amU Faculté des sciences Aix Marseille Université

Année 2024-25

Domaine Sciences et Technologies
Master informatique
Génie logiciel avancé : TP 1
Codes UE : SINB33DL
Gestion d'emploi du temps (1/6)

1 Introduction

L'objectif de cette séance de TP est de commencer le projet. Les étapes que vous allez devoir faire sont les suivantes (description détaillée dans la suite de la planche) :

- Créer votre équipe sur etulab : notion de *Group* de *gitlab*.
- Ajouter les enseignants à votre *Group*.
- Créer un tableau Kanban au sein de *gitlab* qui permettra de tracker les tâches à faire.
- Forker puis cloner les deux dépôts (un pour le client et un pour le serveur) pour avoir vos propres dépôts :
  - Lien dépôt client : https://etulab.univ-amu.fr/enseignants-gla/gla-client-template
  - Lien dépôt serveur : https://etulab.univ-amu.fr/enseignants-gla/gla-server-template
- S'approprier le code
- Faire des modifications sur vos dépôts (client et serveur) pour améliorer le code de base en créant à chaque fois une *issue* dans le tableau Kanban :
  - Ajouter la gestion des requêtes des requêtes autre que le *get* :
    - mise à jour (*update*) d'un créneau (*slot*) : HTTP PUT
    - récupération (*get*) d'un créneau avec identifiant : HTTP GET
    - création (*create*) d'un créneau : HTTP POST
    - suppression (*delete*) d'un créneau : HTTP DELETE
  - Améliorer les modèles :
    - Côté client : ajout d'un modèle contenant les créneaux téléchargés du serveur afin de limiter les requêtes.
    - Côté serveur : ajout de méthodes permettant de mieux contrôler le modèle (suppression, mise à jour, ...).

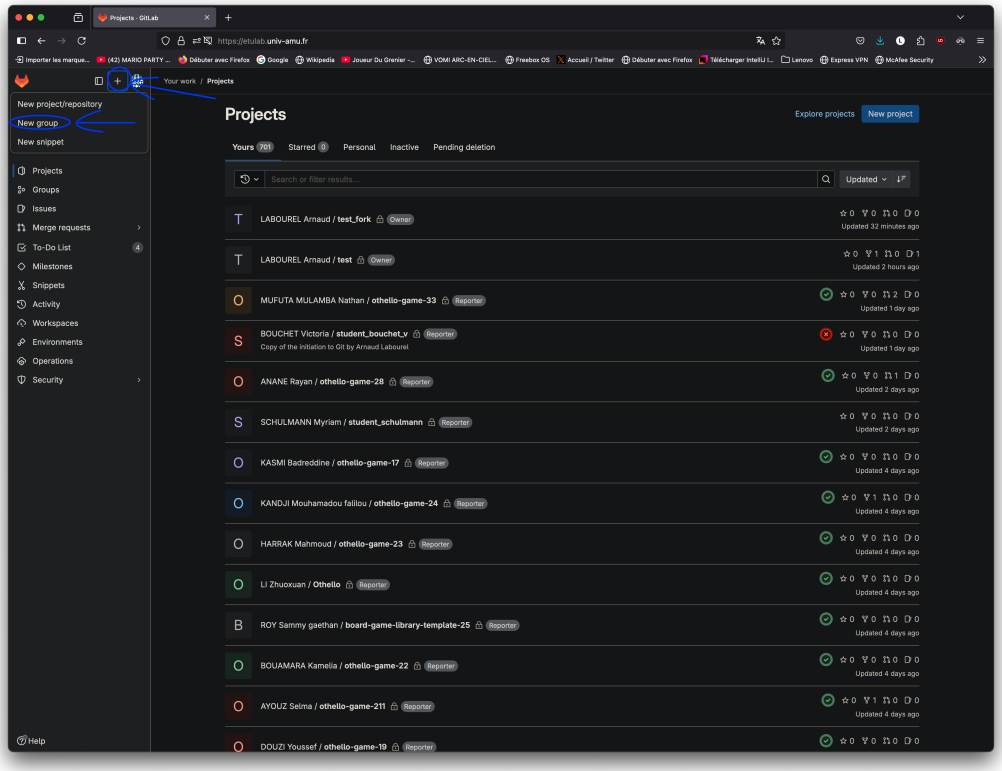
2 Mise en place des projets Gitlab

La première partie du TP consiste à mettre en place tout ce que vous aurez besoin sur *gitlab* pour permettre une gestion de projet efficace.

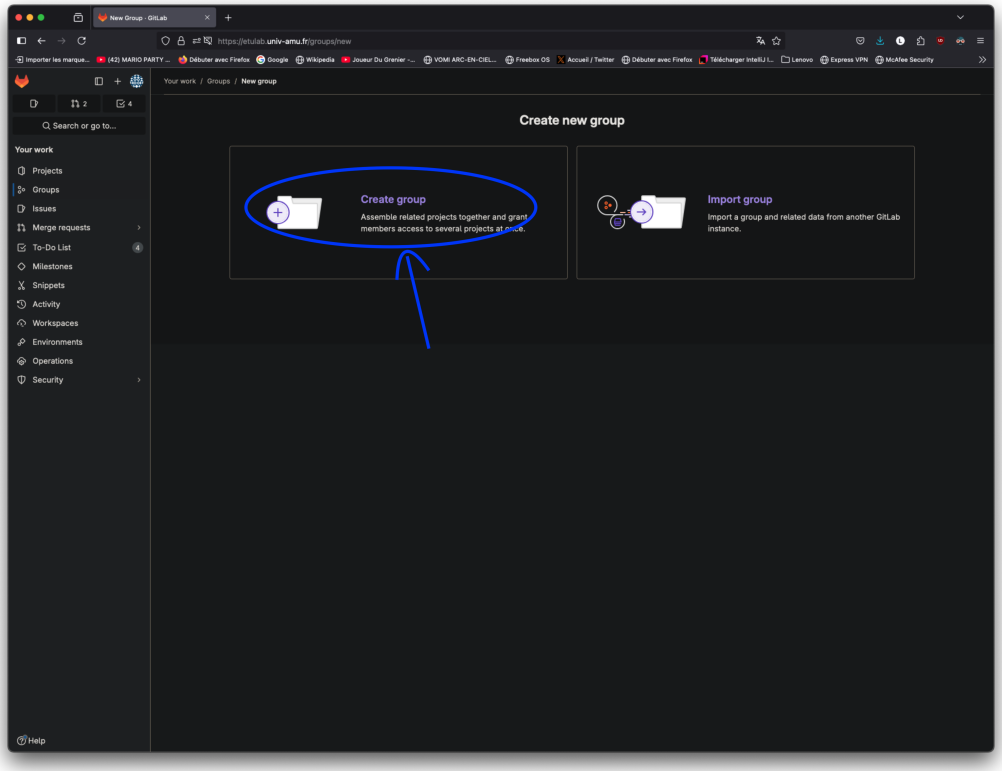
2.1 Création de groupe

La première étape consiste à créer un groupe qui correspondra à votre équipe sur le gitlab d'AMU. La personne créant le groupe sera la personne propriétaire du groupe (rôle *oweer*) qui aura les droits les plus importants dans le *group* :

