# Apache Commons-email

Luigi Allocca, Simone Della Porta, Rocco Iuliano

**ACM Reference Format:**
Luigi Allocca, Simone Della Porta, Rocco Iuliano. 2023. Apache Commons-email. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

## 1 INTRODUCTION

The aim of this project is to analyze and improve an existing project by Apache foundation. We will improve the availability (i), reliability (ii), safety (iii), security (iv) and resilience (v) of the project in order to improve the dependability. We will focus on improve security by improving the coverage of the test cases and creating new ones, fix the bugs related to security and reduce security hotspot issues which refer to a specific area or component of a software system that has a higher risk of security vulnerabilities or breaches. These issues are often identified through a security analysis or review, and they can pose a significant threat to the overall security of the system. By removing these security hotspots, the overall security of the project will be improved, reducing the risk of security breaches and protecting sensitive user data. This can lead to increased trust from users and stakeholders, and can help to ensure that the project is compliant with relevant security regulations and standards. Overall the general goals are:

- fix as many project bugs as possible, prioritizing the crucial bugs;
- improve the coverage of project testing by developing new test cases and improving the existing ones;
- minimize the number of code smells;
- reduce security hotspot issues;
- verify the project performance.

## 2 PROJECT PRE REQUIREMENTS

We choose a project with the following characteristics in order to improve the dependability of the software with CI/CD paradigm using the tools introduced in the course:

- Git Actions to check code quality, coverage, Java CI and security.
- The project should use Maven to manage the project build, so it should have the pom.xml file.

## 3 CONTEXT OF THE PROJECT

Apache Commons-Email is an open source project and it is released in 01/08/2017 and it is available on GitHub to the following link: https://github.com/apache/commons-email. It aims to provide a API for sending email. Apache Commons-Email is built on top of the Java Mail API, which it aims to simplify. This project is composed by twentyfour contributors and has got 78 fork, 5 branches and 1132 commits of which the last commit was made in 18/03/2023 (id commit: f41dd76). Moreover, Apache Commons-Email has got three type of actions:

(1) **CodeQL** that allows us to verify the quality of the project;
(2) **Coverage** that allows us to verify the per cent of coverage of project test cases;
(3) **Java CI** that allows us to apply all the test cases to each commit that we make, so we are sure that the changes that we applied to the software doesn't introduce a bug.

Apache Commons-Email has got a user guide and a JavaDoc API documentation that are available to the following link:

- User guide: https://commons.apache.org/proper/commons-email/userguide.html
- JavaDoc: https://commons.apache.org/proper/commons-email/javadocs/api-release/index.html

Some popular Java-based applications that use Apache Commons Email are Apache Jenkins, Apache OFBiz, and Apache Syncope. Additionally, many Java-based web applications and enterprise systems use Apache Commons Email to handle email sending functionality.
To apply the changes to the project we must follow this rules:

(1) No tabs, instead use spaces for indentation.
(2) Respect the code style.
(3) Create minimal diffs - disable on save actions like reformat source code or organize imports. If you feel the source code should be reformatted create a separate PR for this change.
(4) Provide JUnit tests for your changes and make sure your changes don't break any existing tests by running mvn.

## 4 PRELIMINAR ANALISYS

After selecting the project Commons-email, we have created a fork of the repository, cloned the repository and built the project in order to run all test cases which result passed successfully. Then we performed a test push to verify the project actions and check the reults.

## 5 METHODOLOGICAL STEPS CONDUCTED TO ADDRESS THE GOALS

Then we have conducted a preliminar analisys of the project by using *sonarcloud* [1] and *codecov* [2] and we obtained this results:


*Conference'17, July 2017, Washington, DC, USA*

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

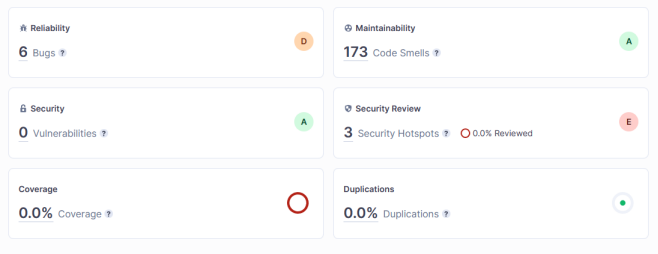


**Figure 1: Sonarcloud analysis**

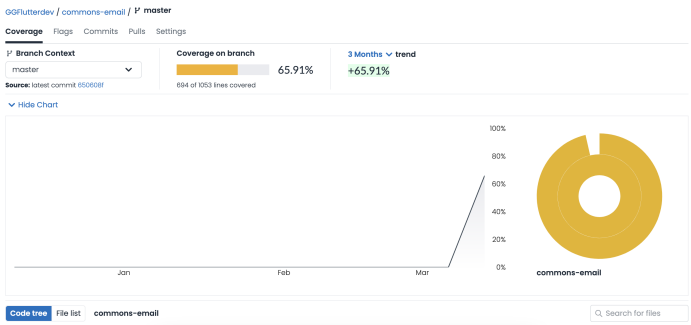


**Figure 2: Codecov analysis**

The result of the analysis are:
The coverage is 65,91%;
6 bugs, of which:

| Kind of problem | Number | Severity | Topic |
|---|---|---|---|
| Bug | 2 | Major | Synchronization |
| Bug | 2 | Major | NullPointerException |
| Bug | 2 | Major | bad formatted try/ catch |

**Table 1: Bugs report**

3 security hotspots of which:

| Kind of problem | Number | Severity | Topic |
|---|---|---|---|
| Security hotspot | 2 | - | DoS attack |
| Security hotspot | 1 | - | Weak cryptography |

**Table 2: Security hotspots report**

173 code smells, of witch:

| Kind of problem | Number | Severity | Topic |
|---|---|---|---|
| Code Smell | 11 | Bloker | Adding at least one assertion in test cases |
| Code Smell | 3 | Critical | Try-catch implementation |
| Code Smell | 5 | Critical | Cognitive Complexity allowed |
| Code Smell | 7 | Critical | Duplicate literals |
| Code Smell | 7 | Major | Commented out code |
| Code Smell | 42 | Major | Assert Equals and Assert Not Equals |
| Code Smell | 4 | Major | Expressions that are always evaluated to `true` |
| Code Smell | 4 | Major | Lambda exceptions |
| Code Smell | 7 | Major | Use of a generic exception instead of create and use a dedicated exception |
| Code Smell | 3 | Major | Testing |
| Code Smell | 3 | Major | Swap arguments |
| Code Smell | 2 | Major | Visibility of one constructor |
| Code Smell | 2 | Major | Try-catch implementation |
| Code Smell | 1 | Major | Number of assertions in a method |
| Code Smell | 1 | Major | Nested block of code |
| Code Smell | 1 | Major | Usage of a method instead of the usage of another method |
| Code Smell | 1 | Major | Usage of the utility of System class instead of the introdiction of a logger |
| Code Smell | 14 | Minor | Deprecated code |
| Code Smell | 2 | Minor | Chatset name argument |
| Code Smell | 2 | Minor | Hard-coded path-delimiter |
| Code Smell | 2 | Minor | Try-catch implementation |
| Code Smell | 1 | Minor | Return statement |
| Code Smell | 48 | Info | Deprecated code |

**Table 3: Code smells report**

Then for having a continuous analisys of the project we decided to integrate these tools in the our project in order to analyze the code after every push or pull request.

### 5.1 Microservices and Docker

A **microservices architecture** improves the dependability of our project by providing greater fault tollerance, scalability, flexibility and error isolation. In order to achieve this, we have used **Docker** which is a platform that lets us to create, deploy, and run applications using containers. A container is used to run images, which we have created with a Dockerfile in the root directory of our project.

The Dockerfile code:

```
#We specified the maven version and jdk version that we
     want in our image
FROM maven:3.8.4-jdk-11-slim AS build

#We created a directory
WORKDIR /app

#We copied the project in the image
COPY src ./src

#We copied the pom file in the image
COPY pom.xml ./

#We copied the conf directory in the image
COPY conf ./conf

RUN mvn clean package
```

In order to share between the members of our team the image of the project we have used **Docker Hub** which is a web based service that allowed us to store, manage, and share docker images.

Then we have integrated Docker with **CI/CD**, through GitHub actions. Infact when we make a commit and the project builds successfully, the image will be updated on Docker Hub, and we all have a working and updated image of the project with the command *docker pull –platform linux/x86_64 ggflutter/commons-email* on linux x86_64 or *docker pull ggflutter/commons-email* on other platforms.

## 6 RESULTS AND FINDINGS

## 7 CONCLUSIONS