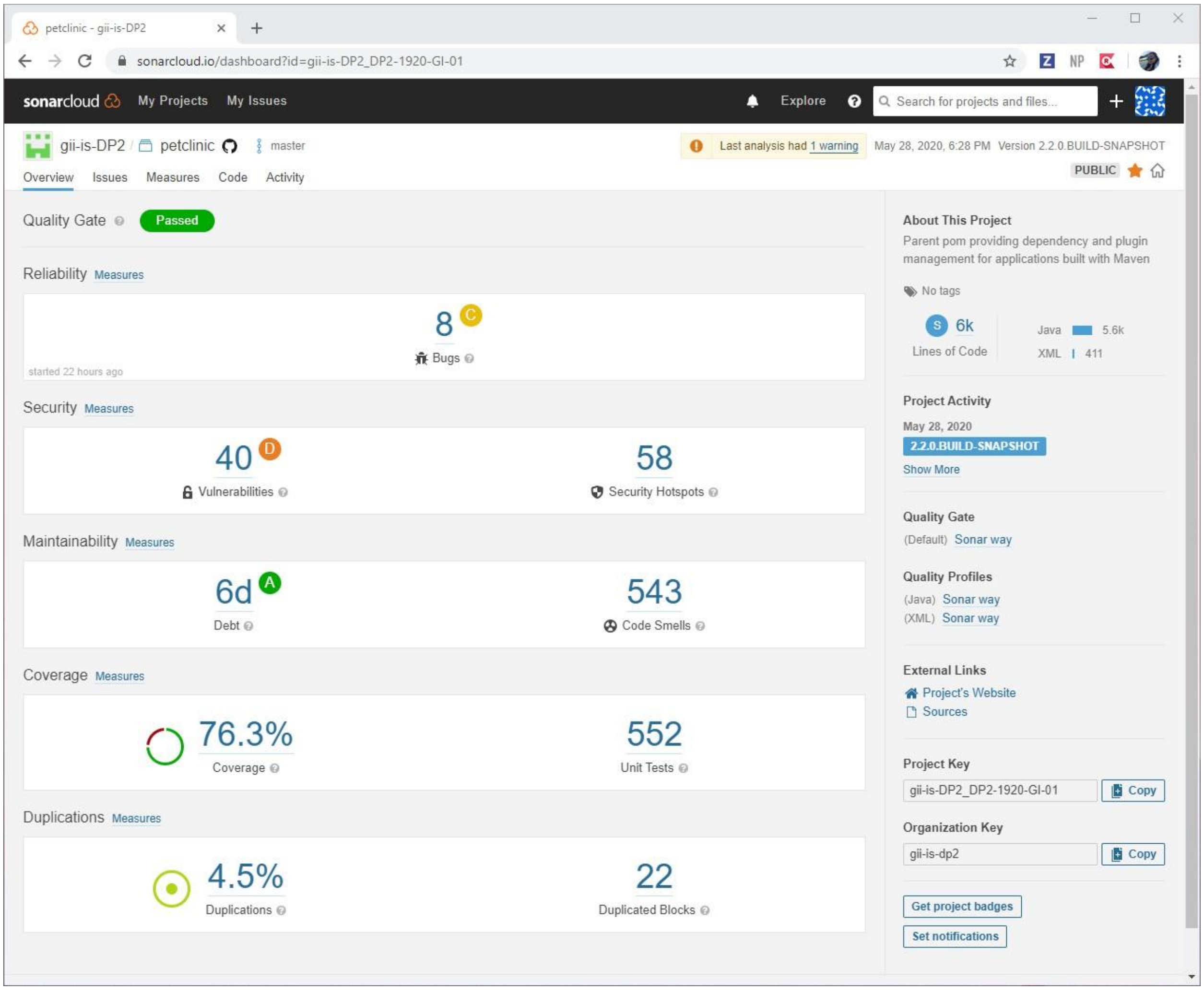
# DP2 – REFACTORING

REPORT

Yoana Dimitrova Penkova
yoadimpen@alum.us.es

For the refactoring tasks for this deliverable, we had to use some analysis tool. In this case, we used sonar cloud which was very easy to configure and sync with the GitHub accounts. We haven't synced it with Travis CI because I didn't see it that necessary as we're in the final stages of the project. By making manual analysis I thought it suffices.

