Compte-rendu DevOps

Ben Moussa Ahmed – Rozelaar Marceline – Imbrogno Tino

Front-end

Unit tests

Nous avons réalisé plusieurs tests unitaires pour les composants Angular ainsi que pour toutes les classes typescript associées. Les fichiers de test ont pour extension `.spec.ts` . Dans le cadre de notre projet, nous avons opté pour l'utilisation de Jasmine et Karma pour la rédaction de ces derniers. Très utilisé dans l'environnement Angular, le framework Jasmine offre une intégration transparente avec une prise en charge de fonctionnalités telles que les injections de dépendances (services) et sa syntaxe de tests clairs et compréhensibles.

Jasmine intègre des fonctionnalités de 'mocking' et 'spies' idéales pour simuler des comportements spécifiques de notre code.

De plus, l'utilisation de Karma en conjonction de Jasmine facilite l'exécution des tests sur les différents navigateurs ainsi que la génération de rapport de couverture de code. Cette solution est complète et garantit la qualité et la stabilité du code.

D'autres solutions étaient à notre portée, dont l'utilisation du framework Jest, très populaire aussi, nous avons trouvé plus judicieux d'utiliser l'outil développé par l'équipe Angular pour notre projet. Mais cette option était tout à fait envisageable.

E2E testing

Concernant les tests de bout-en-bout (E2E tests) nous avons choisi d'utiliser l'outil Cypress. Ce dernier offre une solution performante pour ce type de test, en effet il propose tout d'abord une interface utilisateur interactive avec laquelle nous pouvons voir en temps réel l'exécution de nos différents tests sur notre jeu de Sudoku. De plus, il est rapide et facile à configurer et à exécuter. Sa syntaxe est claire et compréhensive.

Nous avons été contraint par le projet d'utiliser cet outil que nous trouvons d'ailleurs très efficace et performant. Il était possible aussi de s'intéresser à Selenium qui est une solution plus ancienne que Cypress mais sa configuration pourrait demeurer plus complexe.

Linting

Pour le linting nous avons utilisé le module ESLint car c'était proposé par le projet, mais aussi pour ces extensions spécifiques à angular qui permettent d'éviter des mauvaises pratiques. De plus, ce module permet de corriger certaines de ces fautes automatiquement, mais il a quand même fallu en corriger un peu plus de la moitié à la main.

Coverage

Comme évoqué dans la partie ‘Unit tests’, la couverture de code a facilement pu être implémentée en paramétrant le fichier `karma.conf.js`. En effet, Karma dispose d’un outil de couverture et permet trivialement de retourner des résultats dépendant des lignes et fonctions couvertes.

Back-end

Du coté back-end nous avons gardé l’utilisation de checkstyle pour le code convention et spotbugs pour le linting mis en place lors du projet CPOO. Cependant nous avons rajouté Jacoco pour mesurer le code coverage et motiver la création de tests. On a choisi Jacoco car ça a été suggéré dans le cours et sa mise en place était simple.

Docker

Dans l'étape de déploiement, notre processus repose sur l'utilisation de deux images Docker distinctes : l'une dédiée au frontend et l'autre au backend. Pour orchestrer la gestion de ces images, nous avons opté pour MicroK8s, une implémentation de Kubernetes (k8s) réputée pour sa légèreté. En configurant soigneusement les routes et les ports au sein du script de déploiement, nous mettons en œuvre le déploiement fluide de l'application. Une fois déployée, celle-ci devient accessible à l'adresse 127.0.0.1.