Week 17: Metrics

blog.bookly.online

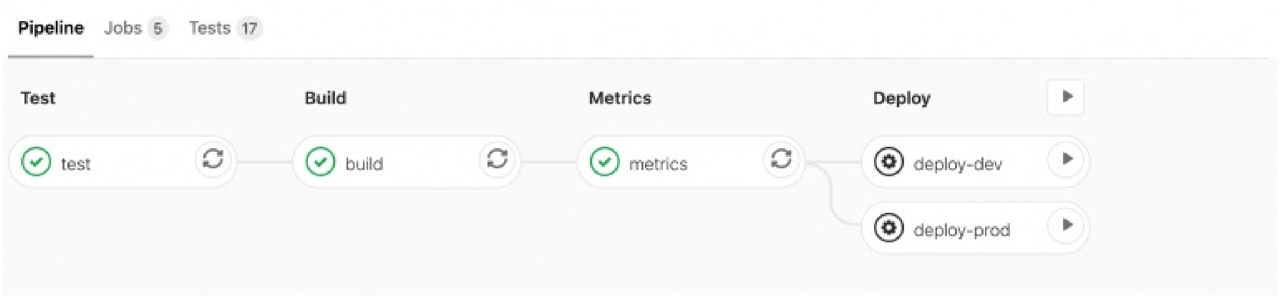
jeanne

May 28, 2020

Hey everyone,

For analyzing the metrics of our code we decided to use sonarqube. You can access the server here: sonarqube.bookly.online

We integrated the server into our deployment pipeline. After every build of any branch the metrics are updated. You can find the output of an example pipeline here.



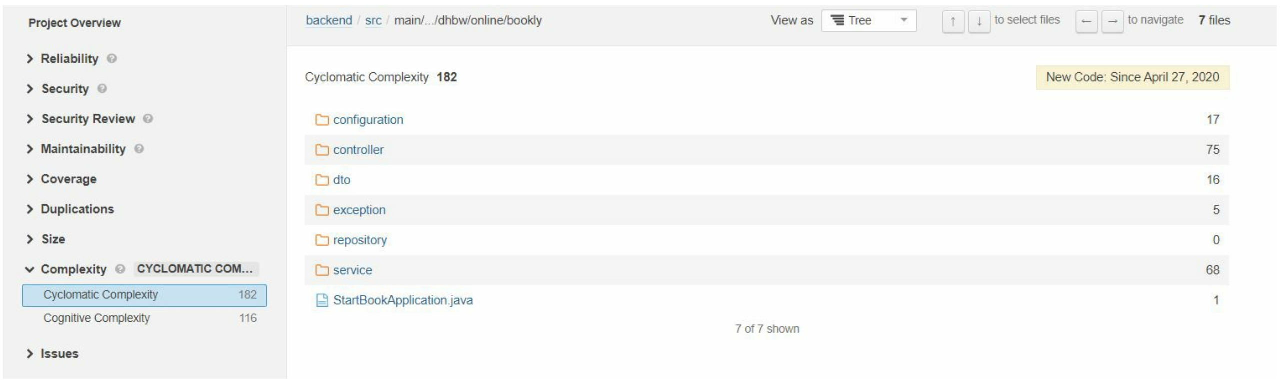
We decided to improve the metrics **Cyclometic Complexity** and reduce our **duplicated lines of code** (discussed and permitted by our Prof).

Cyclomatic Complexity (CYC)

In its simplest form, CYC is a count of the number of decisions in the source code. The higher the count, the more complex the code. So the less nested if-statements the better the rating for cyclometic complexity.

We tried to achieve this in the classes FriendshipBookController and EditPageController. We summed up some if-statements and deleted unnecessary requests. You can find the changes in this COMMIT and in you can see the improvements in the following pictures.

Before:



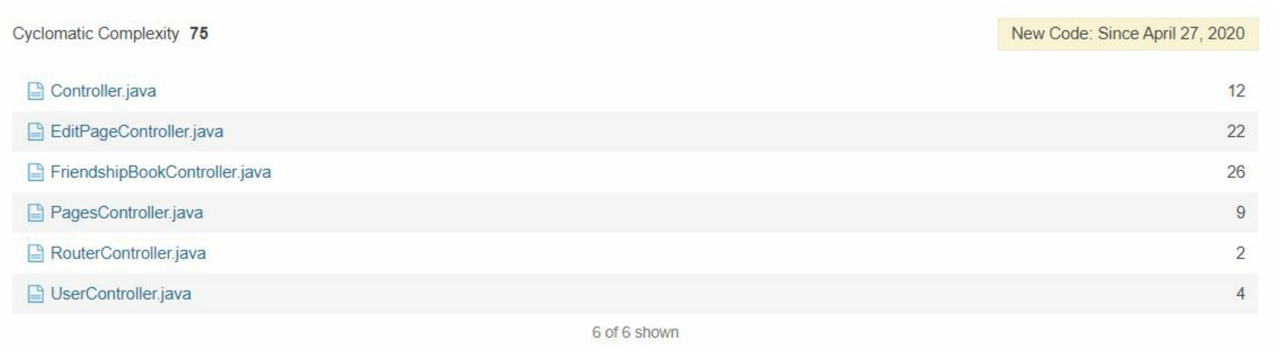
Cyclomatic Complexity	182
Cognitive Complexity	116

Cyclomatic Complexity 182 — New Code: Since April 27, 2020

configuration	17
controller	75
dto	16
exception	5
repository	0
service	68
StartBookApplication.java	1

7 of 7 shown

Before Refactoring: all Classes

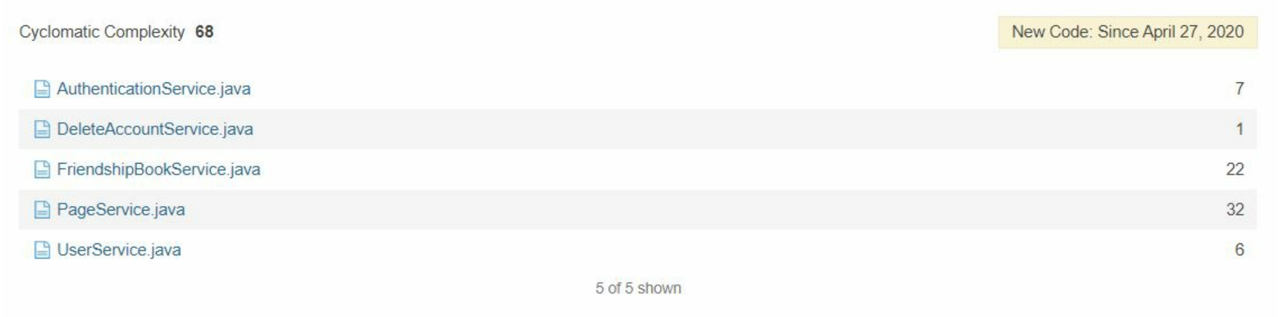


Cyclomatic Complexity 75 — New Code: Since April 27, 2020

Controller.java	12
EditPageController.java	22
FriendshipBookController.java	26
PagesController.java	9
RouterController.java	2
UserController.java	4

