**Folosire serverului Jenkins pentru a porni SonarQube folosind un tool in c#**

**Descrierea sistemului**

Sistemul prezentat, se va folosi de serverul Jenkins pentru a putea porni SonarQube pe diferite proiecte. In serverul Jenkins se poate contrui o interfata cu campuri de intrare unde se introduc datele diferitor proiecte software. Acestea se prelucreaza si se salveaza intr-un fisier xml folosind un tool in C#. Datele salvate apar automat pe interfata din Jenkins. Aici utilizatorul poate alege pe ce proiect sa incepa analiza de cod cu SonarQube.

**Cerinte functionale si nonfunctionale**

Ca cerinte functionale, sistemul trebuie sa salveze datele din interfata Jenkins, sa le proceseze iar mai apoi sa le trimita serverului de SonarQube pentru a proni analiza de cod. Am optat pentru a salva datele in format xml in loc de o baza de date, pentru a fi mai usor de accesat si modificat. Putem zice ca acest mod de a salva datele intra tot la cerinte nonfunctionale, dar modificare modului in care se salveaza datele va modifica in cele din urma si putin din arhitectura.

La cerinte nonfunctionale intra interfata din Jenkins, aceasta poate fi modificata fara a afeca arhitectura sistemului, mai exact ordinea in care apar butoanele sau denumirile lor. Adaugarea de butoane noi sau campuri text noi, duc la o modificare al arhitecturii.

**Descompunearea in componente**

1. Serverul Jenkins

Serverul Jenkins este practic interfata dintre utilizator si aplicatie, mai exact dintre datele de intrare si modu in care sunt prelucrate. Interfata este usor de inteles si utilizat. Exista campuri de introducere de proiecte, modificare a celor existate sau stergerea lor. In plus tot de aici se alege care proiecte vor fi rulate de SonarQube

2.Serverul SonarQube

SonarQube este o platforma gratuita folosita pentru inspectia calitatii codului. Serverul primeste datele de intrare dintr-un tool c#, care mai departe primeste comanda ca, care proiect sa fie rulat din interfata Jenkins.

3. Tool C# pentru procesarea datelor de intrare

Dupa ce utilizatorul a ales comanda din interfata Jenkins(adaugare, actualizare si stergere), datele sunt trimise la un tool, care le prelucreaza, verificand ca datele sa fie valide si le salveaza intr-un fisier XML.

4. Tool C# pentru pornirea serverului SonarQube

Cand utilizatorul alege un proiect pentru a fi rulat, acest tool citeste datele care sunt in XML, verifica existenta proiectului, apoi scrie in Command Prompt comenzile necesare prin care ruleaza SonarQube.

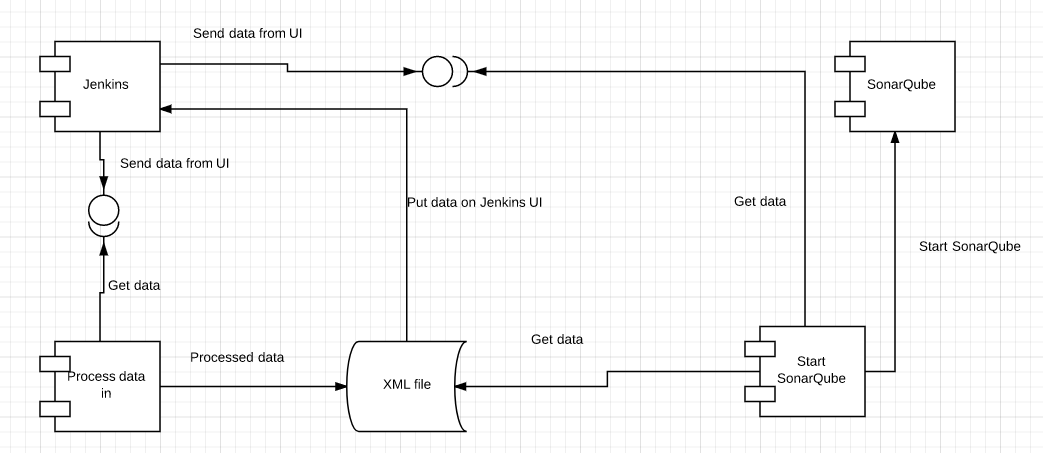
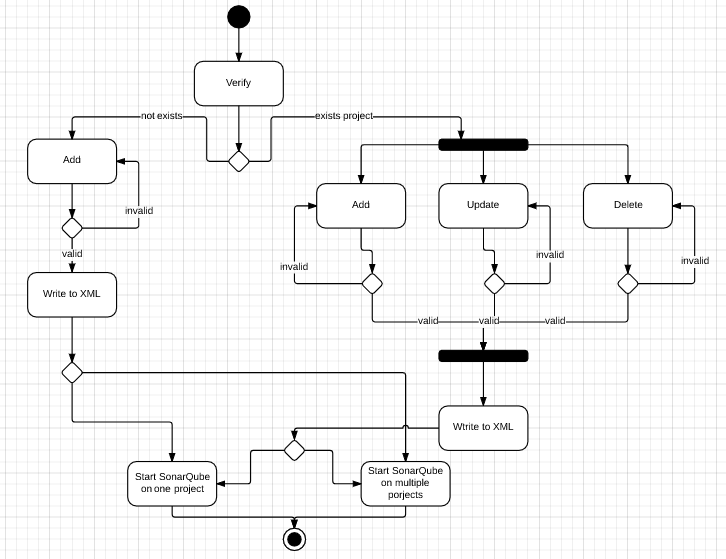