- Connectez-vous avec vos identifiants AMU à etulab ;
- Ouvrez l'interface de création de groupe en cliquant sur le bouton `+` puis `New group`

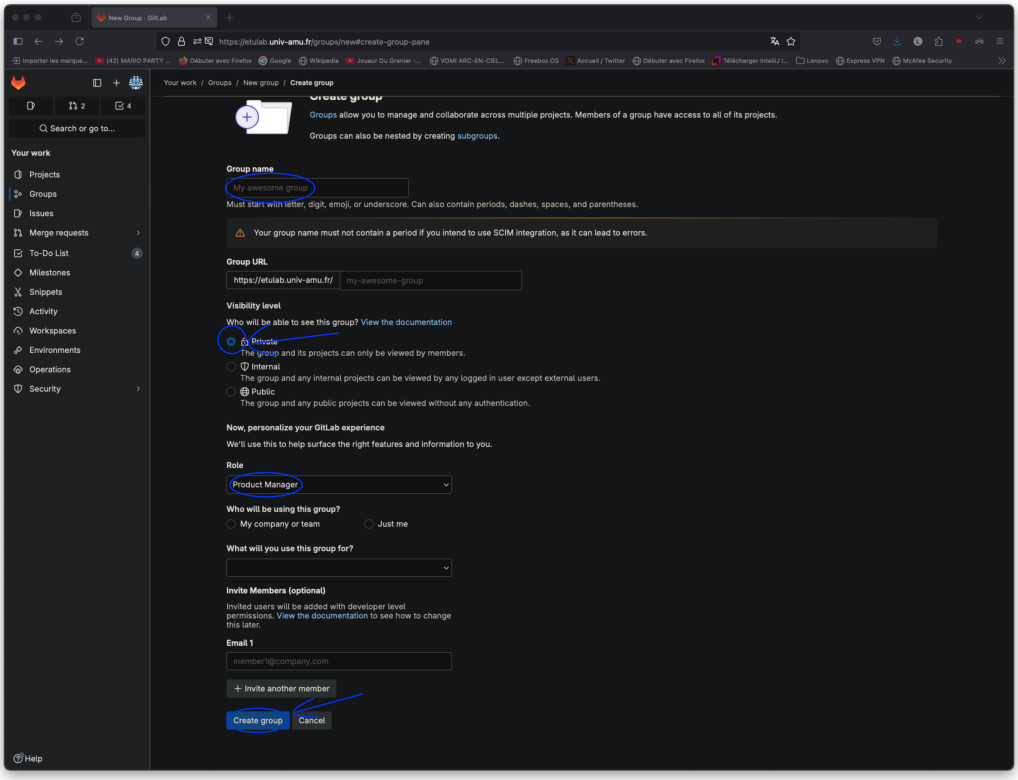


— Cliquez sur `Create group`

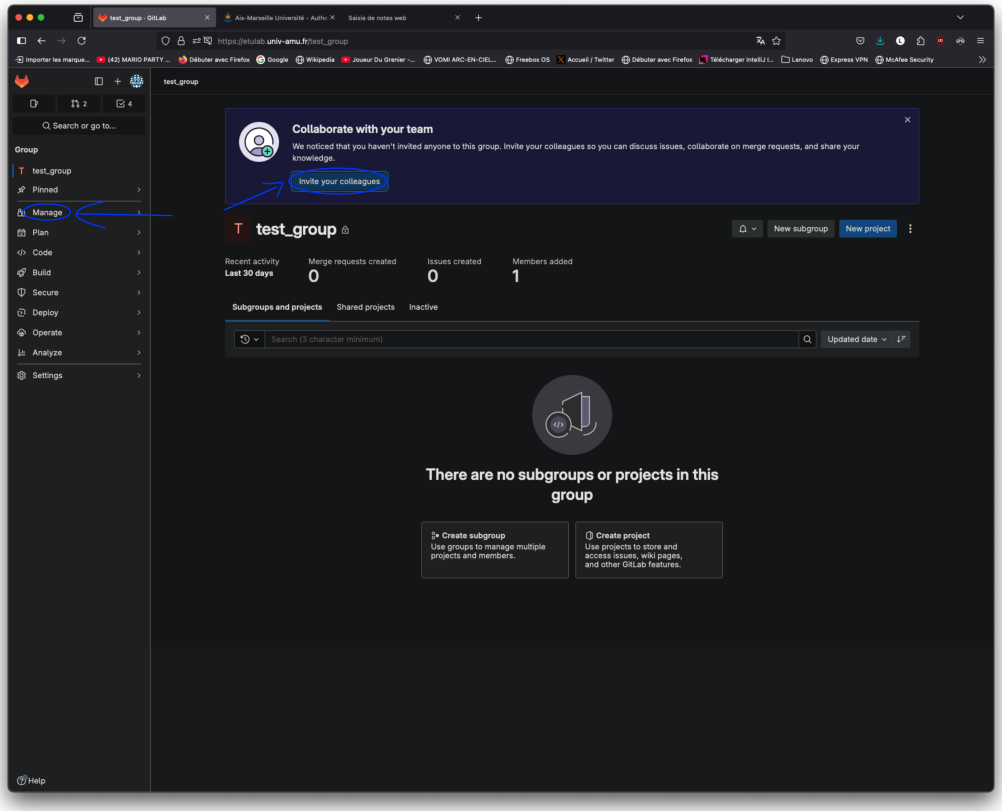


— Remplissez le formulaire de création de groupe :

1. Entrez le nom de votre équipe dans le champ `Group name` ;
2. Laissez la visibilité du groupe en privé ;
3. Laissez comme rôle `Product Manager` ;
4. Cliquez sur le bouton `Create group` pour créer le groupe.