Therefore, after performing the first analysis of the project, the situation was a little chaotic, as you can observe in the following screenshot:



gii-is-DP2 / petclinic master

Last analysis had 1 warning May 28, 2020, 6:28 PM Version 2.2.0.BUILD-SNAPSHOT

PUBLIC

Quality Gate Passed

Reliability Measures

8 C Bugs

started 22 hours ago

Security Measures

40 D Vulnerabilities

58 Security Hotspots

Maintainability Measures

6d A Debt

543 Code Smells

Coverage Measures

76.3% Coverage

552 Unit Tests

Duplications Measures

4.5% Duplications

22 Duplicated Blocks

About This Project

Parent pom providing dependency and plugin management for applications built with Maven

No tags

S 6k Lines of Code

| | |
|---|---|
| Java | 5.6k |
| XML | 411 |

Project Activity

May 28, 2020

2.2.0.BUILD-SNAPSHOT

Show More

Quality Gate

(Default) Sonar way

Quality Profiles

(Java) Sonar way

(XML) Sonar way

External Links

Project's Website

Sources

Project Key

gii-is-DP2_DP2-1920-GI-01 Copy

Organization Key

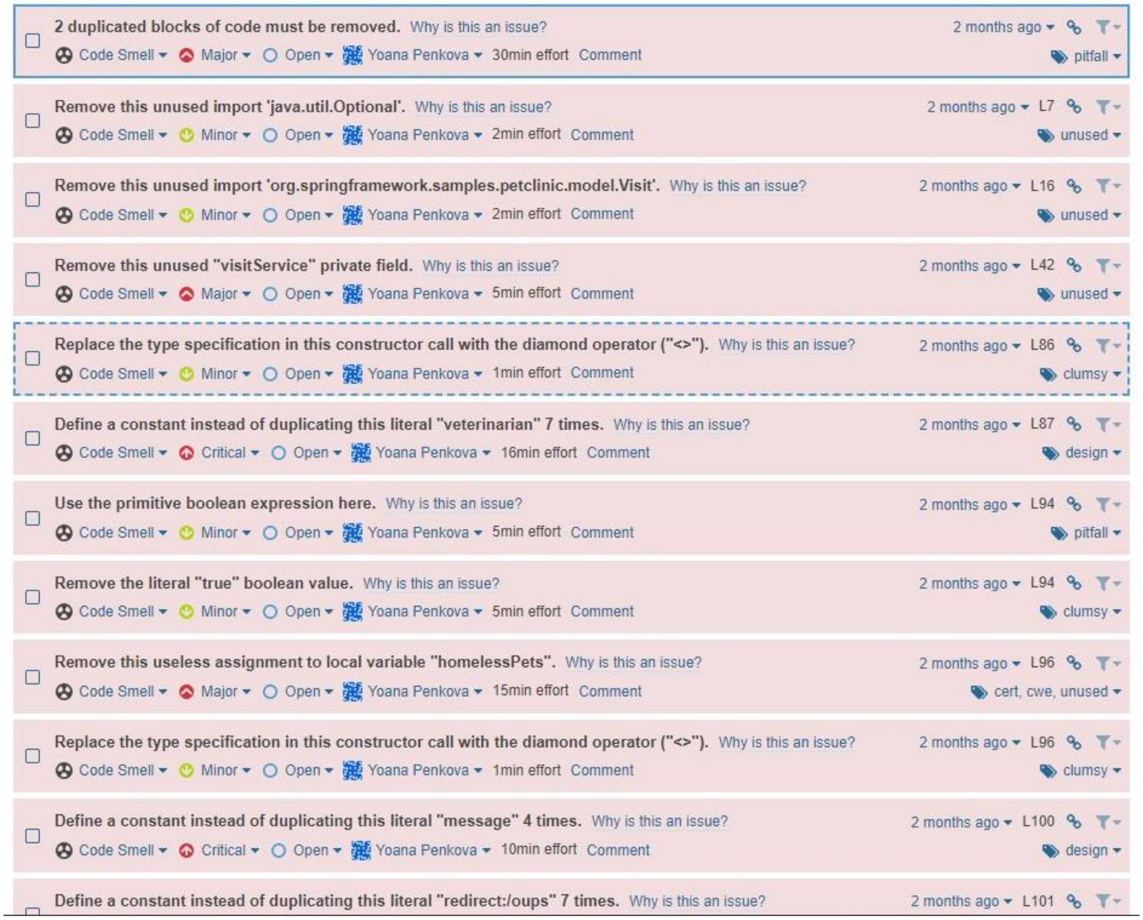
gii-is-dp2 Copy

Get project badges

Set notifications

The first thing that can catch your eye is those 6 days of technical debt and those 543 code smells. Most of the code smells were minor or just information but they had to be fixed as well. I was in charge of fixing the code smells related to the US 011 – Homeless Pet Management. This user story had to simple scenarios of creating a pet with good and wrong values. But, I wanted to go a bit further than that and try to fix the code smells related to not only, creating the homeless pet, but dealing with its interventions, visits, rehabs, etc. So, for this reason, in our project, the main files were most of the code smells (regarding this US) were centered in the controllers and some of the services.

Hereunder, I'll show the code smells related to the HomelessPetController.java file in our project just because the other homeless pet controllers have the same code smells and seems unnecessary to add them too (they are fixed in the same way though).



□ 2 duplicated blocks of code must be removed. Why is this an issue? — 2 months ago
Code Smell Major Open Yoana Penkova 30min effort Comment — pitfall

□ Remove this unused import 'java.util.Optional'. Why is this an issue? — 2 months ago L7
Code Smell Minor Open Yoana Penkova 2min effort Comment — unused

□ Remove this unused import 'org.springframework.samples.petclinic.model.Visit'. Why is this an issue? — 2 months ago L16
Code Smell Minor Open Yoana Penkova 2min effort Comment — unused

□ Remove this unused "visitService" private field. Why is this an issue? — 2 months ago L42
Code Smell Major Open Yoana Penkova 5min effort Comment — unused

□ Replace the type specification in this constructor call with the diamond operator ("<>"). Why is this an issue? — 2 months ago L86
Code Smell Minor Open Yoana Penkova 1min effort Comment — clumsy

□ Define a constant instead of duplicating this literal "veterinarian" 7 times. Why is this an issue? — 2 months ago L87
Code Smell Critical Open Yoana Penkova 16min effort Comment — design

□ Use the primitive boolean expression here. Why is this an issue? — 2 months ago L94
Code Smell Minor Open Yoana Penkova 5min effort Comment — pitfall

□ Remove the literal "true" boolean value. Why is this an issue? — 2 months ago L94
Code Smell Minor Open Yoana Penkova 5min effort Comment — clumsy

