Report on Static Analysis

Choosing a Project

Choosing an open-source project, that fits into the scope of the assignment for this week, is no trivial task. We are looking for a small, less popular project written in Java. Small, because it suffices in demonstrating the capabilities of a static linter like SonarLint, and less popular since I assume that linters are less likely to have been used there.

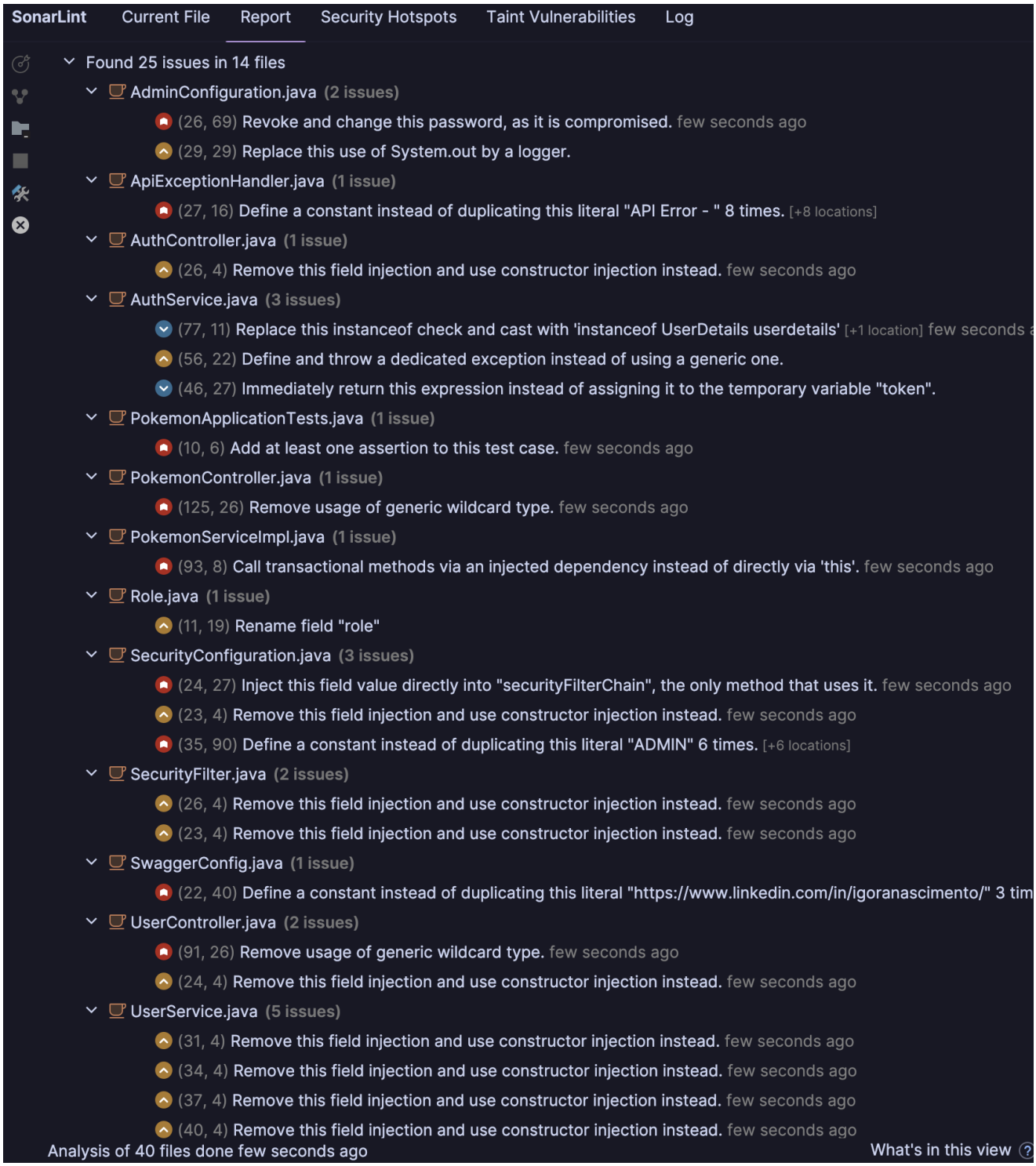
The sort-by-topic search query on GitHub yields limited query options and Google-ing for a small Java-based project, which is less popular, yielded results, that are either too in-depth in topic-wise, e.g. stenography tools, or poorly maintained, e.g. WIP projects without maintenance.

