

# ANALYSE SONARQUBE

## EQUIPE 1



### Contexte

Suite à une étude de santé publique sur les étudiants en cursus informatique, le ministère de la santé nous a demandé de développer une solution permettant la réalisation d'un parcours virtuel via la pratique d'une activité sportive en réel. Elle doit permettre de créer des parcours virtuels via un site web et de se déplacer via une application mobile. Durant la mise en place de l'application, il nous a paru judicieux de déployer un outil permettant d'analyser la qualité du code afin de le rendre plus robuste, moins de redondant et facilement maintenable.

Le but du présent document est de présenter l'outil utilisé et la démarche que nous avons mise en place pour améliorer la qualité du code source de notre application.

Composition de l'équipe :

- Alabic Nemanja
- Derausseaux-Lebert Nathanael
- Metzger Nicolas
- Sacher Marilyn
- Steiner Kélio

### SonarQube

#### Présentation de l'outil

SonarQube est un logiciel permettant de mesurer la qualité d'un code source en continu. Il permet d'avoir une évolution dans le temps et des vues différentielles. Il supporte plus de 25 langages et génère des reportings sur :

- Le respect de règle de programmation
- La présence de bugs ou de faille de sécurité
- La duplication de code
- La complexité du code
- La couverture de code

#### Comprendre une analyse SonarQube

L'analyse SonarQube se base sur une série de règles de codage et d'indicateurs défini par défaut dans l'application pour les langages pris en charge. Ces réglés et indicateurs peuvent être complétés ou modifiés par les utilisateurs.

Lors de l'analyse SonarQube va soulever des "problèmes" chaque fois qu'il rencontre un élément qui va à l'encontre des règles définies. Il y a 3 types de problèmes possibles :

1. **Bug** : une erreur dans le code qui doit être fixé immédiatement
2. **Vulnerability** : un endroit de votre code qui constitue une faille de sécurité
3. **Code Smell** : un point de votre code qui est confus et peut s'avérer difficile à maintenir

Pour chacun de ces problèmes, vous avez 5 niveaux de sévérité :

1. **Bloquant** : bug ayant un fort impact sur votre code en production qui doit être fixé immédiatement.
2. **Critique** : bug avec un faible impact qui doit être examiné.
3. **Majeur** : problème de qualité de code qui impacte la maintenabilité du code
4. **Mineur** : problème de qualité de code avec un impact faible la maintenabilité du code
5. **Info** : juste informatif

# ANALYSE SONARQUBE

## EQUIPE 1



Par ailleurs, à chaque analyse, SonarQube compare à la précédente analyse. Il affiche une analyse globale et une sur les nouveaux problèmes rencontrés. L'analyse va permet d'obtenir un rapport qui affichera entre autres les informations suivantes :

- Nombre de bugs
- Nombre de code smell
- Nombre de vulnérabilités et **Security Hotspots** (une faille de sécurité devant être vérifiée par le développeur doit vérifier avant d'appliquer ou non un correctif).
- Complexité : simple, à savoir le nombre de subdivisions du code (ex if else, switch, ....) et cognitive, à savoir la difficulté à suivre le « flux » du code, c-à-d le lire et le comprendre.
- Duplication : nombre de lignes dupliquées, de fichiers contenant des duplications, ....
- Maintenabilité : temps nécessaire pour corriger, par rapport à celui déjà dédié au développement
  - $\leq 5\%$ , la note est A
  - Entre 6 et 10 % la note est un B
  - Entre 11 et 20 % la note est un C
  - Entre 21 et 50 % la note est un D
  - Tout ce qui dépasse 50 % est un E
- Dette technique : temps nécessaire pour corriger les code smell en minutes (1 jours = 8h)
- Le statut "**Quality Gate**" : État du projet par rapport au quality gate (Valeurs possibles ERROR et OK).  
Le "**Quality Gate**" : liste de conditions appliquées sur des mesures que l'analyse du code doit respecter, pour exemple

Conditions ⓘ

Conditions on New Code

Metric	Operator	Value
Coverage	is less than	80.0%
Duplicated Lines (%)	is greater than	3.0%
Maintainability Rating	is worse than	A
Reliability Rating	is worse than	A
Security Hotspots Reviewed	is less than	100%
Security Rating	is worse than	A

## Organisation du projet et technologies

Pour la mise en place de notre application, nous avons réalisé 3 parties distinctes :