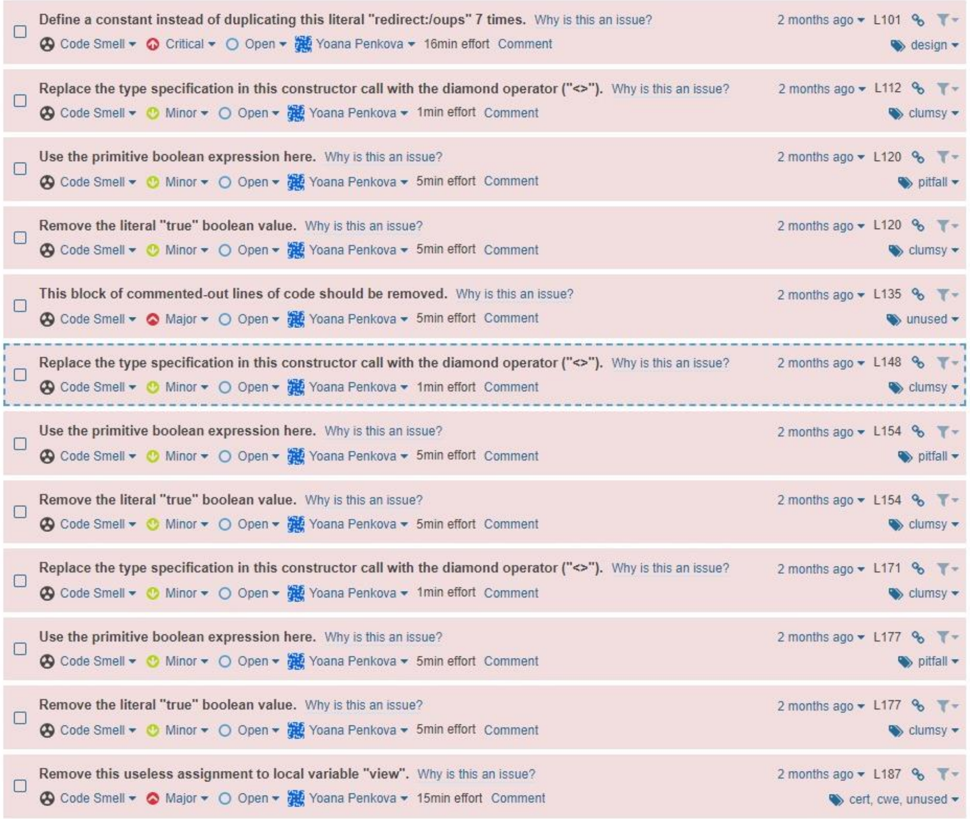
□ Remove this useless assignment to local variable "homelessPets". Why is this an issue? — 2 months ago L96
Code Smell Major Open Yoana Penkova 15min effort Comment — cert, cwe, unused

□ Replace the type specification in this constructor call with the diamond operator ("<>"). Why is this an issue? — 2 months ago L96
Code Smell Minor Open Yoana Penkova 1min effort Comment — clumsy

□ Define a constant instead of duplicating this literal "message" 4 times. Why is this an issue? — 2 months ago L100
Code Smell Critical Open Yoana Penkova 10min effort Comment — design

□ Define a constant instead of duplicating this literal "redirect:/oups" 7 times. Why is this an issue? — 2 months ago L101

It continues in the next screenshots.



□ **Define a constant instead of duplicating this literal "redirect:/oups" 7 times.** Why is this an issue? 2 months ago ▾ L101
Code Smell ▾ Critical ▾ ○ Open ▾ Yoana Penkova ▾ 16min effort Comment design ▾

□ **Replace the type specification in this constructor call with the diamond operator ("<>").** Why is this an issue? 2 months ago ▾ L112
Code Smell ▾ Minor ▾ ○ Open ▾ Yoana Penkova ▾ 1min effort Comment clumsy ▾

□ **Use the primitive boolean expression here.** Why is this an issue? 2 months ago ▾ L120
Code Smell ▾ Minor ▾ ○ Open ▾ Yoana Penkova ▾ 5min effort Comment pitfall ▾

□ **Remove the literal "true" boolean value.** Why is this an issue? 2 months ago ▾ L120
Code Smell ▾ Minor ▾ ○ Open ▾ Yoana Penkova ▾ 5min effort Comment clumsy ▾

□ **This block of commented-out lines of code should be removed.** Why is this an issue? 2 months ago ▾ L135
Code Smell ▾ Major ▾ ○ Open ▾ Yoana Penkova ▾ 5min effort Comment unused ▾

□ **Replace the type specification in this constructor call with the diamond operator ("<>").** Why is this an issue? 2 months ago ▾ L148
Code Smell ▾ Minor ▾ ○ Open ▾ Yoana Penkova ▾ 1min effort Comment clumsy ▾

□ **Use the primitive boolean expression here.** Why is this an issue? 2 months ago ▾ L154
Code Smell ▾ Minor ▾ ○ Open ▾ Yoana Penkova ▾ 5min effort Comment pitfall ▾