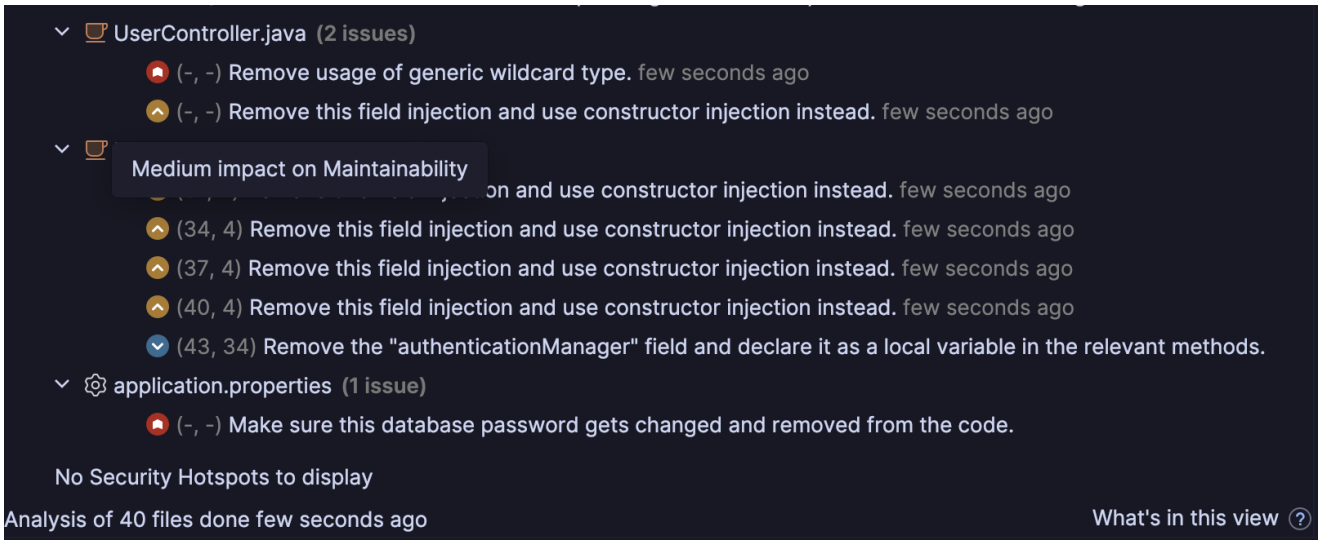
After a few hours of searching, I found a young project, which does support Maven and uses Spring as a Backend framework. Its only downside is its lack of tests, but for demonstrating the capabilities of a static linter, it is a suitable option.

The project is called GitHub - notAvoiid/pokedex-pokemon: This API returns a complete pokedex! (Commit `90897f5887a6d85058be3f8e204d4f2f4ebab7f5`) and a fork is available under valerius21/pokemon-crud.

Fixing SonarLint-related Issues

Running SonarLint on the projects' root folder yielded the following suggestions:


SonarLint
Current File
Report
Security Hotspots
Taint Vulnerabilities
Log
Found 25 issues in 14 files
AdminConfiguration.java (2 issues)
(26, 69) Revoke and change this password, as it is compromised. few seconds ago
(29, 29) Replace this use of System.out by a logger.
ApiExceptionHandler.java (1 issue)
(27, 16) Define a constant instead of duplicating this literal "API Error - " 8 times. [+8 locations]
AuthController.java (1 issue)
(26, 4) Remove this field injection and use constructor injection instead. few seconds ago
AuthService.java (3 issues)
(77, 11) Replace this instanceof check and cast with 'instanceof UserDetails userdetails' [+1 location] few seconds
(56, 22) Define and throw a dedicated exception instead of using a generic one.
(46, 27) Immediately return this expression instead of assigning it to the temporary variable "token".
PokemonApplicationTests.java (1 issue)
(10, 6) Add at least one assertion to this test case. few seconds ago
PokemonController.java (1 issue)
(125, 26) Remove usage of generic wildcard type. few seconds ago
PokemonServiceImpl.java (1 issue)
(93, 8) Call transactional methods via an injected dependency instead of directly via 'this'. few seconds ago
Role.java (1 issue)
(11, 19) Rename field "role"
SecurityConfiguration.java (3 issues)
(24, 27) Inject this field value directly into "securityFilterChain", the only method that uses it. few seconds ago
(23, 4) Remove this field injection and use constructor injection instead. few seconds ago
(35, 90) Define a constant instead of duplicating this literal "ADMIN" 6 times. [+6 locations]
SecurityFilter.java (2 issues)
(26, 4) Remove this field injection and use constructor injection instead. few seconds ago
(23, 4) Remove this field injection and use constructor injection instead. few seconds ago
SwaggerConfig.java (1 issue)
(22, 40) Define a constant instead of duplicating this literal "https://www.linkedin.com/in/igoranascimento/" 3 tim
UserController.java (2 issues)
(91, 26) Remove usage of generic wildcard type. few seconds ago
(24, 4) Remove this field injection and use constructor injection instead. few seconds ago
UserService.java (5 issues)
(31, 4) Remove this field injection and use constructor injection instead. few seconds ago
(34, 4) Remove this field injection and use constructor injection instead. few seconds ago
(37, 4) Remove this field injection and use constructor injection instead. few seconds ago
(40, 4) Remove this field injection and use constructor injection instead. few seconds ago
Analysis of 40 files done few seconds ago
What's in this view


UserController.java (2 issues)
(-, -) Remove usage of generic wildcard type. few seconds ago
(-, -) Remove this field injection and use constructor injection instead. few seconds ago
Medium impact on Maintainability
(34, 4) Remove this field injection and use constructor injection instead. few seconds ago
(37, 4) Remove this field injection and use constructor injection instead. few seconds ago
(40, 4) Remove this field injection and use constructor injection instead. few seconds ago
(43, 34) Remove the "authenticationManager" field and declare it as a local variable in the relevant methods.
application.properties (1 issue)
(-, -) Make sure this database password gets changed and removed from the code.
No Security Hotspots to display
Analysis of 40 files done few seconds ago
What's in this view

Resulting in 24 issues, ranging from security suggestions to minor refactoring recommendations.

The screenshots above indicate the most severe issues in red to less important ones in yellow and blue.

Most notably, a password change is suggested in `AdminConfiguration.java`, since the used password has already been compromised.

Other red issues include maintenance suggestions, which presses the recommendation and practise to reuse values inside variables instead of using literals, `ApiExceptionHandler.java` with "API Error -" being used 8 times.

Moreover, SonarLint tries to steer the developer in the direction to stick with the naming conventions, as seen as in the suggestion for `Role.java`, recommending renaming the field to a suitable alternative.