6 of 6 shown

Before Refactoring: Package Controller



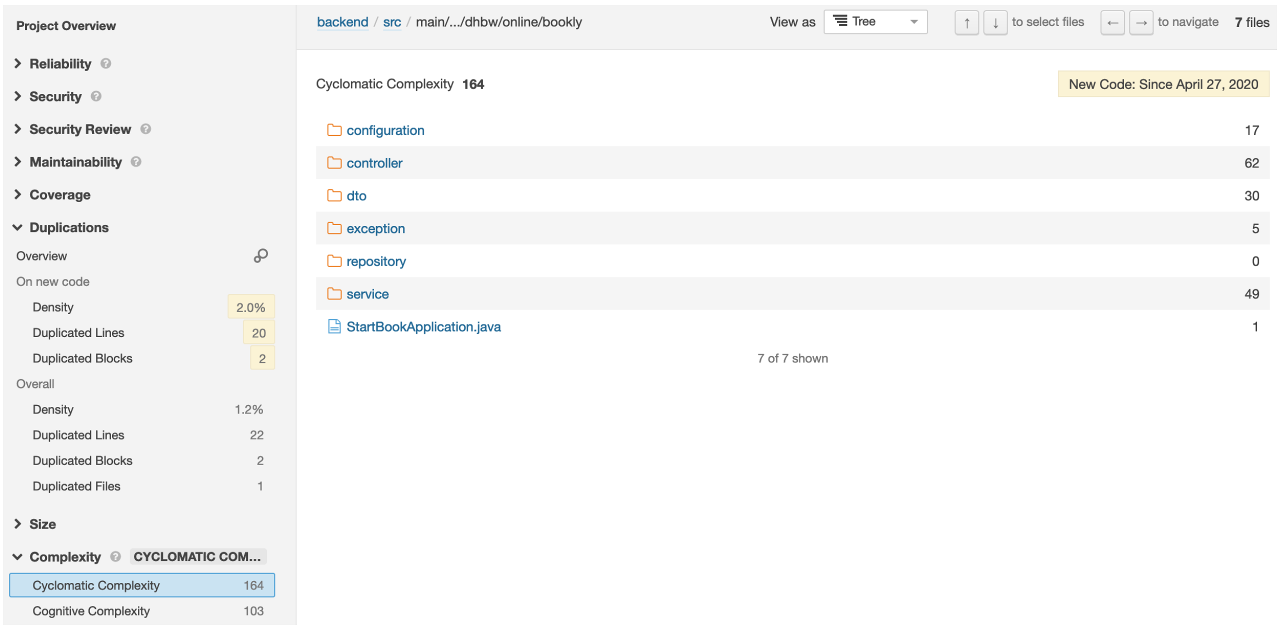
Cyclomatic Complexity 68 — New Code: Since April 27, 2020

AuthenticationService.java	7
DeleteAccountService.java	1
FriendshipBookService.java	22
PageService.java	32
UserService.java	6

5 of 5 shown

Before Refactoring: Package Service

After:



Project Overview

- Reliability
- Security
- Security Review
- Maintainability
- Coverage
- Duplications
  - Overview
  - On new code

Density	2.0%
Duplicated Lines	20
Duplicated Blocks	2

Overall

Density	1.2%
Duplicated Lines	22
Duplicated Blocks	2
Duplicated Files	1

- Size
- Complexity CYCLOMATIC COM...

Cyclomatic Complexity	164
Cognitive Complexity	103

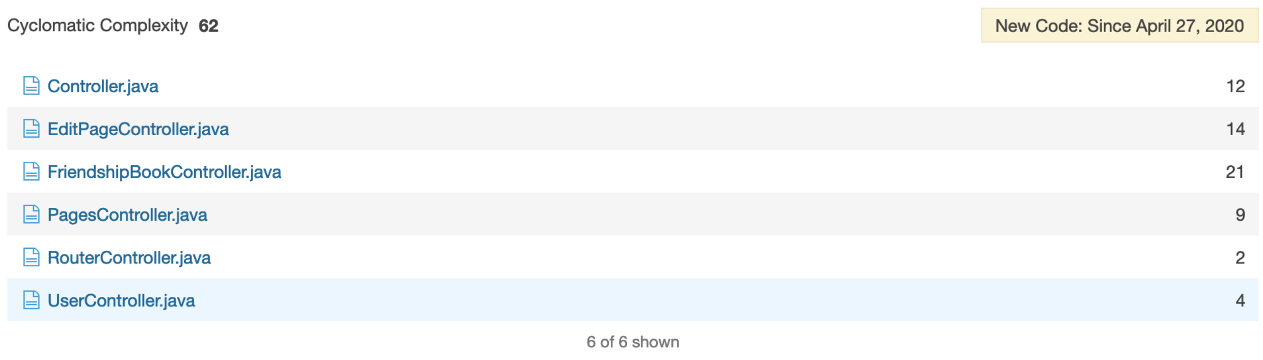
backend / src / main/.../dhbw/online/bookly — View as Tree — to select files — to navigate — 7 files

