README

Developed by Beatriz Vaz (1190429)

This folder includes all artifacts developed for the Second Part of QESOFT Project.

This file is structured as follows:

1. Introduction

The objective of the second period of the QESOFT project is to dissect a JHipster application, created from the task utilized in the main stage.

The application will be evaluated based on a number of quality indicators, including: accessibility, compatibility, maintainability, performance and security. Metrics will be measured based on the GQM in the overall report for each quality attribute.

After evaluating the ProductCategoryResource class and its methods, this report will draw its conclusions.

2. Accessibility

Accessibility is the practice of making your websites usable by as many people as possible. Traditionally, accessibility is associated with people with disabilities, but the practice of making sites accessible also benefits other groups such as those using mobile devices, or those with slow network connections. [1]

Six metrics were identified to determinate the accessibility of the application in the GQM presented in the global report.

To measure these metrics, it was used Lighthouse. Lighthouse is an open-source, automated tool for improving the quality of web pages. You can run it against any web page, public or requiring authentication. It has audits for performance, accessibility, progressive web apps, SEO, and more. [2]

With Lighthouse the accessibility audits were checked. The accessibility audits checks highlight opportunities to improve the accessibility of your web app. Only a subset of accessibility issues can be automatically detected so manual testing is also encouraged. [3]

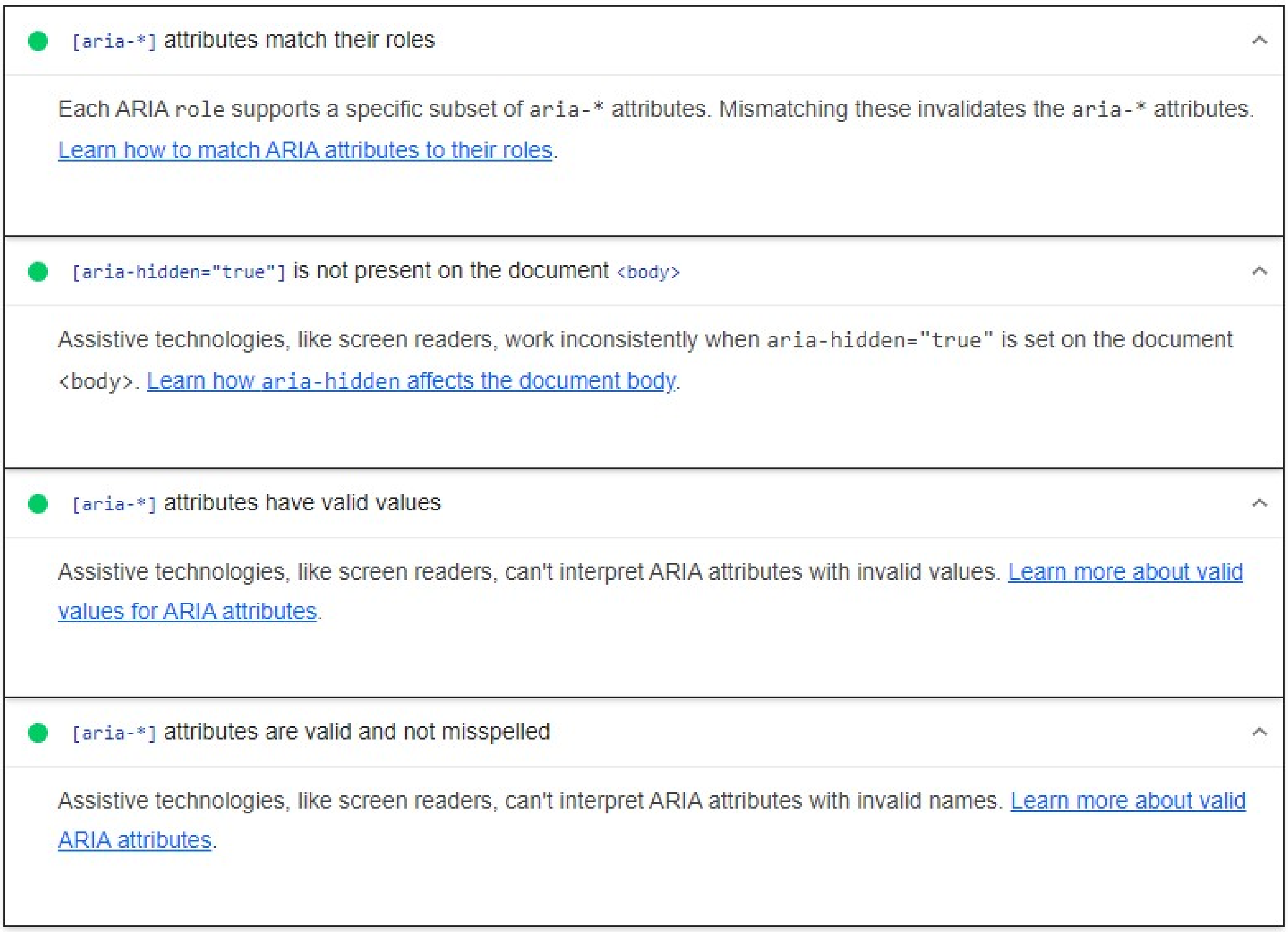
Report generated by LightHouse: Lighthouse Report - Accessibility

