UNIVERSITATEA POLITEHNICA DIN BUCUREȘTI
FACULTATEA DE AUTOMATICĂ ȘI CALCULATOARE
DEPARTAMENTUL DE CALCULATOARE

PROIECT SASPS

Avantajele și dezavantajele folosirii serviciului sau bibliotecii pentru respectarea principiului DRY

Mărăcine Mihail-Robert

2023

CUPRINS

1 INTRODUCERE

1.1 Context

În lumea dezvoltării software, unul dintre principiile software fundamentale este DRY (Don't Repeat Yourself) sau în română, "Nu te repeta". Acest principiu reprezintă o axioma esențială în programare, cu scopul de a minimiza redundanța și de a promova reutilizarea codului. Respectarea acestui principiu este crucială pentru dezvoltarea unei aplicații eficiente, ușor de întreținut și citit și scalabilă. DRY spune ca fiecare piesa de cunoastere a sistemului ar trebui sa aiba o singura reprezentare autoritara si neambigua. Fiecare piesa de cunoastere in dezvoltarea unui sistem ar trebui sa aiba o singura reprezentare. Cunoasterea unui sistem este mult mai larga decat doar codul sau. Se refera la schemele de baze de date, planurile de testare, sistemul de construire, chiar si documentatia [3]. Principiul DRY subliniază necesitatea de a evita duplicarea informațiilor și logicii într-un sistem software. Atunci când un fragment de cod se regăsește de mai multe ori într-o aplicație, acest lucru poate duce la divergențe și dificultăți de menținere a coerenței între aceste segmente. Respectarea principiului DRY contribuie la reducerea complexității, creșterea eficienței și menținerea consistenței în cadrul codului. Principiul DRY poate fi încălcat în cazul în care fragmente similare de cod sunt replicate în diverse părți ale aplicației fără o centralizare a funcționalității comune. Aceasta poate fi observată în implementările repetitive ale anumitor secvențe de cod, cauzând divergențe, dificultăți de actualizare și creșterea riscului de erori.

Repetarea codului nu este o soluție absoluta. Există principii care încurajează repetitivitatea codului, cumar fi WET. WET, un acronim opus conceptului DRY, se traduce prin "scrie totul de doua ori" (alternativ, scrie de fiecare data, ne place sa tastam sau pierdem timpul tuturor). Solutiile WET sunt comune in arhitecturile cu mai multe niveluri, unde un dezvoltator poate fi incarcat cu, de exemplu, adaugarea unui camp de comentarii intr-o aplicatie web. Sirul de text "comentariu" ar putea fi repetat in eticheta, tag-ul HTML, intr-un nume de functie de citire, o variabila privata, DDL-ul bazei de date, interogari si asa mai departe. Abordarea DRY elimina aceasta redundanta prin utilizarea de cadre care reduc sau elimina toate aceste sarcini de editare, cu exceptia celor mai importante, lasand extensibilitatea adaugarii de noi variabile de cunoastere intr-un singur loc. Kevin Greer a numit si descris acest principiu de programare.[2]

Contextul e cu atât mai relevant cu cât rata de copiere a codului în mediile de lucru este mare. Comform unui studiu făcut în Software Factories, 80 din codul dintr-o aplicație este duplicat [1]. Inginerii și dezvoltarea evită reimplementarea unei funcționalități deja existente și optimizate, în schimb, găsesc folosință în repetarea codului existent în alte surse pentru a

rezolva probleme punctuale. Din acest punct de vedere este important ca un dezvoltator să creeze cod în minte cu scopul ca mai târziu acesta să poată fi portat sau refolosit. Astfel de moduri de gândire duc la realizarea unor componente precum API-uri sau biblioteci. Utilizarea unei biblioteci permite integrarea funcționalităților standardizate, deja implementate și testate în cadrul unei aplicații. Aceasta poate reduce considerabil cantitatea de cod duplicat și poate simplifica implementarea, oferind funcționalități de bază care sunt disponibile și extensibile. API-urile oferă o modalitate eficientă de comunicare între diferite aplicații sau componente, permițând accesul la funcționalități și date fără a recrea logica deja existentă. Acestea contribuie la modularizarea și abstractizarea codului, reducând duplicarea și facilitând interoperabilitatea între sisteme.

1.2 Scopul și obiectivele cercetării

Scopul acestei cercetări este de a analiza și evidenția avantajele respectării principiului DRY (Don't Repeat Yourself) în dezvoltarea de software, prin evaluarea a două abordări diferite: utilizarea unei biblioteci și utilizarea unui API. Cercetarea va pune accentul pe demonstrarea impactului respectării principiului DRY asupra eficienței și performanței în dezvoltarea unei aplicații.

Modul realizării acestui obiectiv este prin identificarea și analiza beneficiilor aduse de integrarea unei biblioteci în cadrul unei aplicații pentru reducerea codului duplicat. Va fi creată și prezentată o bibliotecă care permite efectuarea unor operații matematice elementare în diferite moduri, și sub anumite reguli. După realizarea acestui obiectiv va urma evaluarea impactului asupra eficienței și ușurinței de întreținere a codului ca urmare a utilizării unei biblioteci. Se va determina performanța prin folosirea unor metrici precum viteza apelurilor, ușurința folosirii metodelor, performanța procesorului si a memoriei. Acesta analiza va fi urmată de investigarea avantajelor respectării principiului DRY prin utilizarea unui API în cadrul unei aplicații software. Pe lângă metricile menționate anterior se va analiza infrastructura necesară, diferențele în modul de proiectare și o parte din avantajele ecosistemului unui microserviciu în care funcționează API-ul.

Se va efectua compararea rezultatelor obținute pentru a evidenția beneficiile și limitările fiecărei abordări în ceea ce privește respectarea principiului DRY. În retrospectivă, se vor prezenta și rezultatele folosirii unui cod duplicat.

2 METODOLOGIE

BIBLIOGRAFIE

[1] Cook Kent John Wiley Sons Greenfield, Short. *Software Factories*. 2004.

[2] Alex Papadimoulis. *The WET Cart*. 2011.

[3] Bill Venners. Orthogonality and the dry principle a conversation with andy hunt and dave thomas, part ii.