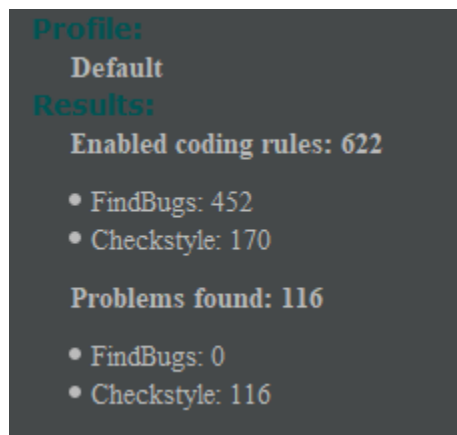


Verificare si validare software

Analiza Statica

Aceasta este o analiza a codului fara a fi rulat, verificand ca s-au respectat normele de programare cat si corectitudinea programului.













Pentru aceasta am folosit pluginurile QAPlug CheckStyle si QAPlug FindBugs din IDE-ul IntelliJ IDEA, acestea fiind rezultatele:



Dupa cum se poate observa, programul nu are nici un bug in structura programului, insa sunt 116 cazuri de imbunatatiri a performantei sau respectarii normelor de programare.

Cateva exemple si motivul pentru care au fost ignorate:




```

>  Avoid Star Import Count: 1
>  Cyclomatic Complexity Count: 1
>  Line Length Count: 10
>  Missing javadoc method Count: 17
>  Missing javadoc type Count: 1
>  Multiple String Literals Count: 9
>  Nested If Depth Count: 2
>  One top level class Count: 1
>  Outer type number Count: 1
>  Package Declaration Count: 1
>  Package javadoc Count: 1
>  Single space separator Count: 1

```

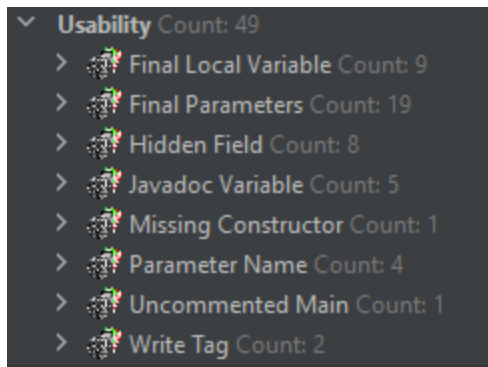
- Avoid Star Import: Este o buna practica sa nu importam mai multe clase decat este necesar, insa in acest proiect am considerat ca este necesar pentru eventuala utilizare a altor functii.
- Cyclomatic Complexity: Nu este o problema majora, mai ales fiindca aceasta se depaseste doar cu 1.
- Line Length: Nu este o problema, liniile nu vor fi trunchiate de compilator.
- Missing javadocs: Nu este nevoie de documentatie in fisierul sursa, aceasta poate fi ulterior scrisa intr-un document ce va descrie programul in mod etnec.
- Multiple String Literals: Nu este o problema prezenta lor in cod.
- Package Declaration: Nu este nevoie de package neaparat intr-un program cu scop si dimensiuni restranse.
- One top level class: Nu este o problema in programul acesta cu scop limitat, ar fi irrelevant daca Main ar fi propria clasa .java sau integrate in WebServer.

```

v Reliability Count: 19
>  Design For Extension Count: 16
>  Equals Avoid Null Count: 2
>  Magic Number Count: 1

```

- Design for Extension: Nu este o problema majora, nu va fi nevoie de documentatie Javadoc pentru ca in scopul proiectului, aceste clase nu vor fi mai departe extinse.
- Equals Avoid Null: Este o buna practica sa punem string-urile pe partea stanga, insa in acest proiect ele vor fi intotdeauna setate si nu exista posibilitatea de a fi "null".
- Magic Number: Este portul recomandat in laboratorul de VVS, prezenta lui nu impacteaza codul.



- Final Local Variable: Acestea sunt prin default declarate final, nu este nevoie de declarare de programator.
- Final Parameters: Nu impacteaza cu nimic codul daca sunt lasati fara final.
- Hidden Field: Nu este o problema, apelul constructorului si a functiilor de get si set sunt corecte.
- Javadoc Variable: Nu este nevoie de documentatie in fisierul sursa, aceasta poate fi adaugata ulterior.
- Missing Constructor: Nu este o problema pentru clasa Main.
- Parameter Name: In normele de programare Java, este recomandat ca variabilele sa nu contina caractere cum ar fi _ ci sa fie cuvinte combinate dupa regex-ul prezentat. Nu este o problema ca aceste 4 variabile nu respecta norma.

Analiza dinamica

Aceasta este o analiza a codului in timp ce acesta ruleaza, verificand in timp real inputurile si outputurile pentru a se asigura ca totul se executa correct.

Pentru aceasta am folosit aplicatia VisualVM pentru a monitoriza performanta WebServerului din mai multe aspecte, odata ce a fost pornit:

