Edit... / Alternate JRE » et choisissez un JDK 17. Si aucun JDK convenable n'est proposé, il faudra cliquer sur « Installed JREs... », « Add... », « Standard VM » et « Next > », « Directory » et choisir le répertoire `/usr/lib/jvm/java-17-openjdk-amd64`.

Sur votre ordinateur personnel, vous pouvez utiliser OpenJDK en choisissant la version 17 ou une version supérieure.

Vous pouvez maintenant réaliser le TME 1 et revenir à ce document en fin de séance pour les instructions de synchronisation de votre travail sur le serveur GitLab et de rendu de TME.

0.4 Synchronisation avec le serveur GitLab

Après avoir effectué une importation dans Eclipse (ou *clone* en terminologie git) en début de TME, nous avons travaillé sur une copie locale du projet. Il est nécessaire de synchroniser périodiquement votre projet local avec le projet GitLab pour :

- communiquer vos fichiers à l'enseignant, notamment pour le rendu hebdomadaire de TME (il a accès aux fichiers sur GitLab, mais pas à ceux sur votre compte PPTI ou votre ordinateur) ;
- vous synchroniser avec votre binôme ;
- éventuellement synchroniser des copies locales sur plusieurs ordinateurs ;
- garder une trace des modifications et pouvoir éventuellement revenir à une version précédente en cas d'erreur.

Les opérations utiles sont donc la propagation d'une copie locale vers le serveur (*push*) et depuis le serveur vers une copie locale (*pull*). Il est possible d'utiliser git, soit depuis l'interface graphique d'Eclipse (plugin EGit), soit en ligne de commande.

Vous pouvez consulter l'état des fichiers sur le serveur GitLab en utilisant le site web <https://stl.algo-prog.info>. Vous y trouverez la dernière version des fichiers connue par le serveur, et l'historique des modifications. Vous pourrez vérifier que le projet a été synchronisé par rapport à votre dernière copie locale, et voir exactement ce que votre chargé de TME verra lors du rendu.

0.4.1 Git sur Eclipse (EGit)

Push. Pour propager des modifications depuis le projet local vers GitLab, faites un clic droit sur le projet dans Eclipse, puis « Team > Commit... ». Un onglet « Git staging » apparaît en bas, ou dans une fenêtre (l'interface varie d'une version d'Eclipse à l'autre ; votre interface peut donc varier sensiblement de celle décrite ici). La zone « Unstaged changes » contient la liste des fichiers modifiés (ou ajoutés) localement mais non encore sur le serveur. Déplacez les fichiers que vous voulez synchroniser (fichiers modifiés ou nouveaux fichiers) vers la zone « Staged changes ». Entrez un bref message décrivant les modifications dans « Commit message ». Vérifiez la validité des zones « Author » et « Comitter ». En principe, elles devraient être identiques et contenir *username* <email> où *username* est votre numéro d'étudiant. Si c'est bien le cas, alors cliquez sur « Commit and push... » sinon, avant de cliquer, vous pouvez renseigner ces zones manuellement.

Pull. Pour récupérer localement, dans Eclipse, des modifications disponibles sur le serveur GitLab, faites un clic droit sur le projet, puis « Team > Pull ».

Conflits. Si des modifications ont été faites sur le serveur (par exemple par une propagation, *push*, de votre camarade) depuis votre dernier *pull*, vous ne pourrez pas propager vos modifications locales directement ; git refusera avec une erreur. En effet, cela provoquerait des conflits entre deux nouvelles versions d'un fichier. git vous force à résoudre les conflits localement, avant de propager vos fichiers corrigés vers le serveur :

- Faites d'abord un *pull*.
- Les fichiers marqués d'un diamant rouge dans le « Package Explorer » dans Eclipse, à gauche, sont les fichiers avec un conflit. Git s'est efforcé de fusionner les modifications locales avec celles présentes sur le serveur, mais il a pu faire des erreurs ; vous devez examiner chaque fichier et corriger à la main les problèmes causés par la fusion. Les zones non fusionnées sont identifiées par des balises <<<<<<, ----- et >>>>>> dans votre source Java. Git vous indique de cette manière les deux versions disponibles (version locale et dernière version disponible sur le serveur). Il s'agit souvent de choisir une des deux versions, en supprimant les lignes redondantes et les balises.
- Après avoir examiné et éventuellement corrigé un fichier en conflit, vous devez faire un clic droit sur le nom du fichier dans le « Package Explorer » dans Eclipse, puis « Team > Add to

index » pour indiquer que le fichier est maintenant correct. Le diamant rouge disparaît. Ceci doit être fait pour chaque fichier ayant un conflit.

- Après suppression de tous les conflits, vous devez faire un *commit*, avec « Team > Commit ». Le message de *commit* est renseigné automatiquement : il indique les fichiers qui étaient en conflit (vous pouvez bien sûr modifier ce message).
- Vous pouvez enfin faire un *push*.

Bonnes pratiques. C'est une bonne idée d'anticiper les conflits en **commençant toute session de travail par un *pull***, pour repartir avec les dernières versions des fichiers disponibles sur le serveur, et en **terminant toute session de travail par un *push***, pour que vos modifications locales soient envoyées sur le serveur et puissent être importées par votre binôme ou vous-même sur un autre ordinateur.

Il est possible d'éditer les fichiers du projet directement sur le serveur GitLab dans l'interface web (même si cela n'est pas conseillé car vous ne disposez pas alors des outils de développement d'Eclipse). Une telle modification compte comme un *push*, et vous devez donc l'importer dans Eclipse avec un *pull* avant de continuer à travailler sur Eclipse.

La documentation complète du plugin EGit se trouve sur <http://www.eclipse.org/egit/documentation>.