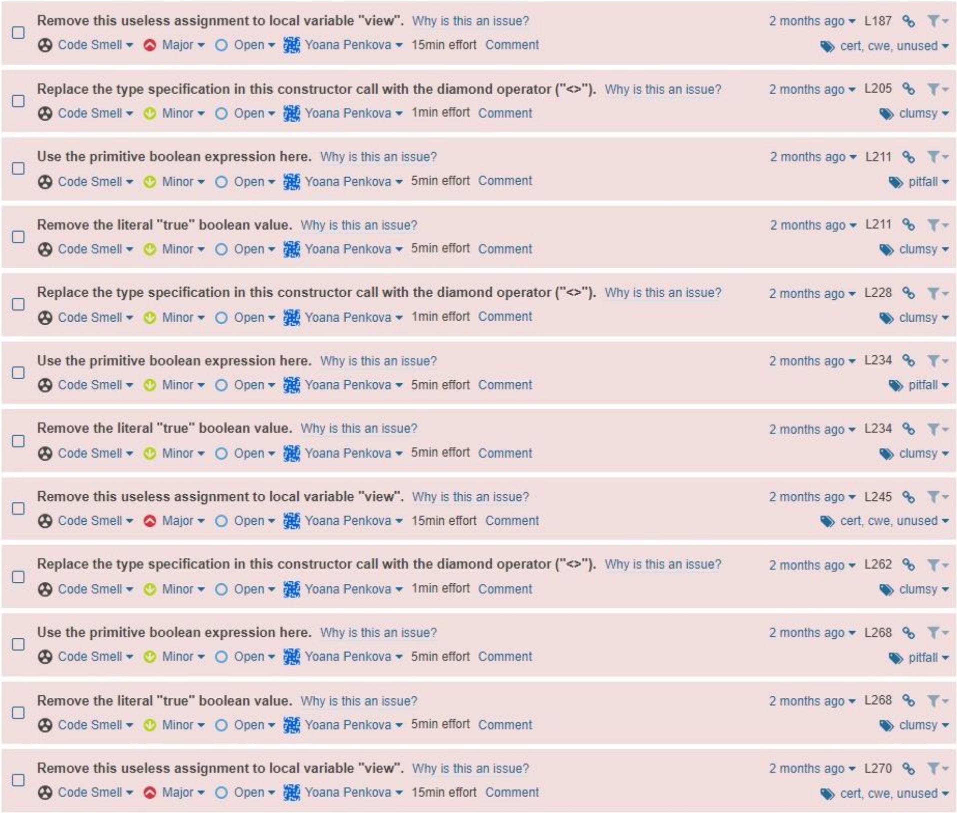
□ **Remove the literal "true" boolean value.** Why is this an issue? 2 months ago ▾ L154
Code Smell ▾ Minor ▾ ○ Open ▾ Yoana Penkova ▾ 5min effort Comment clumsy ▾

□ **Replace the type specification in this constructor call with the diamond operator ("<>").** Why is this an issue? 2 months ago ▾ L171
Code Smell ▾ Minor ▾ ○ Open ▾ Yoana Penkova ▾ 1min effort Comment clumsy ▾

□ **Use the primitive boolean expression here.** Why is this an issue? 2 months ago ▾ L177
Code Smell ▾ Minor ▾ ○ Open ▾ Yoana Penkova ▾ 5min effort Comment pitfall ▾

□ **Remove the literal "true" boolean value.** Why is this an issue? 2 months ago ▾ L177
Code Smell ▾ Minor ▾ ○ Open ▾ Yoana Penkova ▾ 5min effort Comment clumsy ▾

□ **Remove this useless assignment to local variable "view".** Why is this an issue? 2 months ago ▾ L187
Code Smell ▾ Major ▾ ○ Open ▾ Yoana Penkova ▾ 15min effort Comment cert, cwe, unused ▾



□ **Remove this useless assignment to local variable "view".** Why is this an issue? 2 months ago ▾ L187
Code Smell ▾ Major ▾ ○ Open ▾ Yoana Penkova ▾ 15min effort Comment cert, cwe, unused ▾

□ **Replace the type specification in this constructor call with the diamond operator ("<>").** Why is this an issue? 2 months ago ▾ L205
Code Smell ▾ Minor ▾ ○ Open ▾ Yoana Penkova ▾ 1min effort Comment clumsy ▾

□ **Use the primitive boolean expression here.** Why is this an issue? 2 months ago ▾ L211
Code Smell ▾ Minor ▾ ○ Open ▾ Yoana Penkova ▾ 5min effort Comment pitfall ▾

