21 Mai 2018

UNIVERSITÉ DE LORRAINE

# Projet Génie Logiciel

-

## Application de calcul d'itinéraire dans le métro parisien

### Sprint n°6

15 au 21 Mai 2018

Équipe :
Rodolphe Aubry, Laurene Cladt, Charlotte Fernandes, Benjamin Rath

Enseignant:
Olivier Perrin

# Sommaire

# Sprint

---

## Liste des tâches

| Tâches | Estimation | Développeurs | Terminée | Temps réel |
|---|---|---|---|---|
| Modéliser l'itinéraire avec le moins de changement de lignes | 1j | Laurene | ✔ | 1j |
| Affichage de l'itinéraire avec le moins de changements dans le main | 0.25j | Benjamin | ✔ | 0.25j |
| Refactoring | 2j<br>0.5j | Charlotte<br>Rodolphe | ✔<br>✔ | 1.5j<br>0.5j |
| Passage de Sonarqube | 0.125j | Rodolphe | ✔ | 0.125 |
| Affichage des horaires dans le main | 0.25j | Benjamin | ✔ | 0.25j |
| Afficher les prochains passages des rames dans l'itinéraire | 0.5j | Laurene | X | / |

## Modélisation de l’itinéraire avec le moins de changements de lignes

La dernière partie de l’itinéraire à modéliser était celle avec le moins de changement de lignes. Cela nous a posé quelques problèmes, car notre implémentation de l’algorithme de Dijkstra ne nous permettait pas de comparer les changements entre plusieurs itinéraires. Nous avons donc décidé d’implémenter un second type d’algorithme grâce à de nouveaux objets, nous permettant d’utiliser l’algorithme de Dijkstra en considérant les lignes de métro comme sommets. Pour cela, nous avons légèrement revu la modélisation de l’itinéraire en lui-même en implémentant des classes abstraites. Nous avons aussi dû réfléchir à comment renseigner la direction et le temps de parcours en considérant deux stations sur une même ligne.

## Afficher les prochains passages des rames dans l’itinéraire

De par la modélisation des horaires et de l’itinéraire ainsi que leur affichage, il nous était difficile d’indiquer le prochain passage d’une rame directement dans l’itinéraire, bien que cela soit possible. Cela aurait augmenté la complexité de l’application ou demandé une refonte des méthodes concernant l’itinéraire et les horaires, ce qui était inenvisageable. Étant donné que notre application permet malgré cela d’afficher les horaires de passage des rames, nous avons fait le choix de laisser cette fonctionnalité de côté.

## Affichage de l’itinéraire avec le moins de changements dans le main

Afin que l’utilisateur ait la possibilité d’effectuer un trajet avec le moins de changements (en cas de charges lourdes à porter par exemple), il était nécessaire de retranscrire la méthode dans l’application lors de son choix de type d’itinéraire.

## Affichage des horaires dans le main

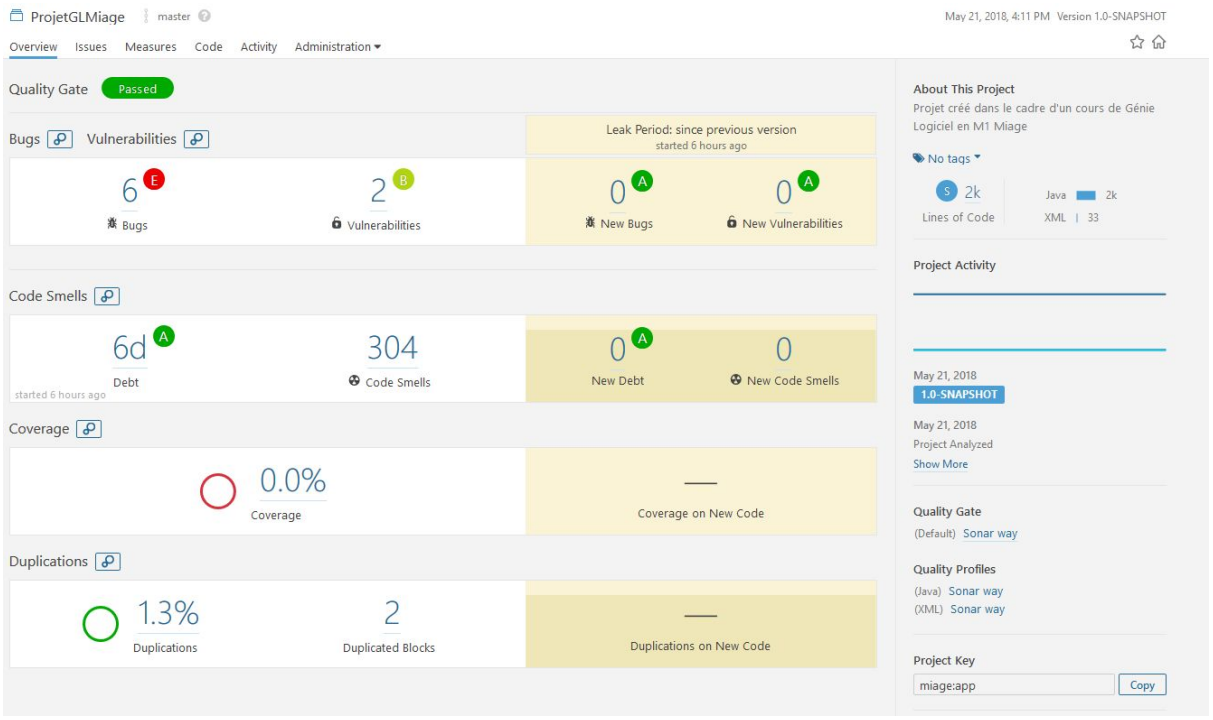
Lorsque l'utilisateur souhaite estimer l'arrivée d'un métro sur une station, il est maintenant possible d'afficher les horaires des prochains métros de passage sur la station d'une ligne.

## Refactoring

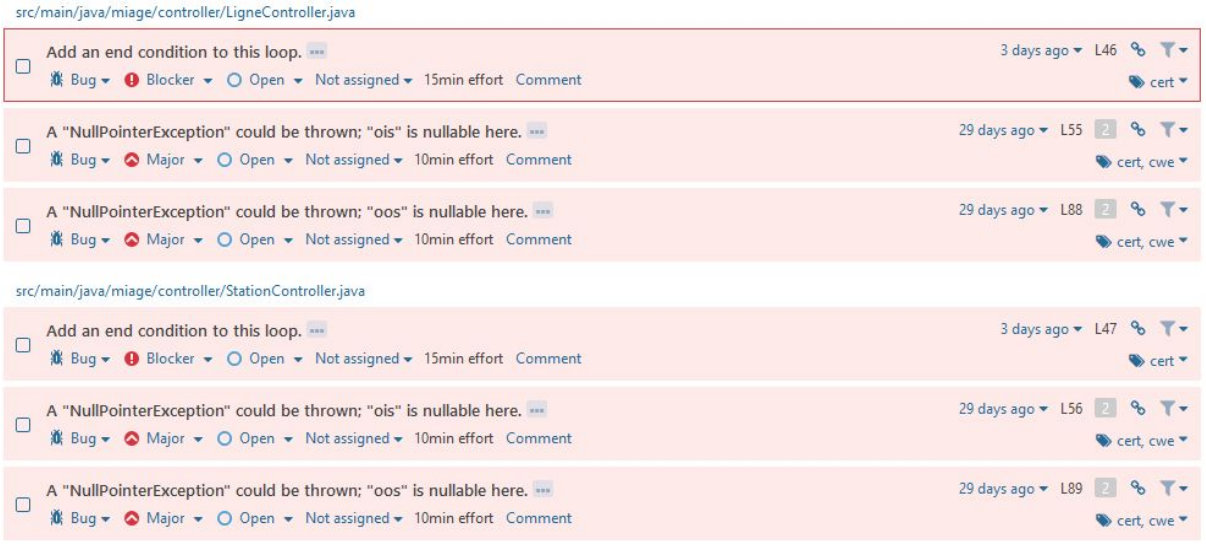
Étant à la dernière itération du projet, nous avons décidé de consacrer une partie de notre travail sur le refactoring de l'application. Tout d'abord, nous nous sommes rendu compte que l'on crée plusieurs objets *LigneController* et *StationController*, or il était plus logique d'en utiliser un seul pour travailler sur le même objet à chaque fois. Pour cela, nous avons dû rendre les méthodes de ces contrôleurs statiques pour pouvoir les utiliser dans tout le code.
Ensuite, dans *Main*, nous avions beaucoup de redondances, nous utilisions exactement les mêmes lignes de code dans plusieurs de nos méthodes. Nous avons donc, pour éviter toute répétition, centralisé ces lignes de code dans des méthodes.