Diagrama de componente

Aici in diagrama de compontete se arata structura dintre componentele specificate anterior. Se observa faptul cum comunica componentele intre ele, de la serverul Jenkins care este practic interfata cu utilizatorul pana la serverul de SonarQube care este finalizarea proiectului.

Diagrama de activitati

In diagrama de activitati se prezinta algoritmul pe baza caruia functioneaza sistemul. In prima faza se verifica daca exista sau nu proiecte, deoarece si interfata din Jenkins se creeaza dinamic, asta insemnand ca dupa ce se adauga, modifica sau sterge proiect, pe interfata apar automat modificarile facute, iar pentru a nu cauza probleme, primul proiect adaugat are un algoritm diferit de scriere in fisierul XML. Daca avem cel putin un proiect existent putem mai departe face operatiile de adaugare, modificare si stergere. In final utilizatorul alege modul in care doreste sa opereze cu SonarQube, scanare pe un singur proiect sau pe proiecte multiple.

**Indicatori de calitate**

1). Performanta

Din punct de vedere al performantei, timpul de adaugare, modificare respectiv stergere, este strict limitat de serverul Jenkins, astfel aplicatiile c# nu contribuie la o performanta scazuta. Pe de alta parte, scanarea de proiecte poate dura mai mult timp, in functie de cat de mari sunt proiectele si pe cate proiecte se face scanarea. Sitemul a fost conceput pentru a rula automat, sau in afara factorului uman.

2). Disponibilitate

Sistemul a fost conceput pentru a rula 24 din 24 fara a fi probleme si fara a avea erori. In prima faza cand se fac operatii de adaugare, editare sau stergere, daca utilizatorul face ceva eronat, automat aplicatia returneaza un mesaj de eroare, iar daca nu au avut erori, din nou aplicatia transmite utilizatorul acest lucru. La fel si pentru partea de analiza de cod, algoritmul din spatele sistemului impiedica blocarea aplicatiei.

3). Securitate

Pentru securitatea sistemului in mare parte se ocupa serverul Jenkins. Aici putem limita accesul sau chiar opri accesul utilizatorilor neautoritazi. Singurul lucru care poate fi considerat o gaura in securitate, esta daca un utilizator are acces la masina pe care sunt puse programele c# si stie unde sa gaseasca fisierul XML de configurare al aplicatie. Aplicatia a fost conceputa pentru a rula de pe o masina la care nu are lumea acces dintr-o firma, doar system adiministrator. Dar totusi, daca cineva neautorizat are acces la masina, nu poate sa exploateze aplicatia fara a sti unde se gaseste fisierul XML in care sunt salvate datele proiectelor.

**Stiluri arhitecturale**

1). Database centric system

Cu toate ca nu avem stric o baza de date sql, am folosit totusi acest stil arhitectural, datele fiind salvate intr-un fisier XML. Faptul ca am folosit un fisier XML pe post de baza de date, ofera o viteza mai mare cu care sunt scrise si citite datele. Acest stil arhitectural s-a potrivit cel mai bine pentru modul in care functioneaza sistemul, toate componentele au nevoie de datele care sunt scrie in fisierul XML.

2). Client server

A fost nesara implementarea de arhitectura client server pentru a putea rula aplicatia de la distanta. Clientii au acces la aplicatie prin serverul Jenkins, prin care pot aduga, modifica sau sterge proiecte noi. Dupa ce are loc analiza de cod, clientii se conecteaza la serverul SonarQube unde vad rezultatele inspectie de cod. Conexiunea se face cu usurinta, deoarece din serverul Jenkins se poate selecta care este adresa pe care ruleaza serverul, iar clientii au nevoie doar de aceea adresa pentru a putea folosi aplicatia.

**Validarea arhitecturii**

Arhitectura este validata in modul in care sunt prelucrate datele de intrare in cele doua tool-uri c#. Se fac verificari in orice pas pe care il face utilizatorul, pentru a vedea daca datale sunt valide, unele dintre cazuri ar fi, date null, nume proiect existent, calea spre proiect nu exista, sau fisierul XML contine date invalide. Dupa fiecare pas pe care il face utilizatorul i se transmite daca pasul facut a avut succes, altfel primeste un mesaj de eroare corespunzator, fara a se bloca aplicatia.