- Une API qui gérera l'accès et les requêtes à la base de données
- Un site web qui permettra la gestion des challenges par l'administrateur
- Une application mobile qui sera utilisée par le joueur pour se déplacer sur un parcours

Le projet est divisé comme suit :

- Un répertoire backend dans lequel se trouve un répertoire server où se trouve l'ensemble des répertoires, librairies et fichiers nécessaires
- Un répertoire frontend avec deux sous répertoires :
  - Web avec l'ensemble des répertoires, librairies et fichiers nécessaires au site web
  - Mobile avec l'ensemble des répertoires, librairies et fichiers nécessaires à l'application mobile

En ce qui concerne les technologies employées :

- L'API a été développée en python avec le framework de développement web, Pyramid
- Le site web et l'application sont en javascript :
  - Le site web, il est développé avec la librairie reactJS et pour la partie graphique, matériel ui
  - L'application mobile est développée avec le framework ReactNative

# ANALYSE SONARQUBE

## EQUIPE 1



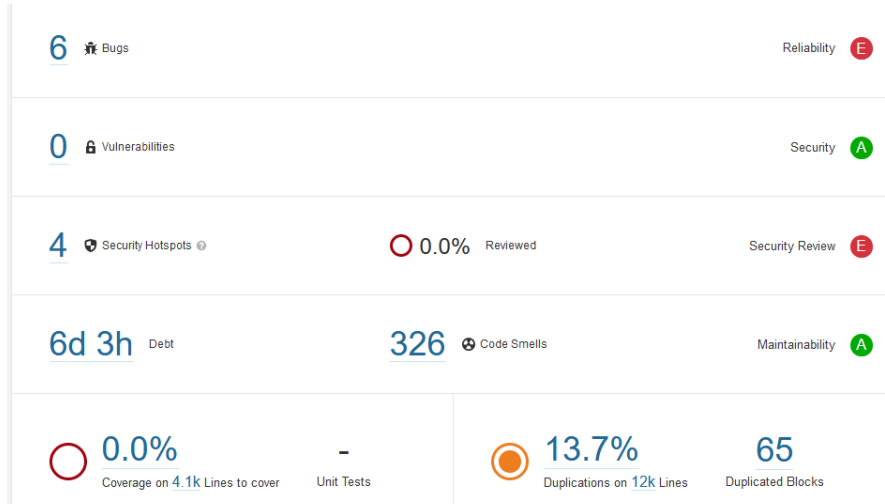
### Analyse SonarQube du projet

Nous avons fait une première analyse, qui a mis en évidence sur l'ensemble de l'application

- 6 bugs
- 4 Sécurité HotSpots
- 326 code smell
- 0% coverage
- 13,7 % de duplication

Passed

All conditions passed.



Nous avons commencé par corriger les 6 bugs en appliquant les correctifs nécessaires :

Filters

Clear All Filters

Type: BUG

Clear

Bug 6

Vulnerability 0

Code Smell 326

Ctrl + click to add to selection

Severity

Blocker 1

Critical 1

Major 4

Minor 0

Info 0

Scope

dev/.../marshmallow\_schema/SegmentSchema.py

Delete this unreachable code or refactor the code to make it reachable. Why is this an issue? 2 min effort

dev/.../server/loftes/resources/CheckChallengeResources.py

Remove or refactor this statement; it has no side effects. Why is this an issue? 10min effort

dev/.../Components/Main/ChallengeCard/ChallengeMap.js

Add a "return" statement to this callback. Why is this an issue? 5min effort

Consider using "forEach" instead of "map" as its return value. Why is this an issue? 5min effort

Pour exemple :

```
else:
    1 raise ValueError("The segment's coordinates must be of the type array.")

    data["coordinates"] = json.dumps(data["coordinates"])

Delete this unreachable code or refactor the code to make it reachable. Why is this an issue? 2 min effort
```

Correction

```
103
104
105
106
107
108
else:
    raise ValueError("The segment's coordinates must be of the type array.")

return data

def check_json(self, data, segment, **kwargs):
```

Nous avons ensuite examiné les « Sécurité Hotspots », mais il s'agissait de code de tests amené à disparaître

4 Security Hotspots to review

Review priority: MEDIUM

Weak Cryptography 4

Make sure that using this pseudorandom number generator is safe here.