Cyclomatic Complexity 164 — New Code: Since April 27, 2020

configuration	17
controller	62
dto	30
exception	5
repository	0
service	49
StartBookApplication.java	1

7 of 7 shown

After Refactoring: All classes

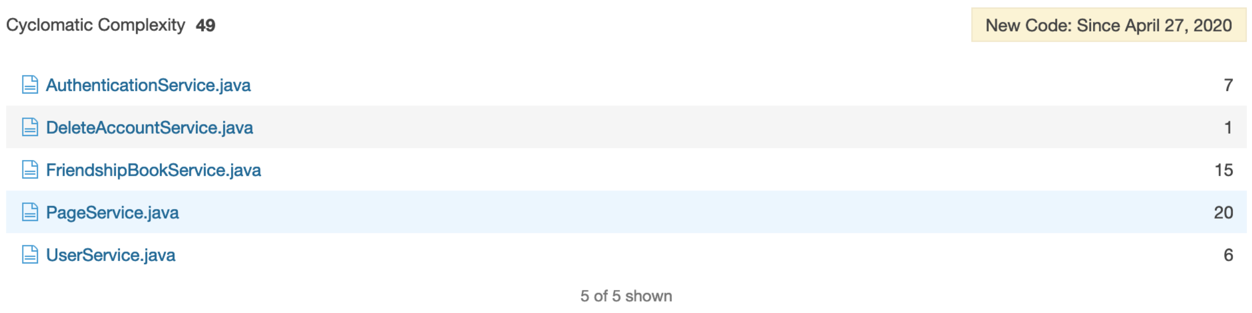


Cyclomatic Complexity 62 — New Code: Since April 27, 2020

Controller.java	12
EditPageController.java	14
FriendshipBookController.java	21
PagesController.java	9
RouterController.java	2
UserController.java	4

6 of 6 shown

After Refactoring: Package Controller



Cyclomatic Complexity 49 — New Code: Since April 27, 2020

AuthenticationService.java	7
DeleteAccountService.java	1
FriendshipBookService.java	15
PageService.java	20
UserService.java	6