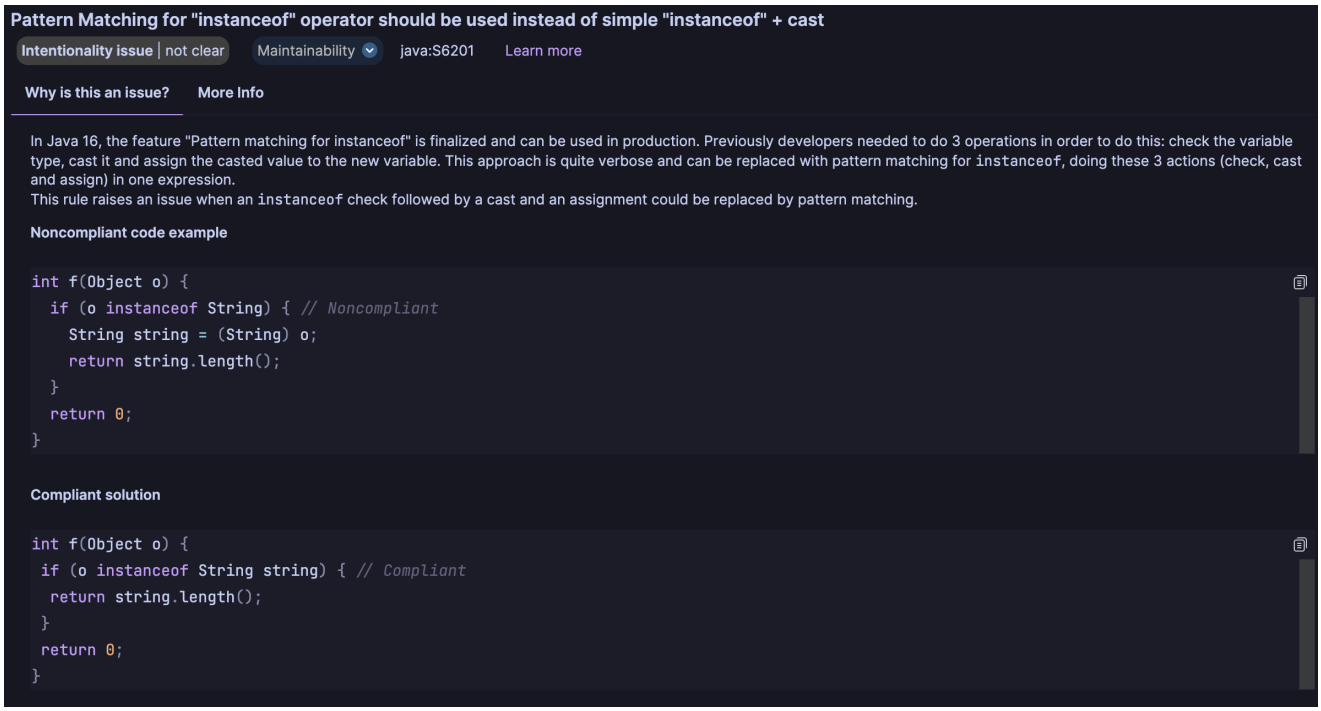
`UserService.java` and `SecurityFilter.java` are marked to remove Injections from the specified attributes and move these to the constructor instead. That means, that SonarLint also finds Framework-Specific issues related to Spring.

With more and more frequent releases of Java versions and syntactic overhauls, SonarLint helps the developer to migrate easily to newer updated versions, helping the developer to produce up-to-date code, an example of which can be seen in `AuthService.java`, showing that the method below can be refactored using modern type-casting:

AuthService.java

```
1       public Optional<User> getAuthUser() {
2           var authentication =
    SecurityContextHolder.getContext().getAuthentication();
3           Object principal = authentication.getPrincipal();
4           if(principal instanceof UserDetails) {
5               String username = ((UserDetails) principal).getUsername();
6           if(principal instanceof UserDetails userDetails) {
7               String username = userDetails.getUsername();
8               return userRepository.loadByUsername(username);
9           } else
10              throw new NullUserException();
11      }
```

> In Java 16, the feature "Pattern matching for instanceof" is finalized and can be used in production. [...]



Another aspect of SonarLint is also shown in the image above, not only linting the issue, but also explaining why this is one and how to fix it. If SonarLint cannot come up with an explanation, most of the time some links to the corresponding documentation is found.

Since creating a report is part of the current week's assignment, I have looked into exporting the result set from the analysis. Since this is currently only possible in using SonarLint in conjunction with SonarCube, which is a cloud-solution offered by the authors of SonarLint, Screenshots as seen above were created to keep track of the fixing process.

Fixing the errors

In the process of fixing all linted issues, I found it helpful to use an implementation of dotenv for Java to address the issues highlighted in `AdminConfiguration.java`, since not only the change of the admin password deemed to be a necessity, but following up to that, SonarLint also recommended to use not-hardcoded credentials, which are then read from the corresponding environment variable.

Using dotenv-java is by no means a requirement, since `System.getEnv` is perfectly able to handle this issue by its own. - It is just a measure of personal taste.

Otherwise, following the recommendations and suggestions provided by SonarLint helped to refactor the code effectively, and all suggested issues. From the observed developer experience, the code seems way more comfortable to refactor and confident to deploy.

Conclusion

After the initial hurdle to find a small-enough but still usable open-source project, SonarLint helped to find some critical errors and code smalls which helps the developer in firstly, finding the right approach to solve the issue or secondly, directly providing solutions through API integrations. All issues were resolved successfully.

The developer can stay focused on the task, while being able to produce up-to-date code, with less security concerns and maintainability issues.

The static analyses from SonarLint compliments the traditional IDE linter and enhances the developer experience and code quality.