□ **Remove the literal "true" boolean value.** Why is this an issue? 2 months ago ▾ L211
Code Smell ▾ Minor ▾ ○ Open ▾ Yoana Penkova ▾ 5min effort Comment clumsy ▾

□ **Replace the type specification in this constructor call with the diamond operator ("<>").** Why is this an issue? 2 months ago ▾ L228
Code Smell ▾ Minor ▾ ○ Open ▾ Yoana Penkova ▾ 1min effort Comment clumsy ▾

□ **Use the primitive boolean expression here.** Why is this an issue? 2 months ago ▾ L234
Code Smell ▾ Minor ▾ ○ Open ▾ Yoana Penkova ▾ 5min effort Comment pitfall ▾

□ **Remove the literal "true" boolean value.** Why is this an issue? 2 months ago ▾ L234
Code Smell ▾ Minor ▾ ○ Open ▾ Yoana Penkova ▾ 5min effort Comment clumsy ▾

□ **Remove this useless assignment to local variable "view".** Why is this an issue? 2 months ago ▾ L245
Code Smell ▾ Major ▾ ○ Open ▾ Yoana Penkova ▾ 15min effort Comment cert, cwe, unused ▾

□ **Replace the type specification in this constructor call with the diamond operator ("<>").** Why is this an issue? 2 months ago ▾ L262
Code Smell ▾ Minor ▾ ○ Open ▾ Yoana Penkova ▾ 1min effort Comment clumsy ▾

□ **Use the primitive boolean expression here.** Why is this an issue? 2 months ago ▾ L268
Code Smell ▾ Minor ▾ ○ Open ▾ Yoana Penkova ▾ 5min effort Comment pitfall ▾

□ **Remove the literal "true" boolean value.** Why is this an issue? 2 months ago ▾ L268
Code Smell ▾ Minor ▾ ○ Open ▾ Yoana Penkova ▾ 5min effort Comment clumsy ▾

□ **Remove this useless assignment to local variable "view".** Why is this an issue? 2 months ago ▾ L270
Code Smell ▾ Major ▾ ○ Open ▾ Yoana Penkova ▾ 15min effort Comment cert, cwe, unused ▾

As you can see in the previous screenshots, we can identify the following main code smells:

- Removing unused imports.
- Removing unused fields.
- Duplication of literals instead of defining a constant for them.
- Replacing things like ArrayList<String> for just ArrayList<>.
- Removing useless assignment to variables.
- Removing the “true” from code pieces like (someBooleanValue == true).

All of these were very easy to fix. Regarding the removing part there is not much to comment about, I just had to remove them because they weren’t used in any part of this class. Concerning the duplication of literals, I defined constants like the following one:

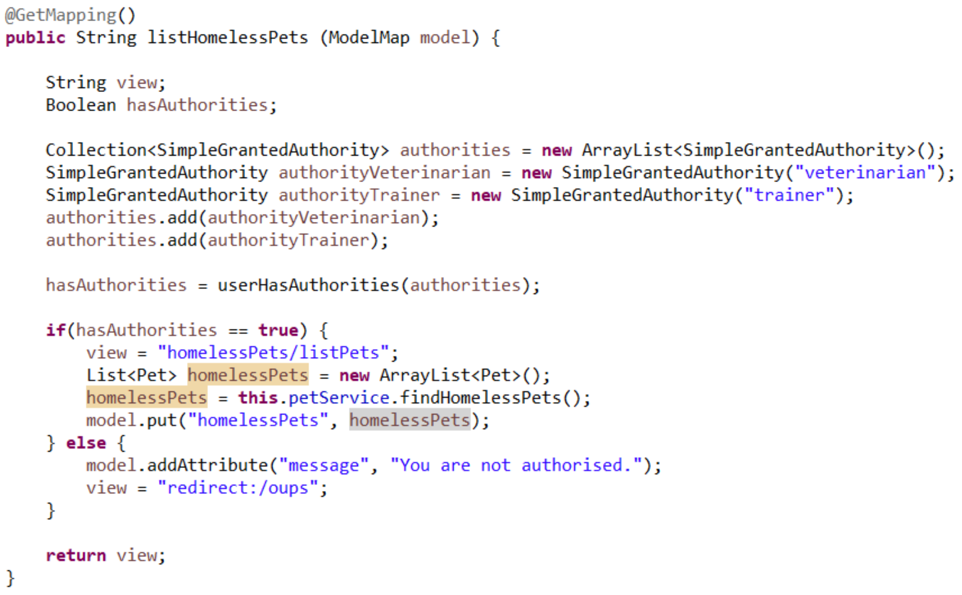
```
private static final String EDIT_FORM = "homelessPets/editPet";
```

And this constant is used every time we want that view, instead of writing “homelessPets/editPet”.