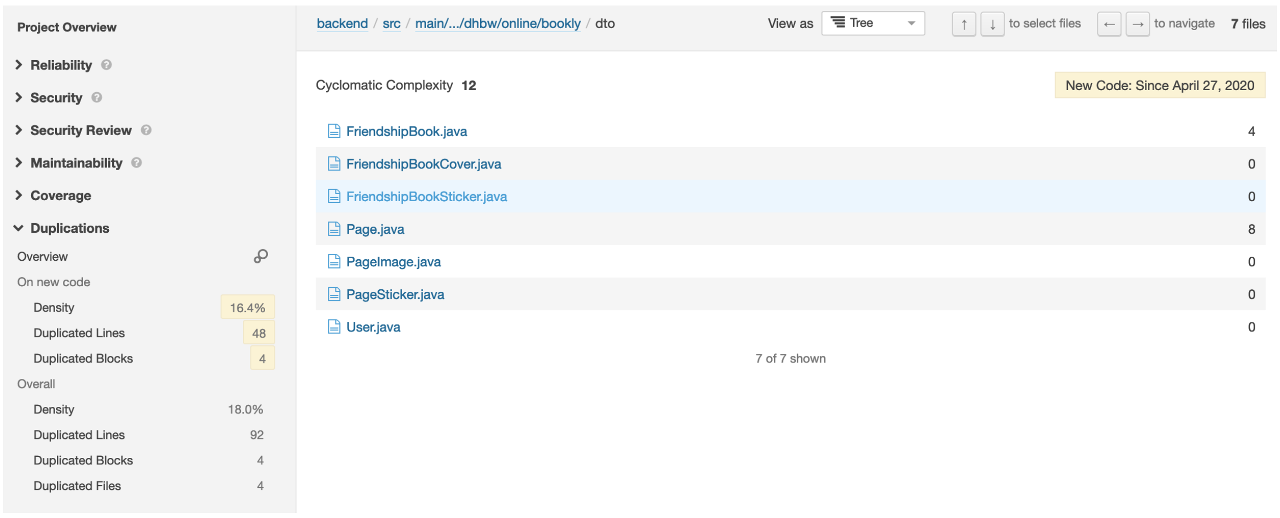
5 of 5 shown

After Refactoring: Package Service

For us, our code seems to be less readable now. We saved more lines of code but we think that our code seems to be more complex.

Duplicated Lines

As the name says, this metric is about the number of duplications in our project. At first we had a lot of duplication left. That's especially because of our multiple usage of image data in different objects but with the same attributes. After we refactored it and decided to use only one class type, our duplication reduced a lot. You can see the difference in this COMMIT and the improvement in the following pictures.


Project Overview

- Reliability
- Security
- Security Review
- Maintainability
- Coverage
- Duplications

Overview

On new code	
Density	16.4%
Duplicated Lines	48
Duplicated Blocks	4

Overall	
Density	18.0%
Duplicated Lines	92
Duplicated Blocks	4
Duplicated Files	4

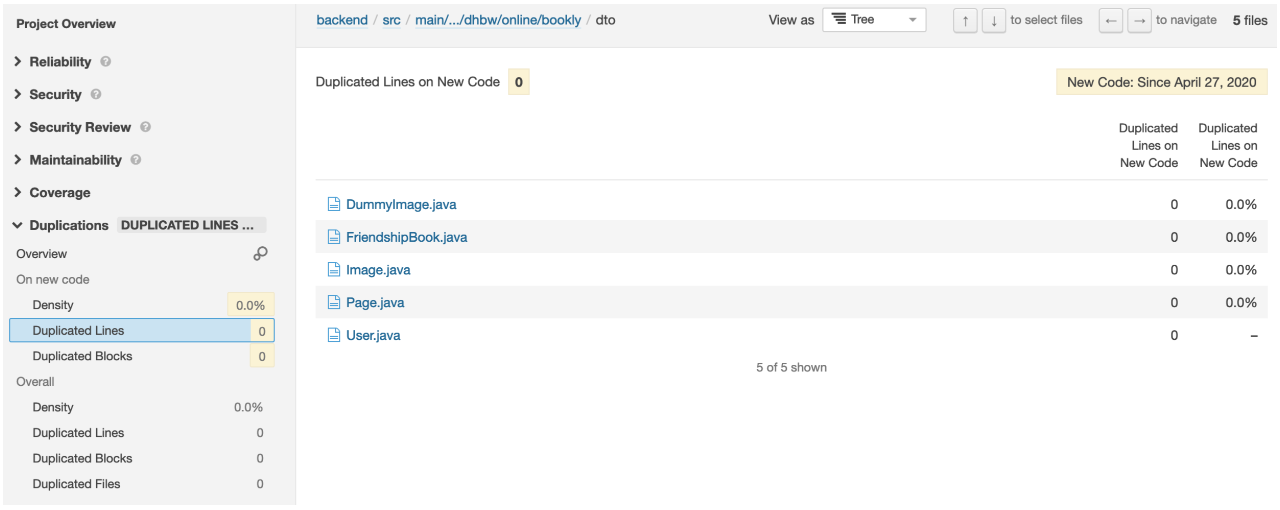
backend / src / main/.../dhbw/online/bookly / dto — View as Tree — ↑ ↓ to select files ← → to navigate — 7 files

Cyclomatic Complexity 12 — New Code: Since April 27, 2020

FriendshipBook.java	4
FriendshipBookCover.java	0
FriendshipBookSticker.java	0
Page.java	8
PageImage.java	0
PageSticker.java	0
User.java	0

7 of 7 shown

Before Refactoring: Package DTO


Project Overview

- Reliability
- Security
- Security Review
- Maintainability
- Coverage
- Duplications DUPLICATED LINES ...

