Traçabilité logicielle des GUIs d'applications web React avec l'infrastructure de traçabilité et testabilité logicielles SoftScanner

Stage proposé en collaboration entre le laboratoire LIRMM (http://www.lirmm.fr/) et l'entreprise Berger Levrault (https://www.berger-levrault.com/fr/)

Encadrants:

- Abdelhak-Djamel Seriai, MCF Dept Info, FDS, UM, http://www.lirmm.fr/~seriai/, seriai@lirmm.fr
- Bachar Rima, Ingénieur d'études au LIRMM, équipe MAREL, et vacataire à la FDS, UM, rima@lirmm.fr

Contexte

Dans les grandes entreprises, la majorité du budget des logiciels est consacré à la maintenance et l'évolution des logiciels existants plutôt qu'au développement de nouveaux logiciels, ce dernier étant assez coûtant et risqué. Parmi les activités de maintenance et d'évolution logicielles, on note le débogage, l'optimisation, les tests, le profilage des utilisateurs, la sécurité, etc. La mise en œuvre de ces activités peut être réalisée par divers stratégies, dont l'exploitation des traces (*i.e., les logs*), devenant de plus en plus adoptée et systématisée dans le monde du génie logiciel.

Les logs sont parfois les seules ressources disponibles pour la compréhension effective d'un système complexe à grand échelle. Assurer la qualité des informations y incluses est donc d'une importance primordiale, impactant par la suite les décisions prises par les développeurs pour la maintenance/évolution de leur système étudié. De plus, vu que les logs sont générés par des instructions de traçage (*Log Printing Statements (LPSes)*) instrumentant le code source du système étudié, assurer la qualité de ses dernières est également indispensable.

Plusieurs solutions systématiques ont été déjà proposées pour la gestion des logs générés. Cependant, l'instrumentation du code source avec des instructions de traçage demeure coûteuse et sujette à l'erreur humaine, étant ad-hoc, majoritairement manuelle, et surtout dépendante de l'expertise des développeurs qui s'en chargent.

Pour pallier ce problème, le projet SoftScanner, une collaboration entre le LIRMM et l'entreprise Berger-Levrault (BL), a été lancé comme infrastructure dédiée à l'automatisation partielle de l'instrumentation de code source avec des LPSes, selon une/plusieurs objectifs de traçage.

L'infrastructure de SoftScanner est constituée d'un ensemble de petits services, appelés microservices, servant chacun comme un mini-projet et interagissant les uns avec les autres par le biais d'APIs REST. Un client interagit ainsi avec SoftScanner en fournissant un projet à analyser et en choisissant les objectifs de traçage désirés, pour ensuite lancer le processus d'instrumentation. La requête est traitée par un sous-ensemble des microservices qui s'échangent les résultats de leurs traitements entre eux en JSON par le biais de leurs APIs REST. À la fin, un lien vers le projet instrumenté est retourné au client qui pourra ainsi le télécharger.

L'un des microservices de SoftScanner possède comme objectif le traçage des widgets graphiques d'une application web. Le microservice a été implémenté pour les applications SPA (Single-Page Applications) créées par Angular.

Objectif et étapes

L'objectif de ce projet TER M1 est de fournir une extension du microservice de traçage des widgets graphiques d'une application web au monde des applications web développées avec la technologie React. Il est structuré en 3 tâches obligatoires et 3 tâches bonus :

1) Conception, implémentation, et test d'une approche permettant d'instrumenter les widgets graphiques d'une application web créée avec React :

- Étudier et comprendre l'approche utilisée pour le traçage des widgets graphiques d'une application web créée en Angular.
- Étudier et comprendre l'architecture des applications créées en React.
- Trouver et utiliser une bibliothèque de traçage permettant de construire les LPSes à injecter et configurer les destinations des logs générés (*i.e., consoles, fichiers, base de données, etc.*)
- Trouver et utiliser un module permettant d'extraire et de manipuler un modèle de représentation du code source d'une application React (i.e., un AST) afin de générer du code instrumenté.
- Concevoir et implémenter les stratégies de traçage.

2) Création d'une API REST permettant de consommer le service de traçage créé.

3) Création d'une interface graphique web du service de traçage créé.

4) (Bonus) Conception, implémentation, et test d'une manière permettant de généraliser l'approche de traçage des widgets graphiques pour toute sorte d'application web, indépendamment de la technologie utilisée (étude bibliographique, ingénierie dirigée par les modèles).

5) (Bonus) création d'une API REST permettant de consommer le service de traçage des applications web Angular.

6) (Bonus) création d'une interface graphique web du service de traçage des applications web Angular.

Pourquoi choisir ce projet :

- Acquérir des compétences certaines en développement logiciel de manière générale (principes de design SOLID, patrons de conception, styles architecturaux, etc.) et en développement web en particulier (Spring, Angular, React, etc.).
- Acquérir des compétences en programmation distribuée et la construction d'APIs REST.
- Découvrir plusieurs thématiques dans le monde du développement logiciel : la rétro ingénierie (*reverse engineering*), la réingénierie (*reengineering*), l'analyse du code source, etc.
- Acquérir des compétences concrètes en traçabilité logicielle et l'analyse dynamique de logiciels.
- Avoir une collaboration avec le monde des entreprises (l'entreprise BL, un des grands éditeurs logiciels en France).
- Etc.

Si vous avez des questions, contactez Bachar Rima rima@lirmm.fr ou Abdelhak-Djamel Seriai seriai@lirmm.fr

Références Bibliographiques

1) Chen, B. (2020). Improving the Logging Practices in DevOps. https://yorkspace.library.yorku.ca/xmlui/handle/10315/37997
2) Li et al. (2020). A Qualitative Study of the Benefits and Costs of Logging from Developers' Perspectives. IEEE Transactions on Software Engineering, 1–1. https://doi.org/10.1109/TSE.2020.2970422
3) Yao et al. (2018). Log4Perf: Suggesting Logging Locations for Web-based Systems' Performance Monitoring. Proceedings of the 2018 ACM/SPEC International Conference on Performance Engineering, 127–138. https://doi.org/10.1145/3184407.3184416
4) Zhao et al. (2017). Log20: Fully Automated Optimal Placement of Log Printing Statements under Specified Overhead Threshold. Proceedings of the 26th Symposium on Operating Systems Principles, 565–581. https://doi.org/10.1145/3132747.3132778
5) Zhu et al. (2015). Learning to Log: Helping Developers Make Informed Logging Decisions. Proceedings of the 37th International Conference on Software Engineering - Volume 1, 415–425.
6) Fielding R. T., & Taylor R. N. (2000). Architectural styles and the design of network-based software architectures. Ph.D. Dissertation. University of California, Irvine. Order Number: AAI9980887.
7) Richardson, C. (2019). Microservices patterns: With examples in Java.
8) Accomazzo Anthony, Murray Nathaniel, and Lerner Ari. 2017. Fullstack React: The Complete Guide to ReactJS and Friends. Fullstack.io.
9) Angular : Développez vos applications web avec le framework JavaScript de Google. Daniel Djordjevic, Sébastien Ollivier, William Klein. Editions ENI. Août 2017.