s12y : Développement d'applications durables

Olivier Le Goaër[1,*]
olivier.legoaer@univ-pau.fr

[1] IUT de Pau et des Pays de l'Adour
Laboratoire LIUPPA

*Le/les auteur(s) avec la marque * sont auteur(s) correspondant(s).*

Thèmes – *Informatique*

Résumé – *Les applications logicielles ont envahi notre vie. Elles sont utilisées pour des actes banals (manger au restaurant universitaire avec IZLY), des démarches administratives (déclarer et payer ses impôts), du réseautage (LinkedIn, Instagram), et bien plus encore. On les trouve sous toutes les formes : web, mobile, desktop, etc. Cette prolifération logicielle implique un déluge d'énergie disponible et une quantité massive de matériel pour la faire fonctionner. Cela pose des questions de soutenabilité : peut-on continuer ainsi longtemps dans un monde fini ? Sans adopter l'approche radicale de « décroissance logicielle » (produire moins d'applications, réapprendre à s'en passer, fabriquer des applications basse technologie), une autre piste est d'intégrer la durabilité (ou « Sustainability », abrégé en « s12y ») dans le logiciel que l'on souhaite construire, comme on le fait déjà pour d'autres attributs de qualité. Cet article court vulgarise les travaux réalisés pour concrétiser le s12y lors du développement d'applications mobiles Android et iOS, et dont l'esprit peut tout à fait être généralisé à d'autres typologies d'applications logicielles.*

Mots-Clés – *logiciel, durable, énergie, carbone*

Section(s) CNU pour cet article –*27.*

Article présenté à l'oral par un(e) doctorant(e) – *NON*

1 Introduction

Selon les principes éprouvés du génie logiciel, le développement d'un « bon » logiciel recouvre un certain nombre d'attributs de qualité, comme sa sécurisation (absence de failles de sécurité), sa fiabilisation (fonctionnement assuré dans différentes conditions) ou encore son internationalisation (adaptation à différentes cultures ou langues). Dans ce dernier cas, les développeurs parlent habituellement de « i18n », abréviation où 18 représente le nombre de caractères entre le « i » et le « n » dans le mot. Dans cette optique, il est d'usage pour un développeur d'extraire du code source informatique tous les textes qui seront affichés littéralement à l'écran, pour pouvoir les charger dynamiquement en fonction de la langue configurée par l'utilisateur final sur son appareil.

Face au réveil écologique du développeur informatique, un nouvel attribut de qualité commence à faire son apparition : la ***sustainability***, abrégé en « s12y ». Le but est de prévoir le code et les ressources pour que le site web (ou l'application mobile) soit le plus durable possible dans un monde physiquement limité. Cela inclut des préoccupations (a) d'efficience énergétique et (b) de longévité des appareils. Le premier point consiste à réduire la quantité d'électricité consommée en phase d'usage (pensez aux applications qui vident votre batterie en quelques heures), tandis que le second point consiste à tenter de freiner le renouvellement prématuré des appareils des utilisateurs, car leur fabrication est la principale source d'émission de GES [1]. Des techno-organismes comme le *World Wide Web Consortium* (W3C) s'intéressent à cette question, par exemple, pour construire un Web mondial plus durable [2].

Dans ce contexte brûlant, la question de recherche est la suivante : à l'instar du i18n, est-il possible de proposer un ensemble de meilleures pratiques de développement pour atteindre un objectif de durabilité/soutenabilité ? En France, le collectif Green IT.fr a été pionnier avec son référentiel des « 115 bonnes pratiques web » (R-WEB). Mon travail a consisté à le compléter par un référentiel de bonnes pratiques pour le développement d'applications mobiles Android/iOS (R-MOBILE), qui représentent une large part des logiciels utilisés au quotidien. L'étape suivante a consisté à accompagner ce référentiel d'une solution d'audit automatisée, libre et gratuite, de haut niveau de maturité [3]. Cette proposition, d'abord baptisée *Creedengo*, puis *ecoCode*, a été référencée dans la boîte à outils gouvernementale pour un numérique éco-responsable [4]. À ce jour, les idées développées initialement pour Android/iOS ont été généralisées à d'autres contextes applicatifs, grâce à la force d'une communauté open source et de plusieurs hackathons.

2 Défauts de code & empreinte carbone

Tous les logiciels peuvent être grossièrement résumés à des lignes de code qui s'exécutent sur un matériel cible. Ces lignes sont écrites par des développeurs plus ou moins expérimentés. Le présent travail sur le s12y repose sur l'idée qu'il existe parfois de meilleures manières d'écrire un code, quitte à faire quelques concessions. À l'échelle de millions de projets à travers le monde et de leurs milliards de lignes de code, les gains potentiels en eqCO2 sont énormes. Mais cela suppose de pouvoir systématiser ces améliorations partout, tout le temps (voir Section 3). Voici deux cas qui font qu'une application mobile n'est pas considérée comme étant « éco-codée ».

2.1 Luminosité de l'écran

Il existe une instruction de programmation spéciale pour forcer la luminosité de l'écran au maximum. L'écran étant le composant le plus consommateur d'un smartphone fonctionnant sur batterie, cette mauvaise pratique entre en contradiction avec la stratégie intelligente de l'OS qui consiste à adapter en permanence la luminosité de l'écran à la luminosité ambiante. Il est alors normal de la remonter comme un défaut énergétique potentiel, afin de limiter la décharge de la batterie. Une fois informée, l'équipe de développement sera chargée de sa résolution, ou pas...

