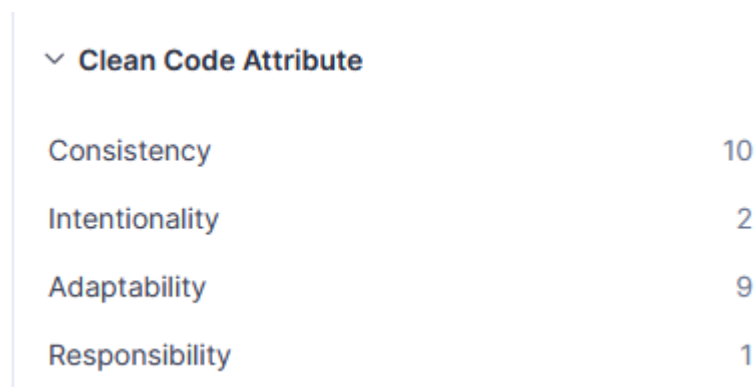


Kwaliteitsproblemen

Het programma SonarQube heeft de volgende problemen gevonden. Dit valt onder de categorie Clean Code en Softwarekwaliteit. In deze pdf worden de gevonden bugs onderverdeeld in deze 2 categorieën en verder toegelicht. Oplossingen worden hierbij ook gegeven.

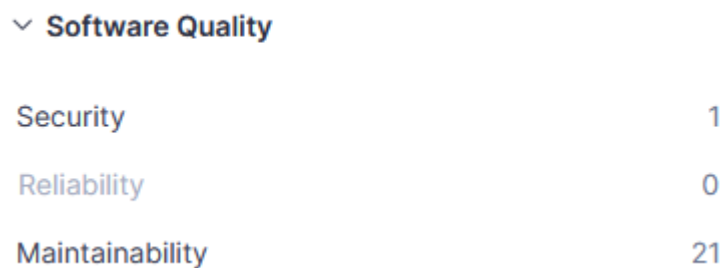
Figuur 1

A screenshot of a SonarQube interface showing a table of Clean Code attributes. The table has a header row with a dropdown arrow and the text 'Clean Code Attribute'. Below it are four rows: 'Consistency' with a value of 10, 'Intentionality' with a value of 2, 'Adaptability' with a value of 9, and 'Responsibility' with a value of 1.

▼ Clean Code Attribute	
Consistency	10
Intentionality	2
Adaptability	9
Responsibility	1

Clean code

Figuur 2

A screenshot of a SonarQube interface showing a table of Software Quality attributes. The table has a header row with a dropdown arrow and the text 'Software Quality'. Below it are three rows: 'Security' with a value of 1, 'Reliability' with a value of 0, and 'Maintainability' with a value of 21.

▼ Software Quality	
Security	1
Reliability	0
Maintainability	21

Software Quality

Clean code

Clean code richt zich op het schrijven van leesbare, begrijpelijke en onderhoudbare code. De gevonden problemen hebben volgens SonarQube voornamelijk te maken dus met consistency, intentionality, adaptability en responsibility:

Consistency:

Variabele- en functienamen:

Weinig consistentie in het benoemen van variabelen en functies leidt tot onduidelijkheid.

Coding style:

Een inconsistent gebruik van inspringing en spaties kan het moeilijk maken om de code te lezen.

Oplossing:

Het is belangrijk om een algemene en betekenisvolle naamgeving te gebruiken. Het handhaven van een consistente coding style vergroot de leesbaarheid.

Intentionality:

Commentaar:

Onvoldoende of slecht commentaar kan de onduidelijkheid van de code vergroten.

Magische getallen en strings:

Gebruik van hardcoded getallen en strings zonder duidelijke uitleg vergroot ook onduidelijkheid van de gegeven code.

Oplossing:

Goede documentatie en commentaren zijn essentieel om anderen en jezelf te helpen begrijpen waarom bepaalde beslissingen zijn genomen. Het is beter om constanten of variabelen duidelijker te benoemen.

Adaptability:

Hard-coded waarden:

Het rechtstreeks invoeren van waarden in de code maakt het moeilijk om wijzigingen aan te brengen.

Overmatige afhankelijkheden:

Te veel afhankelijkheden tussen verschillende delen van de code maken het moeilijker om aanpassingen te doen.

Oplossing:

Het is beter om configuraties en instellingen apart te houden om de aanpasbaarheid te verbeteren. Het gebruik van interfaces en het minimaliseren van koppelingen kan helpen de aanpasbaarheid te vergroten.

Responsability:

Single Responsibility Principle (SRP): Functies en klassen moeten een duidelijk afgebakende verantwoordelijkheid hebben.

