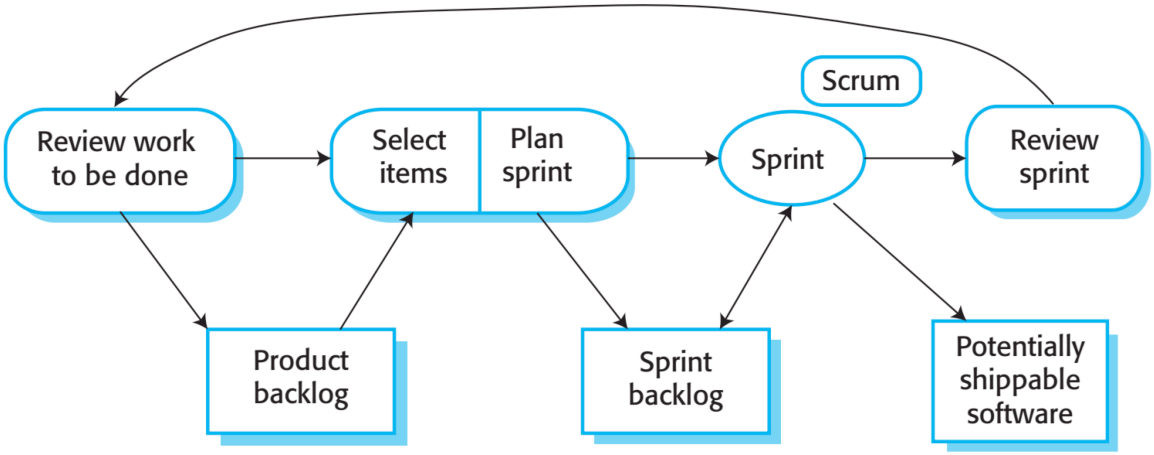
INGÉNIERIE LOGICIELLE (SPRINT 3)



Votre mission est de planifier et réaliser un troisième sprint de 4 semaines et de rapporter vos expériences lors de la mise en œuvre des pratiques agiles listées ci-dessous. Prenez des captures d'écran et essayez de documenter raisonnablement vos progrès. Les deux dernières semaines du semestre vous permettront de consolider votre rapport et de finaliser votre rendu.

PRATIQUES AGILES

Conception incrémentale

Utilisez des outils de modélisation du système (diagramme de cas d'utilisation, diagramme de classes, diagramme de séquences, etc.) pour concevoir votre générateur de site statique. Un nouvel acteur (le système de fichier) est introduit par la story « génération du site statique à la volée ». Introduisez cet acteur dans votre diagramme de cas d'utilisation et mesurez l'impact de ce changement dans votre architecture de manière à organiser le travail et le refactoring du code existant. Introduisez une abstraction simple qui encapsule et cache la complexité de la WatchService API.

Refactoring

Utilisez des outils de refactoring (IntelliJ ou autre) pour faire évoluer votre générateur de site statique. Discutez des mesures prises lors de ce sprint pour éviter les conflits associés au refactoring lors de la collaboration au sein d'un groupe.

Stories

Utilisez votre conception pour subdiviser la story « génération du site statique à la volée » en sous tâches. Assignez les tâches de travail aux membres de votre équipe. Estimez la durée de chaque tâche en trois points (optimiste, pessimiste, attendu). Effectuez la tâche et rapportez le temps pris dans la pratique. Discutez des éventuels deltas entre temps estimés et temps observés.

Tests d'intégration et tests systèmes

Ajoutez des tests d'intégration et des tests systèmes à votre générateur de site statique et assurez qu'ils couvrent les parties importantes du système (code coverage).

Automatisation

Automatisez la génération d'un ou deux rapports liés aux stories javadoc, code coverage ou code quality.

Commit early, commit often

Committez régulièrement vos changements et décrivez les précisément à l'aide de messages.

BACKLOG PRODUIT

Les stories suivantes sont ajoutées à votre backlog produit.

Génération du site statique à la volée

En tant qu'utilisateur, j'aimerai que les commandes build et serve puissent régénérer le site statique à la volée, c'est-à-dire de manière automatique lorsque des changements sont effectués sur le système de fichier. Pour cela, ajoutez une option --watch aux commandes build et serve et assurez-vous que leur exécution ne se termine pas à la fin du premier build.

Les générateurs de site statique Jekyll[1] et Hugo[2] disposent tous deux d'une option --watch. La WatchService API[3] de la librairie standard Java permet d'écouter les changements effectués sur le système de fichiers.

Publication du site dans un répertoire distant

En tant qu'utilisateur, j'aimerai que la commande publish permette de publier le dossier build dans un répertoire distant accessible par Internet. Le choix technologique lié à l'accès au répertoire distant est libre. Il peut par exemple s'agir d'un dossier accessible avec git[4], ssh[5], cifs ou webdav.

Manuel utilisateur

Rédigez un manuel utilisateur reproductible qui guide l'utilisateur lors de la création d'un site statique avec votre générateur.

Javadoc

Configurez Maven de manière à pouvoir publier une javadoc au format HTML[6]. Ajoutez la Javadoc à votre release.

Code coverage

Configurez Jacoco[7] ou un outil équivalent pour calculer le degré de couverture de votre code par des tests unitaires et d'intégration. Ajoutez les données de couverture à votre rapport.

Code quality

Configurez LGTM[8] , SonarQube[9] ou un outil équivalent pour détecter des bugs et des vulnérabilités dans votre code. Expliquez en quoi l'un de ces outils vous aura permis d'améliorer vos pratiques de développement.

Code benchmarking

Utilisez JMH[10] pour mesurer les performances de votre compilateur markdown ou de votre moteur de rendu. Utilisez VisualVM[11] pour profiler une exécution de la commande build. Incluez ces données dans votre rapport et discutez d'une éventuelle optimisation (sans nécessairement l'implémenter).

---

1 https://github.com/jekyll/jekyll-watch
2 https://gohugo.io/getting-started/usage/
3 https://docs.oracle.com/javase/tutorial/essential/io/notification.html
4 https://projects.eclipse.org/projects/technology.jgit
5 http://www.jcraft.com/jsch/
6 https://maven.apache.org/plugins/maven-javadoc-plugin/
7 https://www.eclemma.org/jacoco/
8 https://lgtm.com/
9 https://www.sonarqube.org/
10 https://www.baeldung.com/java-microbenchmark-harness
11 https://visualvm.github.io/