The replacement in regard to “ArrayList<String>  ->  ArrayList<String>” was because of something that I dealt with in the first deliverable. I recall in the unit testing, at first we had to be able to check the security of the system as well but that wasn’t quite possible having these annotations in the test classes:

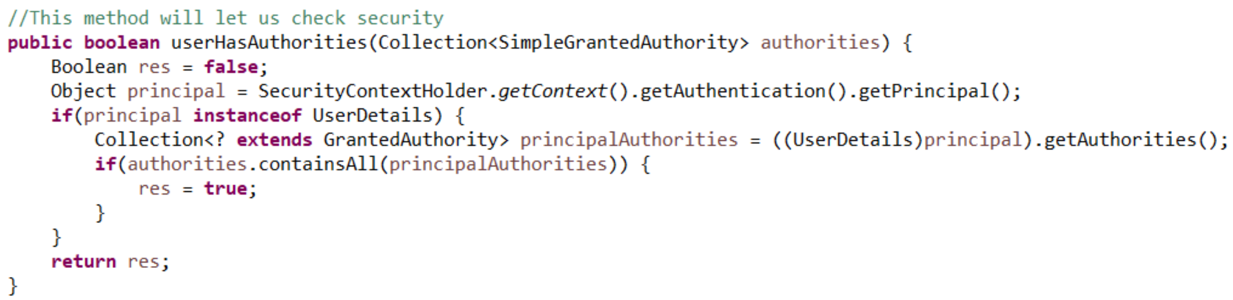
```
@WebMvcTest(value = HomelessPetController.class,
       includeFilters = @ComponentScan.Filter(value = PetTypeFormatter.class,
                                              type = FilterType.ASSIGNABLE_TYPE),
       excludeFilters = @ComponentScan.Filter(type = FilterType.ASSIGNABLE_TYPE,
                                              classes = WebSecurityConfigurer.class),
       excludeAutoConfiguration= SecurityConfiguration.class)
```

These annotations specify that the security configuration of our system shouldn’t be taken into account while performing the unit testing. Therefore, we couldn’t really check that it was functioning properly. I managed to think of a way in which we could check it and it was by adding some lines in each method of the controller. Let’s see the listHomelessPets() method as an example of this:



```
@GetMapping()
public String listHomelessPets (ModelMap model) {

    String view;
    Boolean hasAuthorities;

    Collection<SimpleGrantedAuthority> authorities = new ArrayList<SimpleGrantedAuthority>();
    SimpleGrantedAuthority authorityVeterinarian = new SimpleGrantedAuthority("veterinarian");
    SimpleGrantedAuthority authorityTrainer = new SimpleGrantedAuthority("trainer");
    authorities.add(authorityVeterinarian);
    authorities.add(authorityTrainer);

    hasAuthorities = userHasAuthorities(authorities);

    if(hasAuthorities == true) {
        view = "homelessPets/listPets";
        List<Pet> homelessPets = new ArrayList<Pet>();
        homelessPets = this.petService.findHomelessPets();
        model.put("homelessPets", homelessPets);
    } else {
        model.addAttribute("message", "You are not authorised.");
        view = "redirect:/oups";
    }

    return view;
}
```

We can see that the first lines of it are there to check whether the user trying to access that has the permissions to do it. There's an additional method used here which is userHasAuthorities() which you can check in the next screenshot:



```
//This method will let us check security
public boolean userHasAuthorities(Collection<SimpleGrantedAuthority> authorities) {
    Boolean res = false;
    Object principal = SecurityContextHolder.getContext().getAuthentication().getPrincipal();
    if(principal instanceof UserDetails) {
        Collection<? extends GrantedAuthority> principalAuthorities = ((UserDetails)principal).getAuthorities();
        if(authorities.containsAll(principalAuthorities)) {
            res = true;
        }
    }
    return res;
}
```

This method basically gets the security context and the user's authorities and sees if she/he has permissions to access the URL. If it does, it returns true, otherwise it returns false. It hasn't anything to do with the code smell but I had to explain where were those lines coming from.