Ongepaste koppeling: Modules die te erg van elkaar afhankelijk zijn, kunnen problemen veroorzaken bij het begrijpen en wijzigen van de code.

Oplossing:

Te veel verantwoordelijkheden in één entiteit maken de code complex en moeilijk te begrijpen.

Het toepassen van het Dependency Inversion Principle (DIP) kan de verantwoordelijkheid beter verdelen.

Software quality

Security:

Onvoldoende authenticatie en autorisatie:

Gebrekkige implementatie van gebruikers authenticatie en autorisatie kan toegang tot gegevens mogelijk maken, waardoor beveiligingsrisico's ontstaan.

Onvoldoende gegevensencryptie:

Het ontbreken van encryptie in bijvoorbeeld database.php kan leiden tot datalekken en andere security kwetsbaarheden.

Onvoldoende logging en monitoring:

Een gebrek aan logging en monitoring maakt het moeilijk om beveiliging te waarborgen

Oplossing:

Het is belangrijk om sterke encryptie protocollen te gebruiken. Regelmatige updates en monitoring van bibliotheken zijn ook belangrijk. Goede logging is belangrijk voor het opsporen van potentiële bedreigingen.

Maintainability:

Slechte codekwaliteit:

Onduidelijke en rommelige code maakt het moeilijk voor ontwikkelaars om wijzigingen aan te brengen en bugs op te lossen.

Gebrek aan modulaire structuur:

Een gebrek aan scheiding maakt het lastig om specifieke functies te isoleren en te onderhouden.

Onvoldoende testdekking:

Het ontbreken van uitgebreide tests kan leiden tot onvoorziene bugs en introduceert risico's bij het aanbrengen van wijzigingen.

Oplossing:

Het naleven van best practices voor schone code bevordert de onderhoudbaarheid. Het toepassen van modulaire ontwerp vergemakkelijkt het onderhoudsproces. Een duidelijke set van tests maakt het onderhoud en de verdere ontwikkelingen van de software een stuk makkelijker.

Gevonden bugs door SonarQube

Database.php:

Consistency issue

[Add a new line at the end of this file.](#)

Consistency issue

[Rename function "get_state" to match the regular expression `^\[a-z\]\[a-zA-Z0-9\]*\$`.](#)

Responsibility issue

[Add password protection to this database.](#)

Consistency issue

[Remove this closing tag `"?>"`.](#)

move.php

Consistency issue

[Add a new line at the end of this file.](#)

Adaptability issue

[Replace "include_once" with namespace import mechanism through the "use" keyword.](#)

Intentionality issue

[Add curly braces around the nested statement\(s\).](#)

Adaptability issue

[Replace "include" with namespace import mechanism through the "use" keyword.](#)

Consistency issue

[Replace "include" with "include_once".](#)

Consistency issue

[Split this 137 characters long line \(which is greater than 120 authorized\)](#)

Consistency issue

[Remove this closing tag `"?>"`](#)

pass.php

Consistency issue

[Add a new line at the end of this file.](#)

Adaptability issue

[Replace "include" with namespace import mechanism through the "use" keyword.](#)

Consistency issue

[Replace "include" with "include_once".](#)

Consistency issue

[Split this 135 characters long line \(which is greater than 120 authorized\).](#)

Consistency issue

[Remove this closing tag ">".](#)

play.php

Consistency issue

[Add a new line at the end of this file.](#)

Adaptability issue

[Replace "include_once" with namespace import mechanism through the "use" keyword.](#)

Intentionality issue

[Add curly braces around the nested statement\(s\).](#)

Adaptability issue

[Replace "include" with namespace import mechanism through the "use" keyword.](#)

Consistency issue

[Replace "include" with "include_once".](#)

Consistency issue

[Split this 133 characters long line \(which is greater than 120 authorized\).](#)

Adaptability issue

[Replace "include_once" with namespace import mechanism through the "use" keyword.](#)

Consistency issue

[Remove this closing tag ">".](#)

restart.php

Consistency issue

[Add a new line at the end of this file.](#)

Consistency issue

[Split this 135 characters long line \(which is greater than 120 authorized\).](#)

Adaptability issue

[Replace "include_once" with namespace import mechanism through the "use" keyword.](#)

Consistency issue

[Remove this closing tag ">".](#)

undo.php

Adaptability issue

[Replace "include" with namespace import mechanism through the "use" keyword.](#)

Consistency issue

[Replace "include" with "include_once".](#)

Consistency issue

[Remove this closing tag ">".](#)

util.php

Consistency issue

[Remove this closing tag ">".](#)

Intentionality issue

[This function has 4 returns, which is more than the 3 allowed.](#)

Intentionality issue

[Add curly braces around the nested statement\(s\).](#)

Consistency issue

[Remove this closing tag ">".](#)