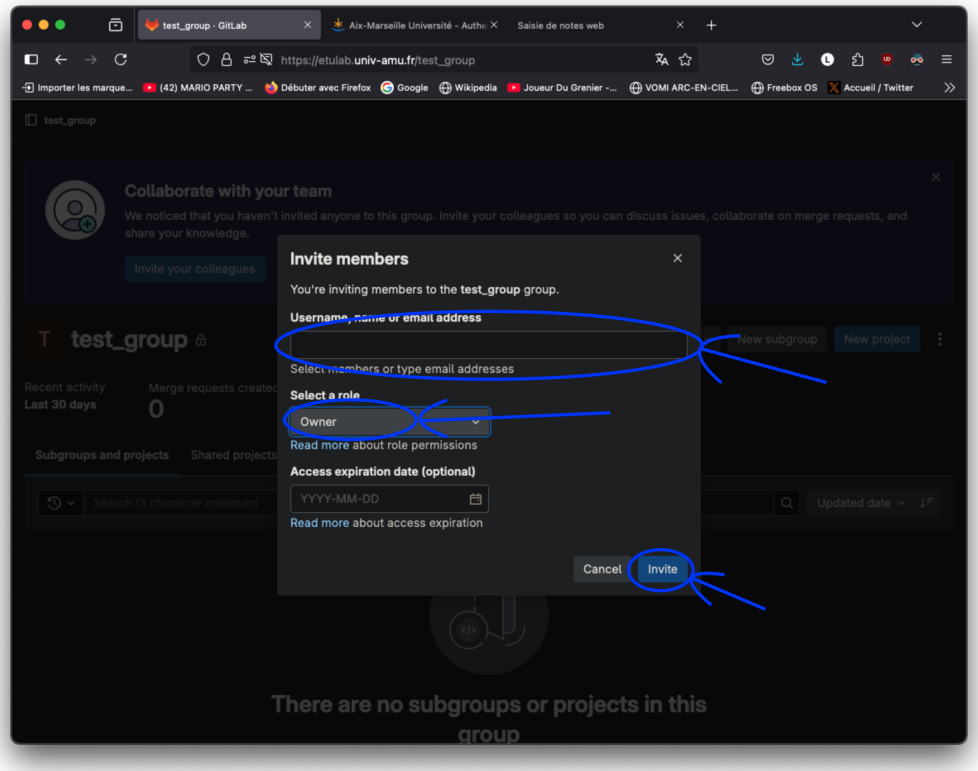


— Cliquez sur le bouton `Invite your colleagues`, ou bien sur `Manage` puis `Members` et enfin sur le bouton `Invite members` en haut à droite de votre fenêtre, pour accéder au menu d'invitation de membres.



— Ajoutez chaque autre membre de l'équipe en procédant comme suivant :

1. Entrez le nom de chaque membre de l'équipe à rajouter puis cliquez sur l'utilisateur proposé (s'il correspond bien à la personne que vous souhaitez ajouter) ;
2. Fixez le rôle (champs `Select a role`) de tous les membres invités à `maintener` ;
3. Cliquez sur le bouton `Invite` pour finaliser l'invitation de tous les membres.



— Ajoutez les enseignants de l'ECUE Génie Logiciel Avancé à votre groupe en procédant de la manière suivante :

1. Retourner sur la page dédiée à l'ajout de membre (en choisissant `Manage` puis `Members` dans le menu à gauche) ;
2. Cliquez sur le bouton `Invite member` en haut à droite de votre fenêtre ;
3. Chercher les deux enseignants de GLA (François=Xavier Dupé et Arnaud Labourel) et ajoutez-la à votre groupe avec le rôle reporter.

2.2 Création d'un tableau Kanban dans gitlab

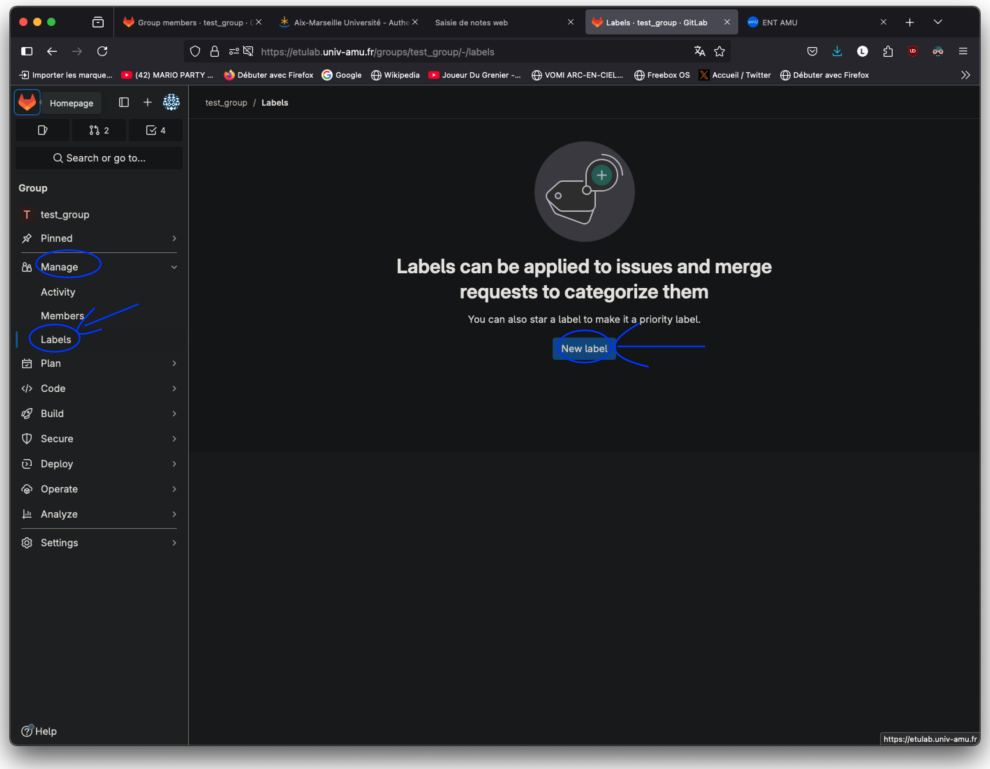
Afin de pouvoir gérer les tâches du projet, vous allez créer un tableau Kanban associé au groupe *gitlab* correspondant à votre équipe. Pour cela, on vous demande de suivre les étapes suivantes. Une description plus détaillée sur la manière d'utiliser *gitlab* pour faciliter l'approche *Kanban* est disponible au lien suivant : Use GitLab to facilitate Kanban.