The refactoring of that method in particular ended up looking like this:

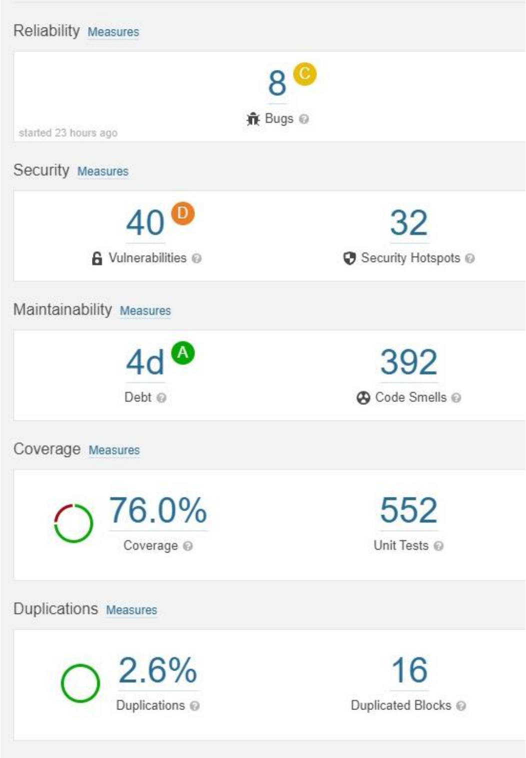
```
@GetMapping()
        public String listHomelessPets (ModelMap model) {

                String view;
                List<String> authorities = new ArrayList<>();
                Boolean hasAuthorities;

                authorities.add(VETERINARIAN);
                authorities.add(TRAINER);

                hasAuthorities = userHasAuthorities(makeAuthorities(authorities));

                if(Boolean.TRUE.equals(hasAuthorities)) {
                        view = "homelessPets/listPets";
                        List<Pet> homelessPets = this.petService.findHomelessPets();
                        model.put("homelessPets", homelessPets);
                } else {
                        model.addAttribute(MESSAGE, "You are not authorised.");
                        view = REDIRECT_URL;
                }

                return view;
        }
```

And since this had to be done in each method I extracted some of the code in a new method which was makeAuthorities():

```
public Collection<SimpleGrantedAuthority> makeAuthorities(List<String> authoritiesString) {
                Collection<SimpleGrantedAuthority> authorities = new ArrayList<>();
                for (String s: authoritiesString) {
                        SimpleGrantedAuthority authority = new SimpleGrantedAuthority(s);
                        authorities.add(authority);
                }
                return authorities;
        }
```

Now in each one of those controller methods the only thing we have to pass is a list of strings which will be transformed in a collection of SimpleGrantedAuthority objects which will be processed by userHasAuthorities().

These kind of modifications had to be made in more classes and it took around an hour or two to solve them. After committing and pushing the changes, merging them and performing a second analysis (a second and third actually because I had forgotten something) the results we get from sonar cloud were way better than those from the first analysis. You can check them in the next screenshots:



Reliability Measures

8 C

Bugs

started 23 hours ago

Security Measures

40 D

Vulnerabilities

32

Security Hotspots

Maintainability Measures

4d A

Debt

392

Code Smells

Coverage Measures

76.0%

Coverage

552

Unit Tests

Duplications Measures

2.6%

Duplications

16

Duplicated Blocks

The improvements from the first analysis are the following:

- From 58 to 32 security hotspots, 26 less.
- From 543 to 392 code smells, 151 less.
- From 6 days debt to 4 days debt, 2 whole days less.
- From 22 duplicated blocks of code to 16, 6 less.

And all that with just one refactoring session. I'm sure that with the aid of my colleagues, the last analysis that we will perform will be splendid.