Démarche CI/CD

1. GitHub Actions

a. Tests du code

Les GitHub Actions sur les tests sont décrites dans les fichiers tests-front.yml et tests-back.yml, qui correspondent respectivement aux tests sur le Front-end et le Back-end. Les tests permettent de vérifier que le code est toujours fonctionnel après l'ajout de corrections ou modifications par un membre de la communauté. Le rapport de couverture permet de vérifier que les tests couvrent bien la majorité du code du projet.

Ces GitHub Actions ont deux principaux objectifs :

- L'exécution et la validation des tests
- La génération du rapport de couverture

Les étapes de ces Actions sont les suivantes :

- Le déclanchement. L'action débute lors d'une « Pull Request » pour la fusion de la branche modifiée sur la branche principale.
- Le « Checkout », ce qui correspond à la récupération du code du projet dans l'environnement d'exécution de l'action.
  ⇨ « actions/checkout@v4 »
- La configuration, ce qui correspond à la configuration de l'outil qui va construire ou exécuter le code. Pour le Front-end, il s'agit de Node.js et pour le Back-end, il s'agit de Java 11.
  ⇨ « actions/setup-node@v4 » et « actions/setup-java@v4 »
- L'installation des dépendances pour la partie Front-end uniquement, avec par exemple Angular ou RxJS.
  ⇨ « npm ci »
- La construction du projet et l'exécution des tests, avec la génération du rapport de couverture.
  ⇨ « npm run test:prod » et « mvn clean install »
- L'archivage du rapport de couverture, afin qu'il puisse être téléchargé par la suite.
  ⇨ « actions/upload-artifact@v4 »

b. Qualité du code

Les GitHub Actions sur l'analyse de la qualité du code sont décrites dans les fichiers sonar-front.yml et sonar-back.yml.
L'évaluation de la qualité du code repose sur l'analyse de divers critères, incluant la détection de bugs, le respect des bonnes pratiques, la mise en évidence de vulnérabilités de sécurité, ainsi que la couverture du code. Cette analyse sera

effectuée en utilisant l'outil SonarCloud.
Les étapes de ces Actions sont les suivantes :

- Le déclanchement. L'action débute lors d'une « Pull Request » pour la fusion de la branche modifiée sur la branche principale.
- Le « Checkout », ce qui correspond à la récupération du code du projet dans l'environnement d'exécution de l'action.
  ⇨ « actions/checkout@v3 »
- La configuration de Java 17, pour le Back-end uniquement.
  ⇨ « actions/setup-java@v3»
- La mise en cache des packages SonarCloud et Maven, pour le Back-end uniquement. Cela permet d'éviter de télécharger les dépendances à chaque fois que l'action est exécutée.
  ⇨ « actions/cache@v3 »
- La construction du projet et l'exécution du scan sur le code par SonarCloud (après authentifications).
  ⇨ « SonarSource/sonarcloud-github-action@master » et « mvn -B verify … »

c. Déploiement des images Docker

Les GitHub Actions sur le déploiement des images Docker sont décrites dans les fichiers docker-front.yml et docker-back.yml.
Les images Docker encapsulent de manière portable et reproductible les parties de l'application avec toutes leurs dépendances. Cela permettra à la communauté de mettre en production facilement l'application BobApp sur la plateforme de leur choix (on-premise, cloud, ordinateur, etc.).

Les étapes de ces Actions sont les suivantes :

- Le déclenchement. L'action se produit lors d'un "Push" sur la branche principale, par exemple lors de la fusion d'une branche modifiée avec la branche principale. De plus, les tests et la qualité du code doivent être validés pour pouvoir fusionner la branche modifiée avec la branche principale.
- Le « Checkout », ce qui correspond à la récupération du code du projet dans l'environnement d'exécution de l'action.
  ⇨ « actions/checkout@v4 »
- L'authentification sur Docker Hub, pour permettre le déploiement.
  ⇨ « docker/login-action@v3 »
- La configuration de Docker Buildx. Il s'agit du moteur de création des images Docker.
  ⇨ « docker/setup-buildx-action@v3 »
- La construction et le déploiement des images Docker.
  ⇨ « docker/build-push-action@v5 »

2. Seuils de qualité du code - KPI

Valeur minimale de couverture des tests à 80%
Le taux de couverture de tests mesure la proportion du code source testée par rapport au code total.
Une valeur minimale de 80% permet de s'assurer que les tests réalisés sont utiles et qu'ils couvrent la majorité du projet.