Overview

On new code	
Density	0.0%
Duplicated Lines	0
Duplicated Blocks	0

Overall	
Density	0.0%
Duplicated Lines	0
Duplicated Blocks	0
Duplicated Files	0

backend / src / main/.../dhbw/online/bookly / dto — View as Tree — ↑ ↓ to select files ← → to navigate — 5 files

Duplicated Lines on New Code 0 — New Code: Since April 27, 2020

	Duplicated Lines on New Code	Duplicated Lines on New Code
DummyImage.java	0	0.0%
FriendshipBook.java	0	0.0%
Image.java	0	0.0%
Page.java	0	0.0%
User.java	0	–

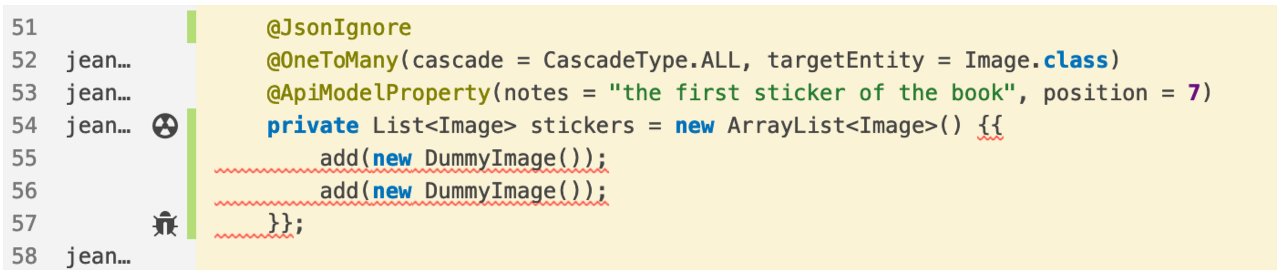
5 of 5 shown

After Refactoring: Package DTO

Functionality of Sonarqube

We've got another spot, where sonarqube detected duplicated code but we decided to keep it like this. In more detail we implemented two classes that generates test data for our application (See here). The first class generates data especially for our local environment as we are using the runtime database H2 that is deleting its data after a restart. The second class is especially for our test suite for the backend. Some code parts are overlapping. As those classes are used for different scenarios we are ignoring this note from sonarqube.

Even though sonarqube might be able to locate common bugs, we are not sure whether they are reliable for us or not. For example this one:



```
    @JsonIgnore
    @OneToMany(cascade = CascadeType.ALL, targetEntity = Image.class)
    @ApiModelProperty(notes = "the first sticker of the book", position = 7)
    private List<Image> stickers = new ArrayList<Image>() {{
        add(new DummyImage());
        add(new DummyImage());
    }};
```

It's one out of two bugs that sonarqube is locating in our code. But for us those "bugs" are okay. We are grateful to have this warning but we decided to ignore it as we are well aware that we want this inline initialization It's shorter and much more readable than some extra initialization method. Maybe one can do it better..
Update: It's funny but later on we really had a bug doing the initialization like in the picture above So we fixed it by introducing an static initialization method for empty images. Thanks SonarQube <3

So.. It's nice

After we clicked us through the dashboard of sonarqube we became apparent that it helps us a lot to find some hotspots where we should reconsider coding decisions or to find vulnerabilities we didn't even think about. As last example sonarqube gave us a hint that we should probably use a POJO instead of our Database DTO's for our controller as it might be a difficult security issue. This warning also had some explanation text why it was marked. Pretty cool!

Test-Plan

We will soon update our testplan. You will find it here.

SAD

We also added the quality and metrics into our SAD here.

We are well aware that the badges for sonarqube aren't displayed correctly on GitLab. That's a strange behaviour that only occurs on GitLab and not in our local drafts or our local preview. At least the badges are linking to Sonarqube… We are working on that!
Update: we finally fixed it.

Best wishes,

bookly