2.2 Poids des images

À côté du code source pur et dur, un autre exemple concerne le format des images empaquetées dans l'application mobile. Le format Webp est nativement pris en charge et fortement recommandé pour diminuer le poids du fichier binaire de distribution (APK pour Android et IPA pour iOS) sur les magasins d'applications. Il est normal de signaler à l'équipe la liste des fichiers images (hors icones de lancement) qui n'ont pas été convertis dans ce format optimisé. C'est une façon de lutter contre l'obésité des applications mobiles, qui nécessitent de plus en plus d'espace de stockage sur nos smartphones (et en mémoire vive) et peuvent nous pousser à les changer pour héberger toujours plus d'apps.

> Le lecteur intéressé par la liste complète des défauts de code pour Android et iOS la trouvera sous forme d'un référentiel open source, en constante évolution [5].

3 Revue de code & amélioration continue

L'amélioration de la qualité en continu consiste à passer en revue toute la structure et le code source des projets Android et iOS, afin de repérer les défauts potentiels vis-à-vis du s12y. Heureusement, il n'est pas nécessaire de confier cette tâche fastidieuse à un expert humain : c'est la revue de code automatisée. Le passage à l'échelle

devient alors possible. Il suffit pour cela de brancher un analyseur dédié en fin de chaîne de production du logiciel. Ci-dessous quelques points saillants concernant le fonctionnement d'un tel outil, lui-même basé sur une solution de qualité de code mondialement utilisée : SonarQube ™.

3.1 Rapport d'analyse s12y

Le principe de l'analyseur est de passer les lignes de code au peigne fin (« scanner du code source » dans le jargon) afin de détecter les défauts s12y. Techniquement, il faut plusieurs scanners, car les projets mobiles impliquent plusieurs langages, et non un seul. Par exemple, un projet Android standard implique du Java ou du Kotlin, du XML et du Groovy. Si un défaut de code est repéré, son numéro de ligne est consigné dans un rapport, accompagné de sa description complète, de sa sévérité et de son temps estimé de correction (Figure 1).

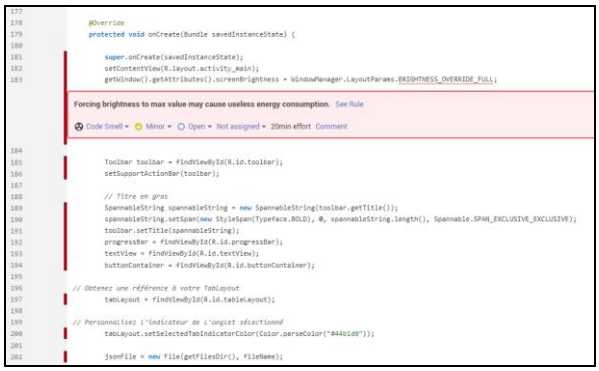

Figure 1 – Défaut détecté (n° de lignes à gauche)

Même s'il n'est pas pris immédiatement en considération, le rapport d'analyse s12y a le mérite de faire prendre conscience aux développeurs que ce qu'ils écrivent peut avoir un impact carbone négatif. Cela encourage les équipes à s'améliorer en continu, sur le projet en cours de développement, ou sur les prochains.

3.2 Porte de qualité s12y

Un concept très intéressant en matière de qualité de code est celui de « porte de qualité ». Cette porte peut être définie sur mesure pour chaque projet. Elle spécifie le seuil de qualité en deçà duquel le logiciel ne peut pas être déployé en production, car jugé encore trop « mauvais ». Ce seuil utilise typiquement le nombre de défauts remontés par l'analyseur (expliqué ci-avant). Ainsi, le chef de projet dispose d'un tableau de bord (Figure 2) avec les projets en cours et l'état de leur porte de qualité : passée (*passed*) ou échouée (*failed*). Appliquée au s12y, cette fonctionnalité prend tout son sens avec la responsabilité numérique des entreprises (RNE) éditrices de logiciel : elles ne sont pas censées mettre en production les logiciels potentiellement néfastes pour la planète !

Figure 2 – Projets « passed » ou « failed »

4 Conclusions

La route vers le s12y est encore longue, mais les travaux présentés dans cet article ont rendu tangibles ce qui n'était au départ qu'un vague objectif à atteindre. Désormais, le développeur logiciel dispose de référentiels de bonnes et mauvaises pratiques et d'un outil pour les détecter à grande échelle. Bien sûr, tout ceci a ses limites : certains critères des référentiels ne font pas nécessairement l'unanimité chez les experts ; quant à l'analyseur de code, il fait remonter des faux positifs et, plus embêtant encore, des faux négatifs. Cela est inhérent à son fonctionnement technique actuel (non détaillé ici), mais l'IA devrait rapidement changer la donne en la matière.

5 Remerciements

Rien de tout cela ne serait possible sans le travail bénévole des membres de la *Green Code Initiative (GCI)*, une association à but non lucratif en faveur des logiciels éco-responsables.

Références

[1] Gouvernement Français, *Proposition de Feuille de route de décarbonation de la filière numérique*, 2023, https://www.ecologie.gouv.fr/sites/default/files/documents/Proposition_feuille_de_route_decarbonation_numerique.pdf

[2] W3C, *Web Sustainability Guidelines (WSG) 1.0*, https://w3c.github.io/sustyweb/, September 2024

[3] Olivier Le Goaër and Julien Hertout, *ecoCode: a SonarQube Plugin to Remove Energy Smells from Android Projects*, The 37th IEEE/ACM International Conference on Automated Software Engineering (ASE), 2022

[4] Gouvernement Français, *Boite à outils numérique écoresponsable*, 2023, https://ecoresponsable.numerique.gouv.fr/publications/boite-outils/fiches/ecocode/

[5] Olivier Le Goaër, *Mobile-specific Best Practices for Sustainable Software (Version 1.5.0)*, 2023 https://github.com/cnumr/best-practices-mobile