HW1: Mid-term assignment report

Ana Alexandra Antunes [876543], v2025-10-23

1 Introduction 1
1.1 Overview of the work 1
1.2 Current implementation (faults & extras) 1

2 Product specification 2
2.1 Functional scope and supported interactions 2
2.2 System implementation architecture 2
2.3 API for developers 2

3 Quality assurance 3
3.1 Overall strategy for testing 3
3.2 Unit and integration testing 3
3.3 Acceptance testing 3
3.4 Non-functional testing 3
3.5 Code quality analysis 3
3.6 Continuous integration pipeline [optional] 4

4 References & resources 4

<All remarks in this color should be removed from the final document!
*This a template for the expected **content/structure**. You may use any editing tool to prepare the report (LaTeX included).*
Feel free to write in Portuguese or English, but do not mix languages between headings and body…>

1 Introduction

1.1 Overview of the work

This report presents the midterm individual project required for TQS, covering both the software product features and the adopted quality assurance strategy.
<briefly introduce your application use case: name the product, if applicable; what is its general purpose, in your words?>

1.2 Current implementation (faults & extras)

<explain the known limitations → features that are missing or faulty but were expected! >

Falar do state pattern, pros and cons (simples enum vs mais autonomia nas transicoes)

< if applicable: clarify which extra features did you implement. Are there relevant additions you decided to include?>

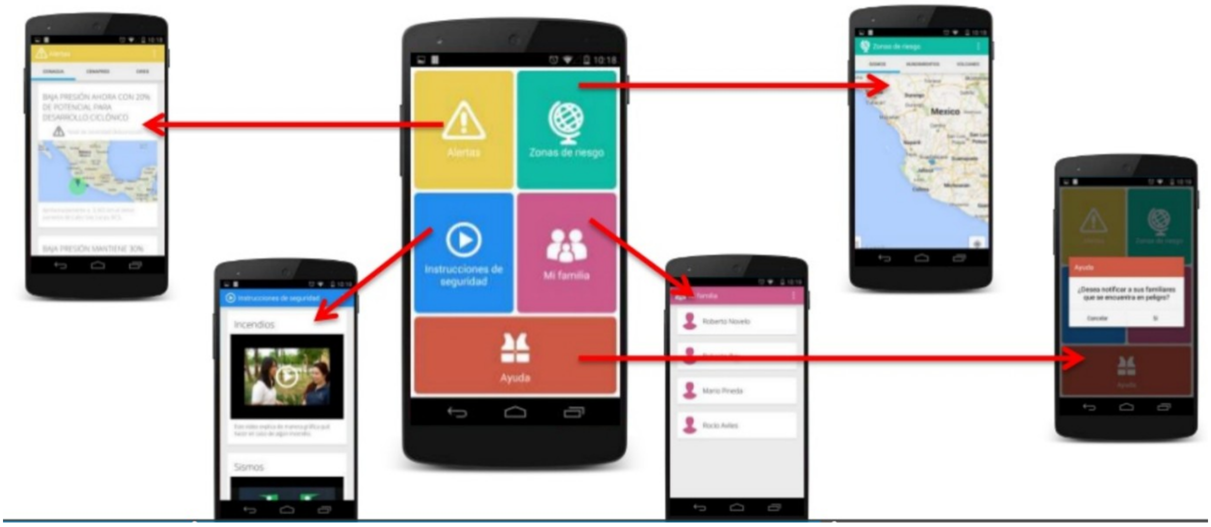
1.3 Use of generative AI

Clarify how did you use AI-assistants for production and test code.
Be clear about the parts you prefer not to delegate.

2 Product specification

2.1 Functional scope and supported interactions

*<who (actors) will use the application and for what? Explain the main **interactions;** include/explain the experience with a visual summary. >*

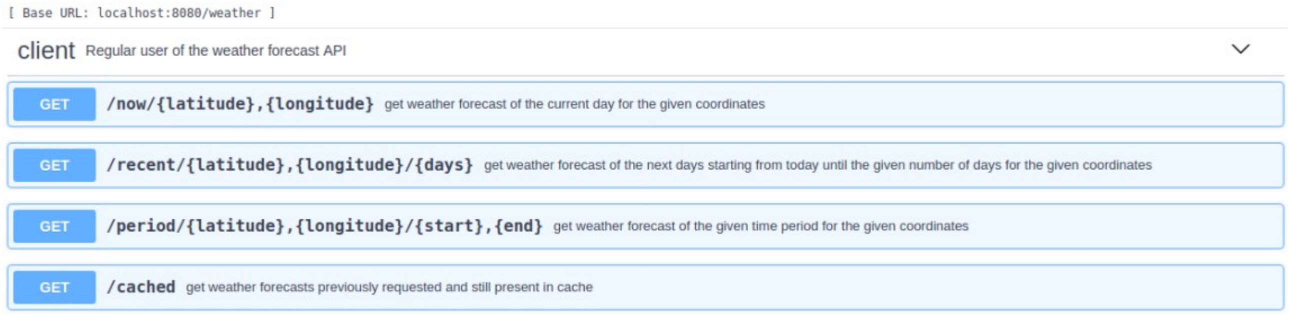


2.2 System implementation architecture

<briefly present the software architecture.>
<detail the specific technologies/frameworks that were used>