Since the global score was 100, it is considered an excellent value.

2.1 Aria

Five audits were found applicable to this metric and all of them passed the tests.

These are opportunities to improve the usage of ARIA in the application which may enhance the experience for users of assistive technology, like a screen reader.



● `[aria-*]` attributes match their roles

Each ARIA `role` supports a specific subset of `aria-*` attributes. Mismatching these invalidates the `aria-*` attributes. Learn how to match ARIA attributes to their roles.

● `[aria-hidden="true"]` is not present on the document `<body>`

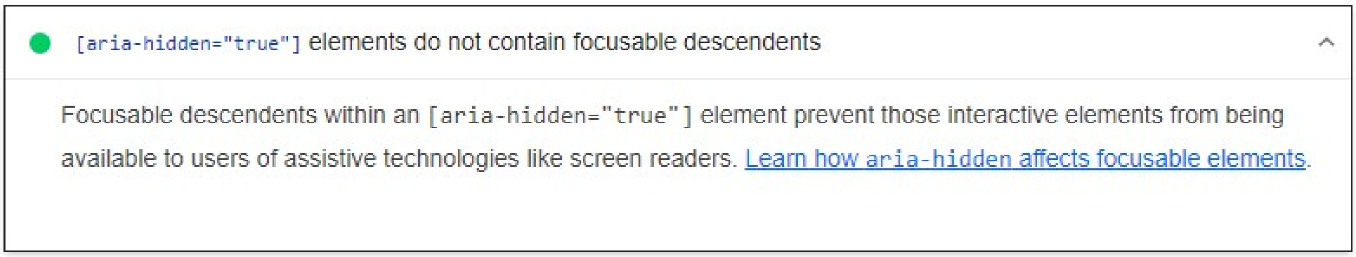
Assistive technologies, like screen readers, work inconsistently when `aria-hidden="true"` is set on the document <body>. Learn how `aria-hidden` affects the document body.

● `[aria-*]` attributes have valid values

Assistive technologies, like screen readers, can't interpret ARIA attributes with invalid values. Learn more about valid values for ARIA attributes.

● `[aria-*]` attributes are valid and not misspelled

Assistive technologies, like screen readers, can't interpret ARIA attributes with invalid names. Learn more about valid ARIA attributes.



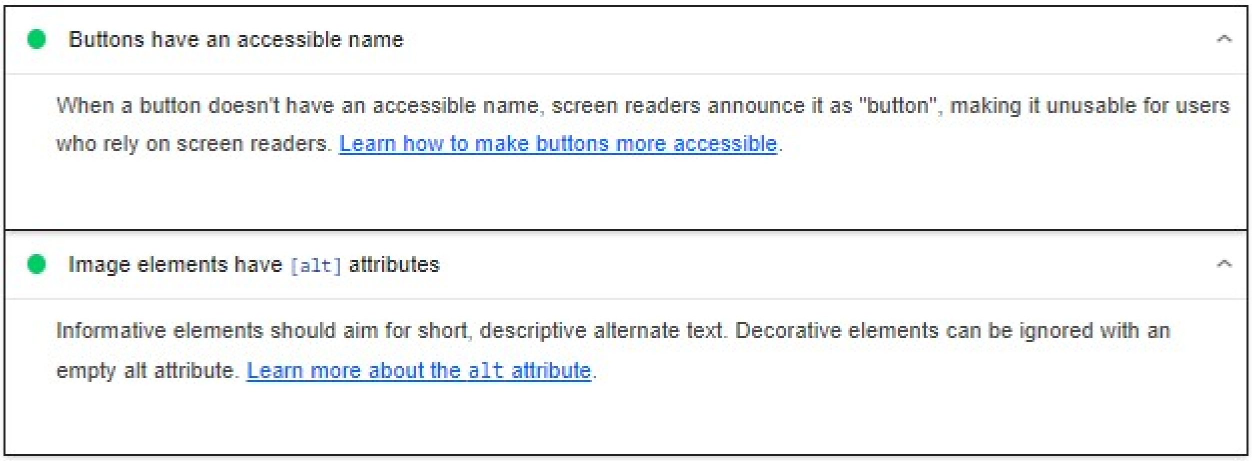
● `[aria-hidden="true"]` elements do not contain focusable descendents

Focusable descendents within an `[aria-hidden="true"]` element prevent those interactive elements from being available to users of assistive technologies like screen readers. Learn how `aria-hidden` affects focusable elements.