## Passage de SonarQube

Le but principal de SonarQube est de fournir une analyse complète de la qualité d'une application en fournissant des statistiques sur des projets. Ces données permettent ainsi d'évaluer la qualité du code et d'en connaître l'évolution au cours du développement. Dans le cas de notre projet, nous arrivons au tableau de bord ci-dessous :

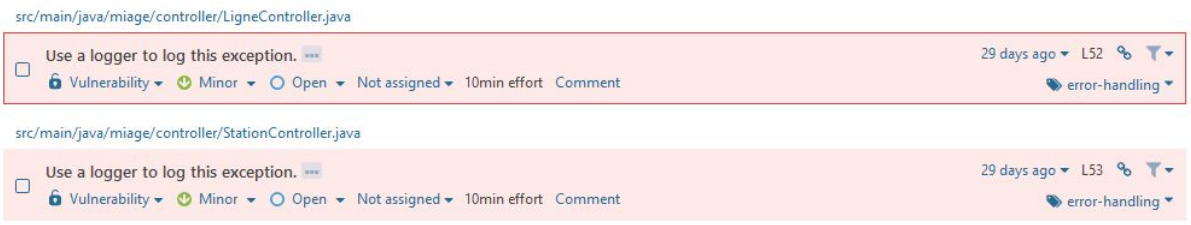



On peut remarquer que notre projet obtient la note de A au contrôle qualité, mais qu'il y a toutefois des bugs considérés comme importants ainsi que deux failles de sécurité.




Voici les 6 "bugs" affichés par SonarQube.

Les vulnérabilités affichées sont celles-ci :



SonarQube nous affiche le temps estimé à la résolution de l'erreur et ces vulnérabilités sont mineures.

Toutes ces erreurs ne demandent pas un grand investissement de temps et seraient corrigibles dans une éventuelle future itération.

# Burndown chart

# Conclusion

Ce projet a été une expérience nouvelle pour chacun de nous. Nous n'avions jamais effectué un projet en étude qui nous demandait de travailler de manière régulière durant 6 semaines.
D'un point de vue organisationnel, le développement en sprint a permis de concevoir une application de façon ordonnée tout en permettant à chacun de participer de manière égale au projet. De plus, avec l'outil github nous avons pu mieux nous organiser et éviter tout conflit. Nous avons également appris à appliquer les différentes commandes git dans un cadre réel, ce qui nous a permis d'apprendre à mieux maîtriser cet outil.
L'une des difficultés que nous avons rencontrées, dans les deux dernières itérations, a été l'implémentation des itinéraires. En effet, nous aurions dû réfléchir aux itinéraires avant la modélisation du réseau, car nous avons été bloqués et nous avons développé un code qui aurait pu être simplifié avec une autre modélisation.

Au niveau de l'application, nous avons réussi à implémenter toutes les fonctionnalités demandées en faisant des choix appropriés pour chacunes d'entre elles. Nous avons également ajouté des tests et complété un rapport à l'issue de chaque itération afin de garder une trace du travail effectué.