Pas de « New Blocker Issues »
Les "New Blocker Issues", ou nouveaux problèmes critiques, sont des problèmes de code présentant une gravité élevée pouvant entraîner des conséquences sérieuses en production. Ces problèmes peuvent comprendre des vulnérabilités de sécurité, des bugs critiques, des violations de règles essentielles de qualité du code, etc.
Ce KPI (Key Performance Indicator) permet de garantir que le code ajouté ou modifié par la communauté ne comporte pas de failles majeures.

Taux de duplication du code inferieur à 3%
Le taux de duplication du code mesure la proportion de code source identique ou similaire répétée dans un projet. Il fournit une indication de la qualité du code en identifiant les sections susceptibles de nécessiter une refactorisation ou une consolidation.
Une valeur maximale à 3% permet de garder le code compréhensible, maintenable et évolutif tout en laissant une petite marge de manœuvre aux développeurs.

3. Analyse des métriques

L'analyse de la qualité du code de l'application BobApp repose sur un ensemble de critères mesurables, désignés sous le terme de métriques.

La liste suivante décrit les principales métriques et leur méthode d'évaluation :

- La fiabilité, évaluée sur une échelle de A à E, à partir du nombre de bugs et de leur gravité.
- La maintenabilité, estimée sur une échelle de A à E, à partir du nombre de « mauvaises pratiques de codage », appelées Code Smells, et du ratio de dette technique induite. La dette technique correspond au temps nécessaire pour résoudre les Code Smells. Le ratio de dette technique correspond au ratio entre le coût de résolution des Code Smells et le coût de développement global du projet.
- La sécurité, calculée sur une échelle de A à E, à partir du nombre de vulnérabilités de sécurité et de leur gravité. Dans ce cas, il s'agit des vulnérabilités de sécurité avérées par SonarCloud.
- La revue de sécurité, jugée sur une échelle de A à E, à partir du nombre de vulnérabilités potentielles de sécurité non examinées. Dans ce contexte, il s'agit de vulnérabilités potentielles de sécurité identifiées par SonarCloud.
- La couverture des tests est exprimée en pourcentage et représente le rapport entre le nombre de conditions testées et le nombre total de conditions à tester dans le code.
- La duplication du code est exprimée en pourcentage et représente le rapport entre le nombre de lignes de code dupliquées et le nombre total de lignes de code dans le projet.
- La complexité du code correspond au nombre de complexités cyclomatiques. C'est-à-dire, le nombre de chemins d'exécution possible dans le code. Ainsi, chaque fois qu'une fonction comporte plusieurs possibilités, comme le choix entre vrai ou faux, cela augmente la complexité de 1.

Le tableau ci-dessous récapitule les valeurs obtenues pour le Front-end et le Back-end vis-à-vis des métriques énoncées.

Métrique	Front-end	Back-end
Fiabilité	A (avec 0 bug)	D (avec 1 bug)
Maintenabilité	A (avec 5 Code Smells)	A (avec 8 Code Smells)
Sécurité	A (avec 0 vulnérabilité)	A (avec 0 vulnérabilité)
Revue de sécurité	E (avec 3 failles non examinées)	E (avec 2 failles non examinées)
Couverture	0% de couverture	38,8% de couverture
Duplication	0% de duplication	0% de duplication
Complexité	18 complexités cyclomatiques	18 complexités cyclomatiques

4. Retours utilisateur

Les retours utilisateurs les plus pertinents sont les suivants :

- « Je mets une étoile car je ne peux pas en mettre zéro ! Impossible de poster une suggestion de blague, le bouton tourne et fait planter mon navigateur ! »
- « #BobApp j'ai remonté un bug sur le post de vidéo, il y a deux semaines et il est encore présent ! Les devs vous faites quoi ???? »
- « Ça fait une semaine que je ne reçois plus rien, j'ai envoyé un email il y a 5 jours mais toujours pas de nouvelles… »

Ces retours mettent en avant les problématiques suivantes, classées par ordre de priorité :

- Une impossibilité d'accès à l'application. Ce problème est signalé par le 3ème avis et est le plus prioritaire si l'application n'est disponible pour aucun utilisateur.
- Une impossibilité d'ajouter du contenu à l'application, blague et vidéo. Ces problèmes sont moins prioritaires car ils n'empêchent pas l'application de fonctionner.
  Le problème concernant le post de blague est peut-être prioritaire par rapport à celui du post de vidéo car il touche à la fonctionnalité fondamentale de l'application. La résolution de l'incapacité d'ajouter des vidéos peut être effectuée dans un second temps, étant donné qu'il s'agit probablement d'une fonctionnalité additionnelle.