0.4.2 Git en ligne de commande (optionnel, alternative à Eclipse)



Dans un terminal, placez-vous dans le sous-répertoire de `~/git` contenant votre projet. Les commandes les plus utiles sont :

- `git add fichiers` pour indiquer les fichiers ajoutés ou modifiés localement (et les conflits résolus) ;
- `git commit` pour enregistrer localement les ajouts ou modifications indiqués par `git add` ;
- `git status` pour voir quels fichiers locaux ont des modifications non encore prises en compte par `git add` ;
- `git push` pour effectivement propager l'enregistrement local vers le serveur ;
- `git pull` pour rapatrier localement les modifications depuis le serveur ;
- `git log` pour lister les *commit*, avec date et message.

Le système git est décrit en détails dans le livre en ligne : <https://git-scm.com/book/fr/v2>.

0.5 Intégration continue sur GitLab

L'intégration continue est une pratique de développement logiciel consistant à s'assurer que, à chaque instant, le projet hébergé sur le serveur compile et passe tous ses tests. Le serveur GitLab est configuré pour l'intégration continue : après chaque propagation de votre copie locale vers le serveur, le code sur le serveur est compilé et les tests JUnit fournis pour les TME sont exécutés.

Après votre premier commit, vous pouvez consulter le résultat des tests sur le serveur GitLab en sélectionnant dans le menu du projet l'option « Build > Pipelines ». Une icône  indique que tous les tests sont passés correctement,  signale qu'au moins un test a échoué, et un croissant ou un symbole pause que les tests sont en cours et qu'il faut patienter (l'état au dernier commit est aussi visible sur la page du projet, au-dessus de la liste des fichiers).

Dans la page « Build > Pipelines », cliquer sur l'icône dans la colonne « Status » permet d'accéder à une page avec le détail des tests. Il y a généralement un jeu de test par TME. Cliquer sur le nom du jeu de test vous montre une console avec le log complet d'exécution. Il est également possible, en cliquant sur l'onglet « Tests », puis sur un TME dans la colonne « Job », de voir un résumé de chaque test du TME, indiquant s'il y a eu une erreur de compilation ou une erreur d'exécution pour chaque méthode de chaque classe de test.

Ce mécanisme vient compléter l'exécution des tests unitaires que vous devriez lancer périodiquement sur votre copie locale depuis Eclipse (au moins avant chaque propagation vers le serveur). Par ailleurs, le chargé de TME a accès aux rapports de tests sur le serveur GitLab, ce qui lui permet d'évaluer votre rendu de TME.

Le serveur est configuré pour exécuter tous les tests des TME 1 à 3. Tant que vous n'avez pas programmé toutes les classes demandées, de nombreux tests vont échouer. Vous ignorerez donc pour l'instant les tests liés aux TME 2 et 3, et essaierez de faire fonctionner les tests liés au TME 1.

L'intégration continue est gérée par le fichier `.gitlab-ci.yml` à la racine de votre projet, qui spécifie les classes de test.

0.6 *Release* et rendu de TME (OBLIGATOIRE À LA FIN DE CHAQUE TME)

Chaque semaine, il est obligatoire de rendre une première version de TME à votre chargé de TME **en fin de la séance**.

Si nécessaire, vous pouvez aussi rendre **une version finale avant le début du TME suivant**.

Pour chaque TME, le rendu final devra comporter toutes les classes demandées dans l'énoncé du TME. La section *Rendu de TME* de chaque séance comportera également des questions auxquelles vous devrez répondre dans votre rendu, ou bien des fichiers ou images à fournir.

Le rendu se fait en propageant vos modifications vers le serveur GitLab, comme indiqué à la sous-section 0.4, puis en créant une *release*. Pour cela, après avoir effectué la synchronisation (*push*) :

- Connectez-vous sur la page de votre projet sur <https://stl.algo-prog.info>.
- Assurez-vous que votre chargé de TME est membre de votre projet avec le rôle « Maintenir ».
- Vérifiez que toutes les classes demandées sont bien présentes dans GitLab et bien synchronisées avec le projet local visible dans Eclipse.
- Vérifiez également que les tests unitaires du TME lancés par l'intégration continue sur le serveur GitLab se sont exécutés correctement (voir 0.5).
- Dans le menu de gauche sur la page de votre projet sous GitLab, sélectionnez « Deploy > Releases » et cliquez sur « New Release ».
- Donnez un nom de *tag* à votre *release*, comme « rendu-initial-tme1 » ou « rendu-final-tme1 » selon qu'il s'agit d'un rendu partiel en fin de séance ou bien d'un rendu de TME finalisé. Pour cela, ouvrez le menu « Search or create tag name » sous « Tag name », entrez le nom du *tag* dans la boîte « Search », puis cliquez sur « Create tag... » en dessous et enfin sur le bouton « Save ».
- Dans le champ « Release notes » entrez les réponses aux questions posées à la fin du TME. Vous pourrez également, quand c'est demandé dans l'énoncé, ajouter des fichiers (texte, image) en les glissant dans le champ. N'hésitez pas à compléter ce champ avec toutes les informations que vous jugerez utiles pour évaluer votre TME : difficultés rencontrées, choix qui s'éloignent de l'énoncé, justification pour l'échec de certains tests JUnit, etc.
- Cliquez sur « Create release ». Votre *release* apparaît dans la page « Deploy / Releases ».
- En cas d'erreur, il est toujours possible de modifier les « Release notes » en cliquant sur le crayon à droite du nom de la *release*. Il est également possible de créer une *release* supplémentaire après avoir modifié votre projet.

Il est impératif de créer une *release* pour chaque rendu et de réaliser au moins un rendu par TME. Ces rendus réguliers (code source, validation des tests, réponses aux questions) sont évalués dans le cadre du contrôle continu.