2.2.1 Création des labels

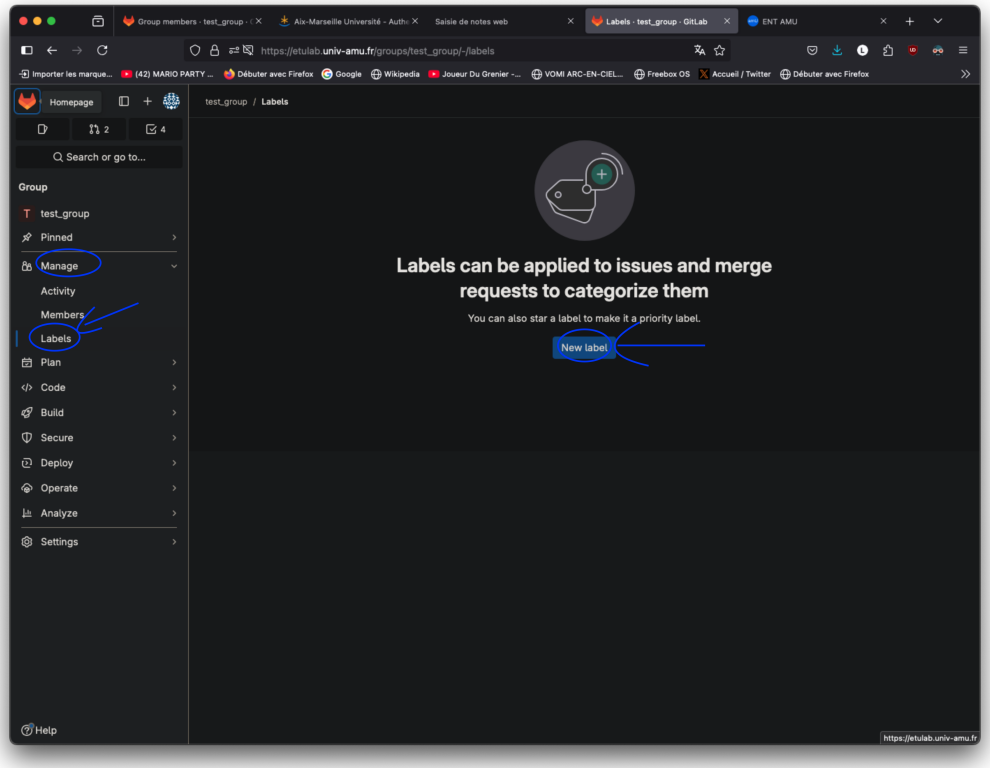
La première étape consiste à créer des labels qui correspondront aux différentes colonnes de votre tableau Kanban : `to do` (à faire), `doing` (en cours) et `done` (finies).

1. Aller dans le `group` de votre équipe ;
2. Sélectionner `Manage -> Labels`

3. Cliquez sur le bouton `new Label`



4. Entrez `status::to do` comme nom au label (`status` correspond à la portée du label) ;
5. Entrez comme description “Tâches à faire” ;
6. Choisissez la couleur cramoisi (*crimson*) ;
7. Cliquez sur le bouton `Create label` pour finaliser la construction :

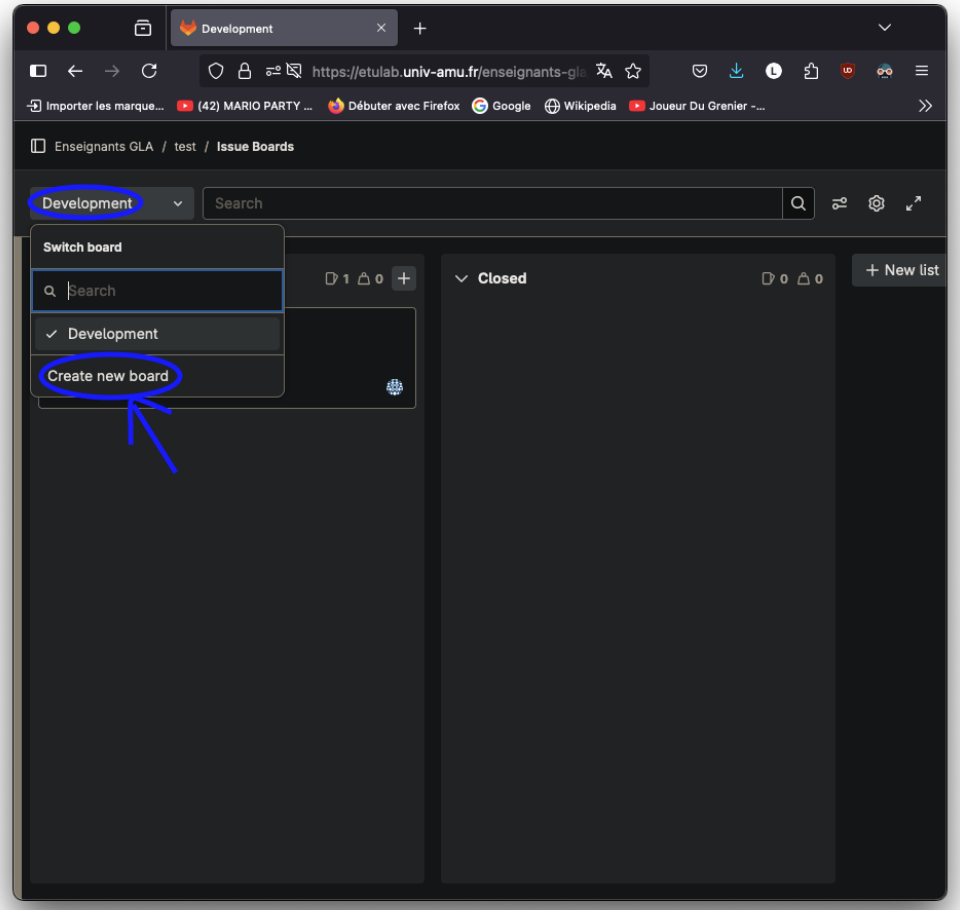


8. Répétez les étapes de création de label pour créer les deux labels suivants :

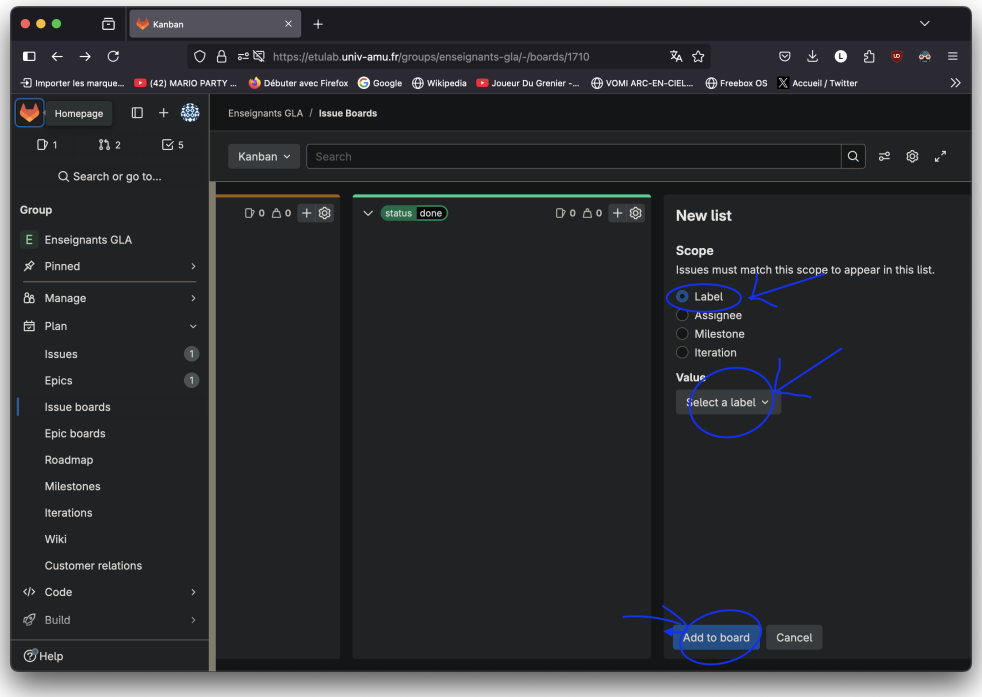
   — Nom : `status::doing`, description : tâches en cours, couleur : *Carrot orange*
   — Nom : `status::done`, description : tâches finies, couleur : *Green-cyan*