2.3 API for developers

<what services/resources can a developer obtain from your project? document your API endpoints>



[Base URL: localhost:8080/weather]

client Regular user of the weather forecast API

GET `/now/{latitude},{longitude}` get weather forecast of the current day for the given coordinates

GET `/recent/{latitude},{longitude}/{days}` get weather forecast of the next days starting from today until the given number of days for the given coordinates

GET `/period/{latitude},{longitude}/{start},{end}` get weather forecast of the given time period for the given coordinates

GET `/cached` get weather forecasts previously requested and still present in cache

3 Quality assurance

3.1 Overall strategy for testing

[what was the overall test development strategy? E.g.: did you do TDD? Did you choose to use Cucumber and BDD? Did you mix different testing tools, like REST-Assured and Cucumber?...]

3.2 Unit and integration testing

[where did you use unit and integration test? for what? which was the implementation strategy?]
[may add some screenshots/code snippets for clarification, but do not dump all tests here….]

3.3 Acceptance testing

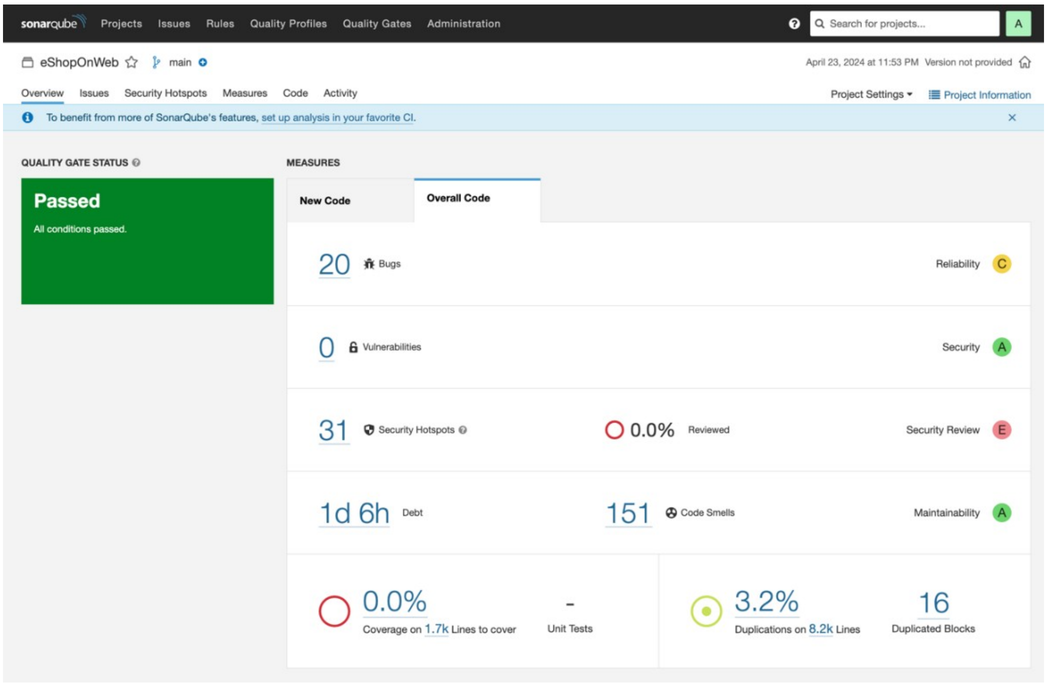
[which user-facing test cases did you considered? How were they implemented and automatized?]
[may add some screenshots/code snippets for clarification]

3.4 Non-functional testing

[which non-functional test cases did you consider? Expected: performance study with, at least, load tests]

3.5 Code quality analysis

[which tools/workflow did you use to for static code analysis?
Show and interpret the results.]
[you may add some interesting lessons learned, e.g., some code smell reported by the tool that was difficult to spot and otherwise you wouldn't address it]



April 23, 2024 at 11:53 PM Version not provided

To benefit from more of SonarQube's features, set up analysis in your favorite CI.

Passed

All conditions passed.

20 Bugs — Reliability C

0 Vulnerabilities — Security A

31 Security Hotspots — 0.0% Reviewed — Security Review E

1d 6h Debt — 151 Code Smells — Maintainability A

0.0% Coverage on 1.7k Lines to cover — - Unit Tests

3.2% Duplications on 8.2k Lines — 16 Duplicated Blocks

3.6 Continuous integration pipeline [optional]

[did you implement a CI pipeline? What was the setup? Illustrate with screenshots, if applicable]

4 References & resources

Project resources

Resource:	URL/location:
Video demo	<short video-demonstration of your solution; copy into the Git folder, under /docs>
QA dashboard (online)	[**optional**; if you have a quality dashboard available online (e.g.: sonarcloud), place the URL here]
CI/CD pipeline	[**optional**; if you have th CI pipeline definition in a server, place the URL here]

Reference materials

*<If applicable: document the key components (e.g.: libraries, API) or key references (e.g.: blog post) that were helpful and certainly **would help other students pursuing a similar work**>*