TO REVIEW

Make sure that using this pseudorandom number generator is safe here.

TO REVIEW

Make sure that using this pseudorandom number generator is safe here.

TO REVIEW

Make sure that using this pseudorandom number generator is safe here.

TO REVIEW

Make sure that using this pseudorandom number generator is safe here.

Add Comment

Open in IDE

Category Weak Cryptography

Review priority MEDIUM

Assignee nicolasmetzger

Status: To review

This Security Hotspot needs to be reviewed to assess whether the code poses a risk.

dev/.../web/lord-of-the-trips/src/konva/konvaExample.js

```
2 import { Stage, Layer, Star, Text } from 'react-konva';
3
4 function generateShapes() {
5   return [...Array(10)].map((_, i) => ({
6     id: i.toString(),
7     x: Math.random() * window.innerWidth,
8     y: Math.random() * window.innerHeight,
9     rotation: Math.random() * 180,
10    isDragging: false,
```

# ANALYSE SONARQUBE

## EQUIPE 1



Puis, les codes smells nous ont permis de mettre en place des optimisations de codes en identifiant :

- Des variables ou des noms de fonctions ne respectant pas les normes de nommage

```
✓1 dev/.../LOTT_Mobile/src/Components/Main/ChallengesList.js
Rename this file to "ChallengeList" Why is this an issue?
Code Smell Minor Open N nderousseaux 5min effort Comment

import * as React from 'react';
import { View, StyleSheet, ActivityIndicator } from 'react-native';
import { useEffect, useState } from 'react';

'react' import is duplicated. Why is this an issue?
Code Smell Minor Open Not assigned 1min effort Comment

import { ListItem } from 'react-native-elements';
import api from '../api/api';

export default function ChallengeList(props) {
```

Correction

```
ChallengesList.jsx 1.63 KB
1 import React from 'react';
2 import { FlatList, View } from "react-native";
3
4 import { ChallengesConsumerHook } from 'store/challenge';
5 import { getChallenges } from 'helpers/ChallengesHelper';
6 import ChallengeCard from 'components/challenges/challer';
7 import Button from 'components/basics/button/ButtonComp';
8 import styles from './ChallengesList.style';
9
10 export default function ChallengesList(props){
```

- Des textes en dur répétés dans le code qui pouvait être mutualisé, et pour lequel nous avons mis en place un fichier de constantes avec le contenu des messages

```
else:
    response = service_information.build_response(
        exception.HTTPNotFound(),
        None,
        "Requested resource 'Challenge' is not found.",
    )

Define a constant instead of duplicating this literal "Requested resource 'Challenge' is not found." 7 times. Why is this an issue?
Code Smell Critical Open sacher 14min effort Comment
```

- Du code en commentaires qui surchargés inutilement les fonctions que nous avons supprimé

```
19 nema... # from Loftes.marshmallow_schema.EventSchema import EventSchema

Remove this commented out code. Why is this an issue?
Code Smell Major Open A alabic 5min effort Comment
```

- Des imports inutiles de données ou des variables inutilisées qui ont été supprimés

```
import * as React from 'react';
import { View, StyleSheet, Image, Modal, Text, ActivityIndicator } from 'react-native';

Remove this unused import of 'Text'. Why is this an issue?
Code Smell Minor Open Not assigned 2min effort Comment
```

- Des portions de code non encore écrites via la détection du texte TODO/todo

```
// Calcul de la position de chaque obstacle passé
if (passedObstacles.length > 0) {
    passedObstacles.map(obstacle => {
        // TODO

Complete the task associated to this "TODO" comment. Why is this an issue?
Code Smell Info Open Not assigned Comment
```

Nous avons aussi été confronté à des problèmes dont la résolution n'était pas toujours aisée, voir pour lesquelles nous avons préféré ne pas intervenir car les modifications auraient pu entraîner des erreurs dans le fonctionnement de l'application en production, c'est le cas notamment :

- Des problèmes de complexité cognitive, dans ce cas les modifications peuvent générer des effets de bord non désirés ou provoquer un dysfonctionnement de la fonction impactée, et parfois le code ne paraît pas forcément complexe