2.2.2 Création du tableau Kanban de l'équipe

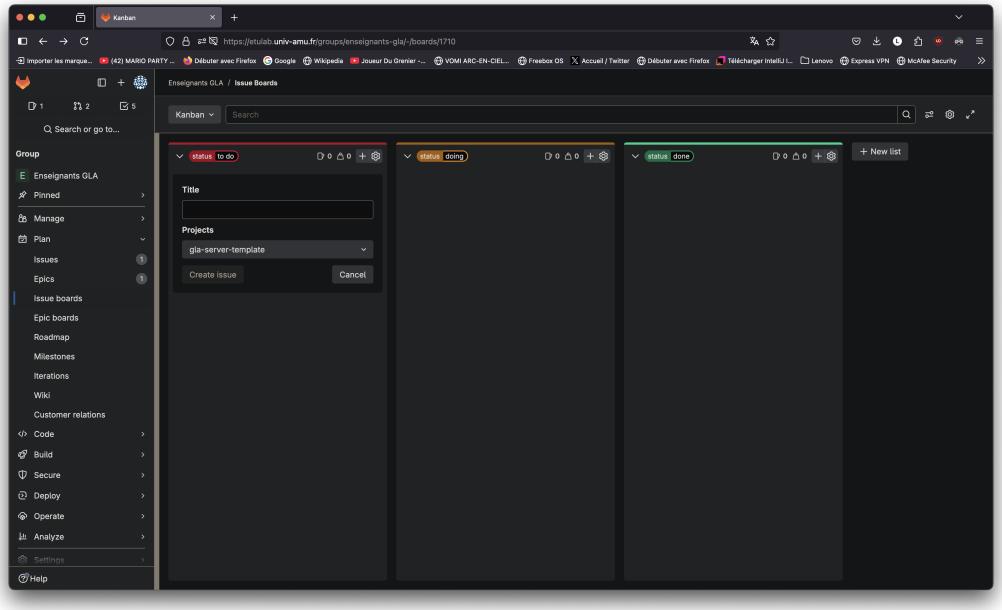
1. Aller dans le `group` de votre équipe ;
2. Sélectionner `Manage -> Issue Boards` ;
3. Dans le coin supérieur gauche du tableau de bord, sélectionnez la liste déroulante contenant le nom du tableau actuel.
4. Sélectionnez `Create new board`.



5. Créez une nouvelle liste d'étiquettes en sélectionnant `+ new list`.
6. Sélectionner Label pour savoir comment seront sélectionnées les issues et la valeur sur `statut::to do` pour `Value` puis ajouter la liste en cliquant sur `Add to board`.



7. Répétez le même processus (étapes 5 et 6) de création de liste d'issues associées à un Label pour créer deux autres listes : `status::doing` et `status::done`.
8. Vous deviez obtenir le tableau suivant :



2.2.3 Outils de visualisation de flux

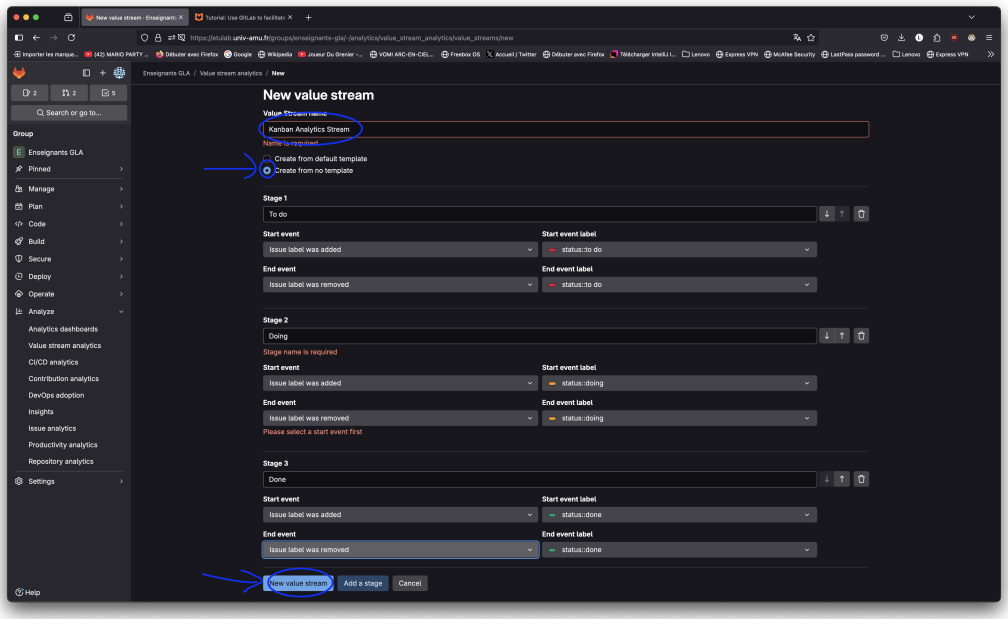
La méthode Kanban utilise des diagrammes de flux cumulatifs pour visualiser la charge et aider à identifier les goulots d'étranglement. Dans GitLab, cela peut être réalisé avec Value Stream Analytics (VSA). Vous allez créer un rapport VSA personnalisé qui correspond à votre flux de travail Kanban.

2.2.3.1 Création d'un flux de visualisation Kanban Pour créer le flux de visualisation du tableau Kanban de votre *group* :

1. Aller dans le `group` de votre équipe ;
2. Dans le menu à gauche, sélectionnez `Analyse -> Value stream analytics`.
3. Cliquez sur `New value stream...`.
4. Saisissez `Kanban Stream Anlytics` dans le champ `Value Stream name`
5. Créer un premier `stage`

    1. Saisissez `To do` pour son nom.
    2. Mettez `Issue label was added` comme `Start event` et `status::to do` comme `Start event label`
    3. Mettez `Issue label was removed` comme `End event` et `status::to do` comme `End event label`

6. Cliquez sur `add a stage` et répéter l'étape précédente pour créer un `stage Doing` avec comme étiquette `status::doing`
7. Répéter l'étape précédente pour créer un `stage Done` avec comme étiquette `status::done`
8. vous devriez obtenir une fenêtre similaire à celle ci-dessous que vous devez ensuite valider en cliquant sur `new value stream` :



2.2.3.2 Affichage du flux de visualisation Kanban

Avec votre rapport VSA personnalisé qui correspond au même flux de travail que votre tableau Kanban, GitLab calcule automatiquement le temps que chaque `issue` passe dans chaque colonne du tableau (correspondant à un `stage` dans le flux).