# ANALYSE SONARQUBE

## EQUIPE 1



```
@challenge.get()
def get_challenges(request):

    service_informations = ServiceInformations()

    user = DBSession.query(User).filter(User.email == request.authenticated_userid).first()

    1 if user != None:

        challenges = []

        # solution pour ce sprint, pensez à changer
        2 if user.id == 1:
            challenges = DBSession.query(Challenge).all()

        3 if request.query_string != "":
            splitter = request.query_string.split("=")
            4 if len(splitter) == 2 5 and splitter[0] == "draft":
                6 if splitter[1] == "false":
                    challenges = ChallengeResources().find_all_published_challenges()
                    # if user is not superadmin and he wants to see all challenges
                7 elif splitter[1] == "true" 8 and user.id != 1:
                    return service_informations.build_response()
```

- get\_obstacle\_for\_validationDes blocks dupliqués qu'il est compliqué d'extraire pour mutualisation

```
    )

    except ValidationError as validation_error:
        response = service_informations.build_response(
            exception.HTTPBadRequest, None, str(validation_error)
        )

    except ValueError as value_error:
        response = service_informations.build_response(
            exception.HTTPBadRequest, None, str(value_error)
        )

    except PermissionError as pe:
        response = service_informations.build_response(exception.HTTPUnauthorized)

    except Exception as e:
        response = service_informations.build_response(exception.HTTPInternalServerError)
        logging.getLogger(__name__).warn("Returning: %s", str(e))
```

Tout en gérant les différents problèmes rencontrés, nous avons plus globalement modifier et refactorisé le code de façon à augmenter sa maintenabilité et sa clarté.

Suite aux modifications effectuées, nous avons relancé une 2<sup>nd</sup>e analyse qui a montré une amélioration sensible par rapport à la précédente :

- Plus de bugs et de Hotspots
- Environ 46% de code smell en moins
- 3 points de moins en lignes dupliquées, et une réduction du nombre de lignes globales de l'application
- Et presque 2 jours (16h) de dettes techniques en moins

★ Loftes_V2 <span>Passed</span>		Last analysis: 9 days ago		PRIVATE		
Bugs	Vulnerabilities	Hotspots Reviewed	Code Smells	Coverage	Duplications	Lines
0 <span>A</span>	0 <span>A</span>	— <span>A</span>	176 <span>A</span>	0.0% <span></span>	10.2% <span></span>	11k <span>M</span> Python, Jav...

Ancienne Dette technique

6d 3h Debt

Nouvelle Dette technique

4d 4h Debt

# ANALYSE SONARQUBE

## EQUIPE 1



### En conclusion

L'outil sonarQube a été très intéressant à utiliser. Pour nous, il nous a permis de découvrir des bugs qu'une simple lecture du code n'aurait pas suffi à détecter. Mais le plus grand avantage de cet outil, c'est qu'il permet d'imposer une régularité dans la syntaxe du code. En effet, dans un travail collaboratif, tout le monde n'applique pas les mêmes normes de nommage, et cela peut rendre difficile la relecture du code, et ainsi réduire sa maintenabilité. Avec sonarQube, le code de notre projet a pu avoir une véritable cohérence.

Précisément, chez nous, il nous a permis de nous alerter sur certains morceaux de code dupliqués dans notre projet.

Cependant, nous avons pu voir certaines limites à l'outil. En effet, l'analyse nous a fait remonter beaucoup de faux positifs, par exemple certaines variables qui étaient initialisées à 'null' étaient considérées comme problématique. Tous ces faux positifs auraient pu être réglés par un paramétrage en amont de sonarQube, afin de définir les normes de codage que l'on aurait décidé d'appliquer à notre code. Mais cela aurait nécessité un vrai travail, et nous n'avions pas le temps. Nous pensons qu'il est là le vrai défaut de sonarQube : en entreprise, cela peut nécessiter un énorme investissement en temps de travail pour poser toutes les règles de code. Ensuite, si l'outil intègre une application longtemps après le début de son développement, il est probable que l'on se retrouve avec une énorme dette technique.

A notre avis, et pour conclure, SonarQube est un outil très utile et puissant, qui permet de faciliter le travail collaboratif sur un projet et d'améliorer la maintenabilité de son code. Il a aussi comme avantage de pouvoir repérer des petits bugs. Il se doit d'être utilisé dans tous les projets ambitieux. Cependant, l'outil demande un véritable travail avant de pouvoir le faire fonctionner correctement, et celui-ci n'a peut-être pas sa place dans des projets plus petit, ou à la durée de vie plus courte.