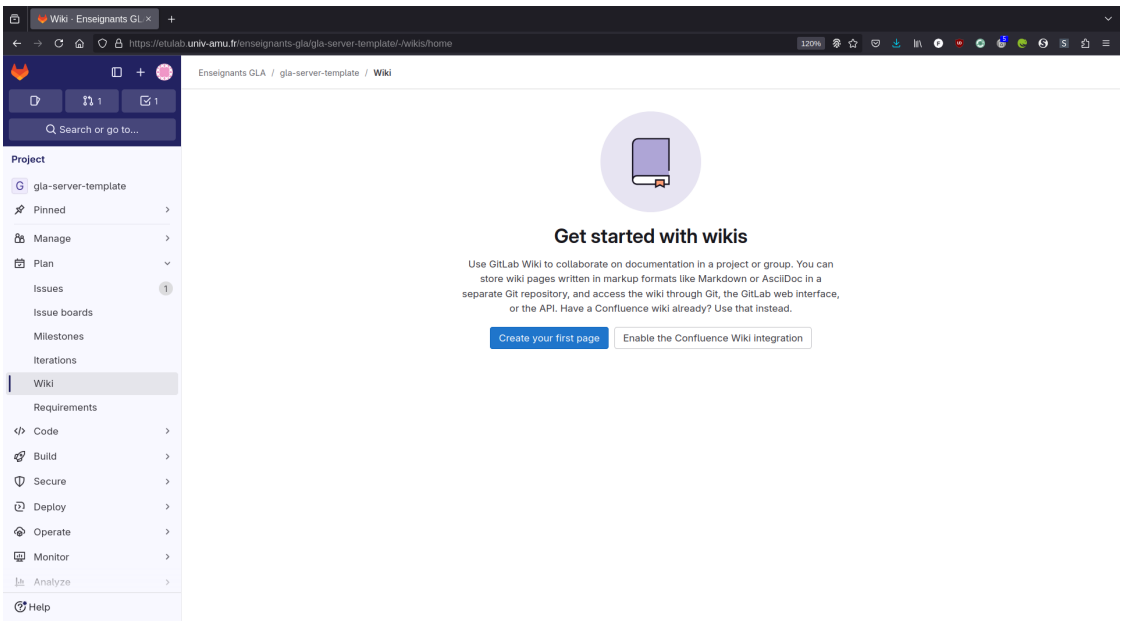
Pour visualiser la distribution :

1. Dans le rapport VSA que vous avez créé, faites défiler vers le bas jusqu'au graphique `Tasks by type`.
2. Sélectionnez l'icône de l'engrenage dans le menu déroulant en haut à droite, puis recherchez et sélectionnez les `Labels` que vous souhaitait afficher. Cela vous permettra de voir l'évolution du nombre d'`issues` de chaque type sélectionné (*to do, doing, done*) sur la durée du projet. Pour le moment, le graphique est vide, car aucune `issue` n'a été créée.

3 Création de la documentation du projet

Avant de commencer à prendre le projet en main, vous allez construire un wiki pour la documentation du projet. Cette documentation contiendra les différents documents techniques présentant les choix, les diagrammes et autres textes utiles pour la compréhension du projet. Ce wiki servira aussi pour les comptes-rendus.

La création du wiki ce fait par l'item `Wiki` dans l'entrée `Plan` du menu à gauche (voir image ci-dessous). Puis cliquer sur le bouton `Create your first page`, qui va créer la page principale (home).



Ce wiki prend en compte la syntaxe Markdown avec des extensions proposées par Gitlab.

Ces pages formeront la documentation de l'architecture de l'application. Vous pouvez vous inspirer des templates de ARC42 pour structurer les pages.

4 Appropriation du projet

4.1 Fork des dépôts

La première chose que vous avez à faire est de récupérer les deux dépôts et les forker pour avoir votre propre version. Pour cela, vous devez donc forker chacun des deux dépôts suivant en mettant comme `namespace` (lorsqu'on vous demande le `Project URL` après avoir cliqué sur le bouton `fork`) le nom de votre `group` :

— Lien dépôt client : https://etulab.univ-amu.fr/enseignants-gla/gla-client-template
— Lien dépôt serveur : https://etulab.univ-amu.fr/enseignants-gla/gla-server-template

4.2 Création de l'epic et des issues de l'appropriation du code des dépôts

La première tâche que vous avez à faire est de lire (pas forcément en entier) le code des deux dépôts afin de comprendre leur fonctionnement. Pour cette tâche ainsi que pour toutes les suivantes, on vous demande de créer des `Epics` et des `Issues` qui vont vous permettre de planifier et gérer les tâches du projet directement dans *gitlab*.

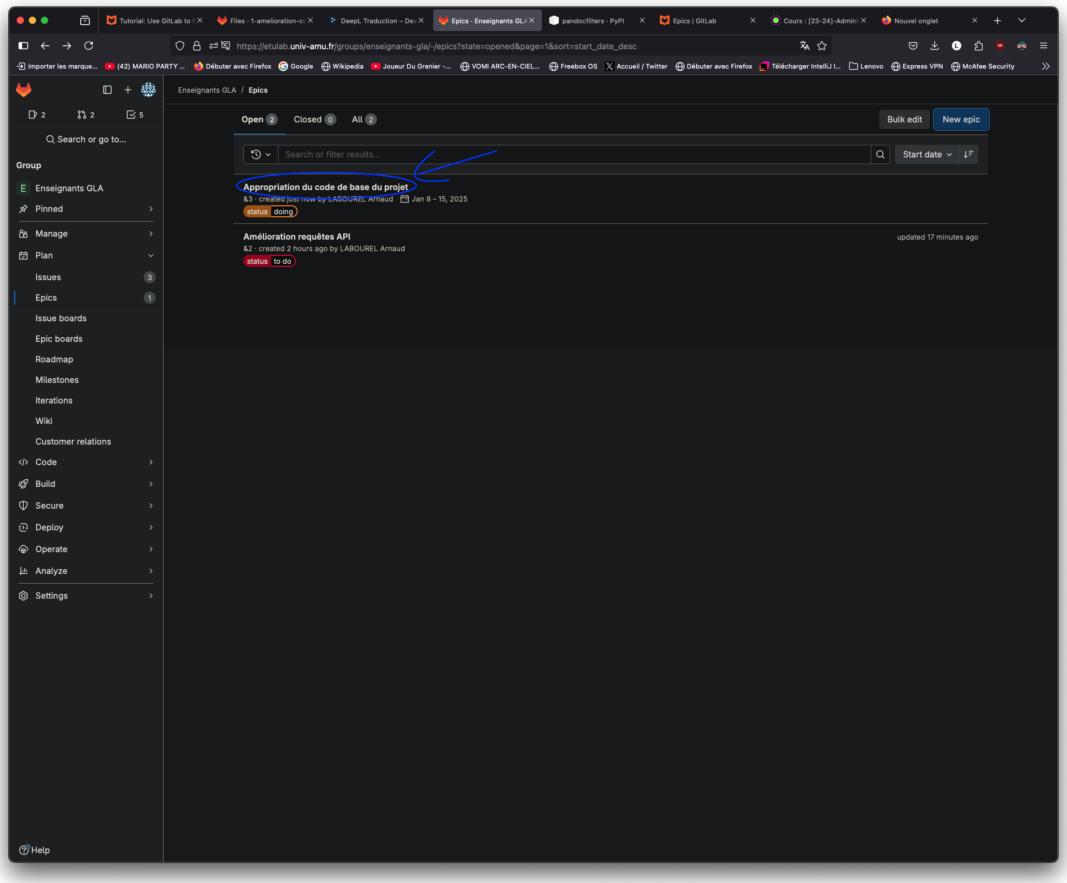
Pour cela, vous allez commencer par créer un `Epic` (manière dans *gitlab* de créer une notion correspondant à une certaine quantité de travail qui pourra contenir plusieurs `Issues` qui correspondront à des tâches).

Pour créer un `Epic` dans *gitlab*, il suffit de vous mettre au niveau de votre `group` de sélectionner dans le menu à gauche `Plan -> Epics` puis de cliquer sur le bouton à gauche `New Epic`. On vous proposera de créer un nouvel `Epic` avec :

— un `Title` correspondant au nom de l'`Epic` (donc par exemple "Appropriation du code de base du projet") ;
— une description avec un champ texte qui vous permet de détailler le but de l'`Epic` ;
— un `Label`, vous pouvez directement mettre *status::doing* ;
— des dates de début et de fin : à décider selon le temps que vous pensez mettre, mais vous pouvez mettre 8 janvier au 15 janvier pour le moment ;
— d'autres paramètres comme la couleur ou le *health status*.

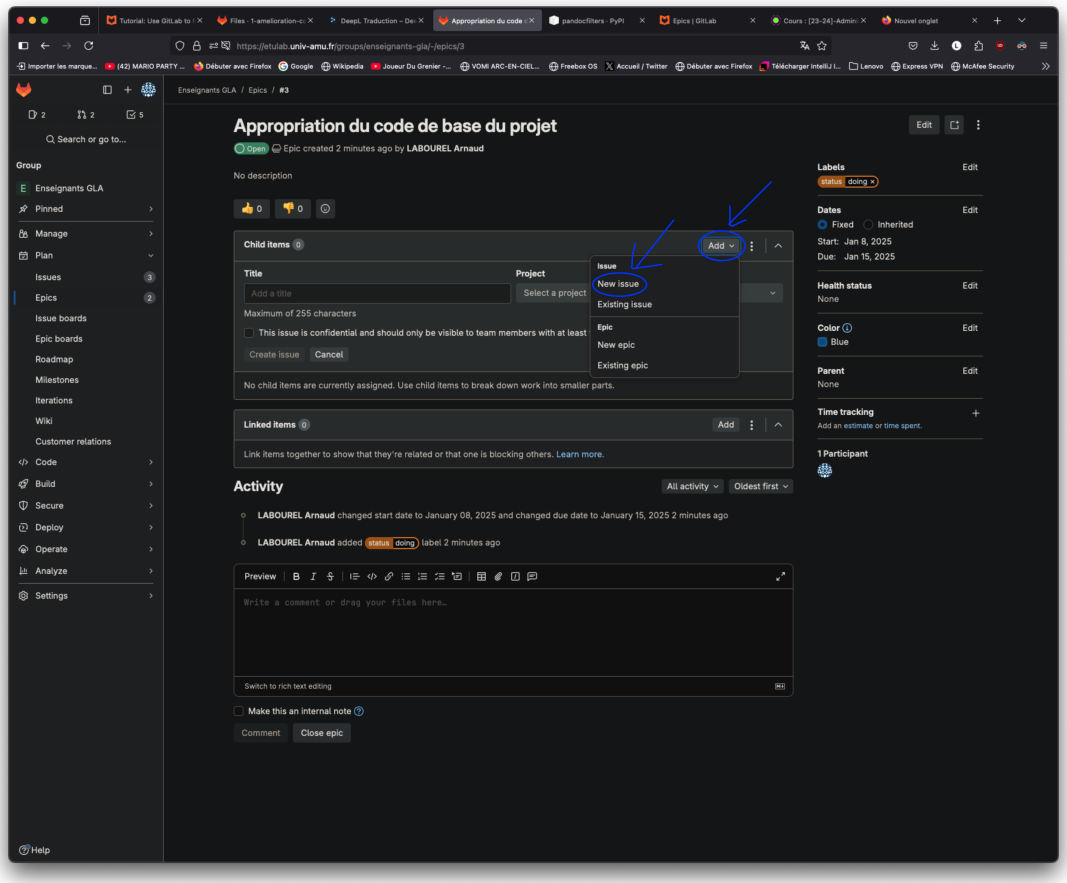
Vous trouverez davantage d'informations sur la manière de gérer les `Epic` dans gitlab dans la documentation Epic de *gitlab*.

Une fois votre `Epic` créé, vous pouvez cliquer sur le nom de l'`Epic` pour l'éditer et rajouter des `Issues` :



Pour rajouter une `Issue`, il faut une fois dans l'`Epic`, aller sur `add` puis `new issue`. Cela vous permet de créer une `Issue` à l'intérieur de l'`Epic`. Pour ce projet, un `Epic` correspondra plus au moins à une *user story* alors que les `Issues` correspondront aux taches pour réaliser l'*user story* (généralement, les `Epics` correspondent plutôt à plusieurs *user stories* dans des projets plus conséquents, mais pour notre cas avec un projet sur 6 semaines la granularité d'*user story* est la plus adaptée).

Une `Issue` correspondra donc à une tâche sur un des deux dépôts. Pour créer une `issue`, il faut cliquer sur `Add` puis `New issue`.



Ensuite, il vous faut donner un `Title` et un `Project` (dépôt git) puis cliquer sur `Create issue`. Vous devez pour cet `Epic` (nommé “Appropriation du code de base du projet”) ajouter deux `Issue` :

— Une `issue` nommé “Appropriation du code du client” associé à votre dépôt client
— Une `issue` nommé “Appropriation du code du serveur” associé à votre dépôt serveur

Une fois l’`issue` créée, vous pouvez l’éditer en cliquant dessus. Vous pouvez paramétrer l’`issue` en donnant :

— Une description en cliquant sur le bouton `edit` ;
— Ajouter une personne à l’`issue` en cliquant sur le bouton `edit` à droite d’`Assignees` (menu de droite) ;
— Ajouter un `Label` avec le bouton `edit` à droite de `Labels` (menu de droite) ;
— Ajouter un `Weight` (poids de la tâche exprimé en heures de travail par exemple) avec le bouton `edit` à droite de `Weight` (menu de droite) ;
— Ajouter une date limite de fin avec le bouton `edit` à droite de `Due date` (menu de droite) ;
— Créer une branche ou une requête de *merge* du dépôt associé en cliquant sur le bouton juste à droite de `Create merge request` et choisissant l’option appropriée.

Vous trouverez davantage d’informations sur la manière de gérer les `Issues` dans gitlab dans la documentation Issues de *gitlab*.

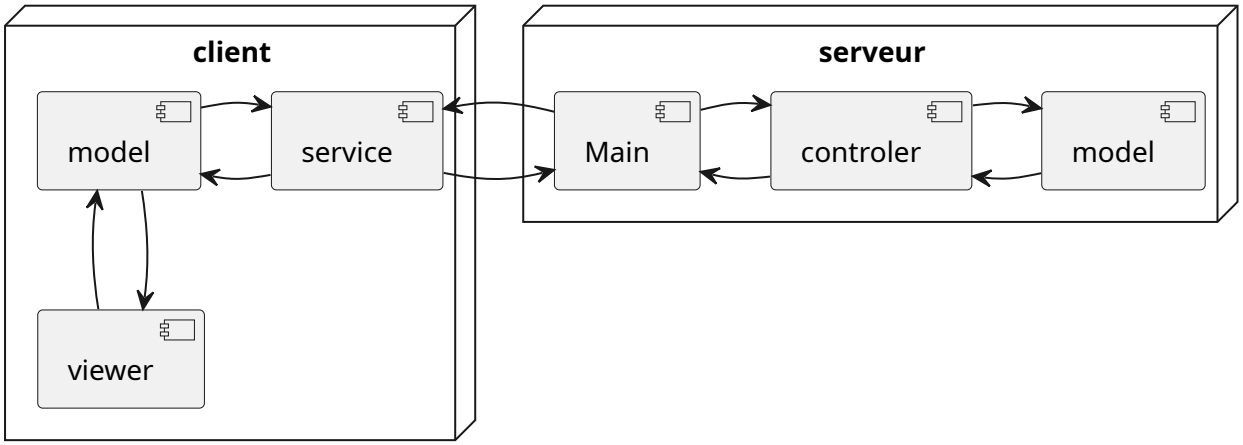
Une fois les issues créées et assignées, vous pouvez commencer à travailler dessus afin de les faire passer de

`status::to do` à `status::doing` puis finalement `status::done`.

De manière optionnelle, vous pouvez aussi créer des `Tasks` à l'intérieur des `Issues` afin de découper encore plus le travail. Vous trouverez les informations sur la manière de gérer les `Tasks` dans gitlab dans la documentation Tasks de *gitlab*.

4.3 Explication du code du dépôt

Pour le moment, l'architecture du projet est la suivante :



Côté client, nous avons les composants suivants :

- Le *package* `model` contient le modèle côté client. Pour le moment, ce modèle se contente d'appeler directement le service requérant le serveur.
- Le *package* `service` contient le code de gestion du requêteur HTTP qui envoie les requêtes HTTP au serveur. Pour le moment, le service ne gère que les requêtes de *get* permettant de récupérer tous les créneaux compris entre deux dates.

Le projet utilise les classes et bibliothèques suivantes :

- Javalin comme framework pour le code HTTP du serveur ;
- HTTPRequest comme classe pour créer les requêtes HTTP côté client ;
- Les classes LocalDateTime, LocalDate et LocalTime pour représenter respectivement une date avec une heure, une date et une heure dans la journée. Ces classes ne gèrent pas la notion de fuseau horaire.
- Les classes Period et Duration pour représenter respectivement une durée en jours et en secondes.
- JUnit 5 pour les tests unitaires.
- AssertJ pour les assertions dans les tests.
- Apache Log4j 2 comme *logger* côté client.
- Simple Logging Facade for Java (SLF4J) comme interface de *logger* côté serveur.
- Jackson pour la sérialisation et désérialisation des objets en *json* côté client et côté serveur.

5 Amélioration de la gestion des requêtes de l'API

Un des premières tâches que vous avez à faire est d'améliorer la prise en compte des requêtes de l'API. N'oubliez pas de créer l'`Epic` et les `Issues` correspondant à votre travail afin de répartir les tâches et faire le suivi. Pour le moment, la seule requête fonctionnelle est celle du *get* permettant de récupérer tous les créneaux compris entre deux dates. On vous demande donc de rajouter du code pour gérer les requêtes suivantes :

- mise à jour (*update*) d'un créneau (*slot*) avec identifiant : HTTP PUT (identifiant dans la route de la requête et nouveau créneau dans le corps de la requête en json). L'*update* n'est acceptée que si la version du nouveau créneau est égale à celle du créneau plus un. Cela permet de ne pas accepter une modification qui en effacerait un autre sans que le client n'ait récupéré la modification potentiellement effacée, car le client va toujours demander l'*update* d'un créneau en incrémentant d'un le numéro de version.
- récupération (*get*) d'un créneau avec identifiant : HTTP GET (identifiant dans la route de la requête)
- création (*create*) d'un créneau : HTTP POST (identifiant dans la route de la requête et nouveau créneau dans le corps de la requête en json)
- suppression (*delete*) d'un créneau : HTTP DELETE (identifiant dans la route de la requête)

Côté client, il vous faudra modifier la classe `fr.univ_amu.m1info.client.service.dao.SimpleCalendarServiceDAO`. Côté serveur, il vous faudra modifier la classe `fr.univ_amu.m1info.server.Main` et sans doute créer une classe séparée pour gérer la gestion des requêtes du serveur. Il faudra aussi modifier les classes `fr.univ_amu.m1info.server.controler.CalendarController` et `fr.univ_amu.m1info.server.model.Calendar` afin de permettre la gestion des nouvelles requêtes.

Documentation et tutoriels :

- Côté client :
  - Documentation HttpClient
  - tutoriel sur l'utilisation de Java HttpClient
- Côté serveur : Documentation de javalin

6 Amélioration du modèle côté client

Une deuxième tâche consiste à améliorer le modèle côté client afin de minimiser les requêtes au serveur. L'idée est de stocker dans le modèle client les données des créneaux afin de ne pas les redemander à chaque fois. On mettra donc en place un système de cache en ne redemandant les créneaux d'une période qu'au bout d'un certain temps ou bien après une modification.