2.2 Name and Labels

Four audits were found applicable to this metric and all of them passed the tests.

These are opportunities to improve the semantics of the controls in the application. This may enhance the experience for users of assistive technology, like a screen reader.

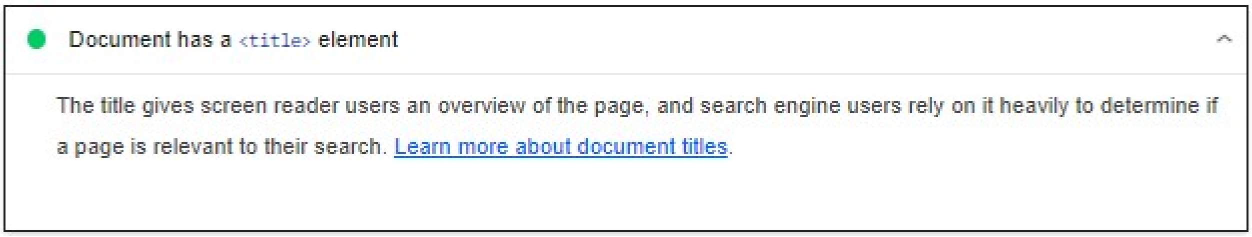


● Buttons have an accessible name

When a button doesn't have an accessible name, screen readers announce it as "button", making it unusable for users who rely on screen readers. Learn how to make buttons more accessible.

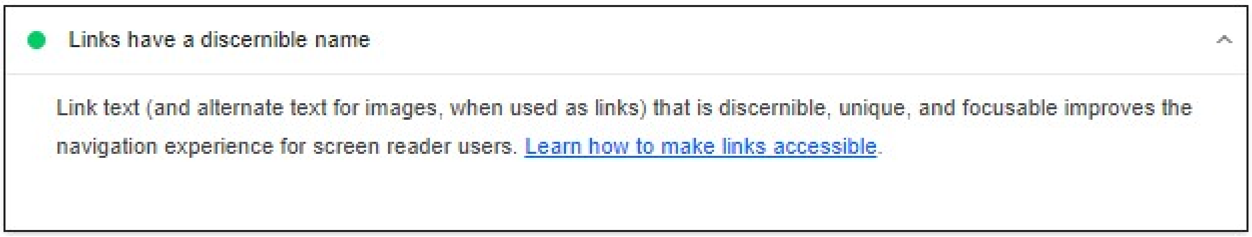
● Image elements have `[alt]` attributes

Informative elements should aim for short, descriptive alternate text. Decorative elements can be ignored with an empty alt attribute. Learn more about the `alt` attribute.



● Document has a `<title>` element

The title gives screen reader users an overview of the page, and search engine users rely on it heavily to determine if a page is relevant to their search. Learn more about document titles.



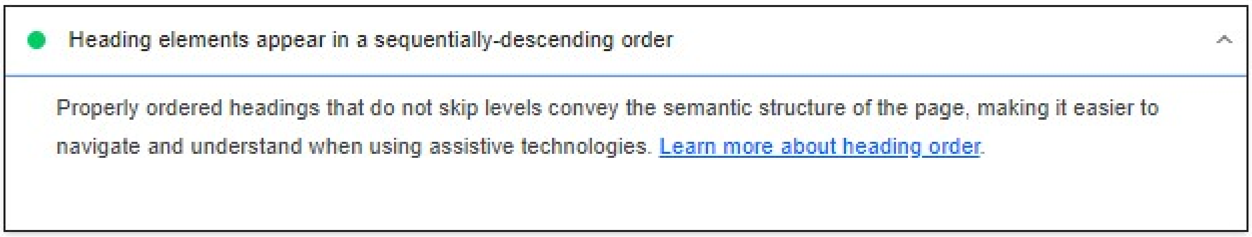
● Links have a discernible name

Link text (and alternate text for images, when used as links) that is discernible, unique, and focusable improves the navigation experience for screen reader users. Learn how to make links accessible.

2.3 Navigation

One audit was found applicable to this metric and passed the test.

These are opportunities to improve keyboard navigation in the application.



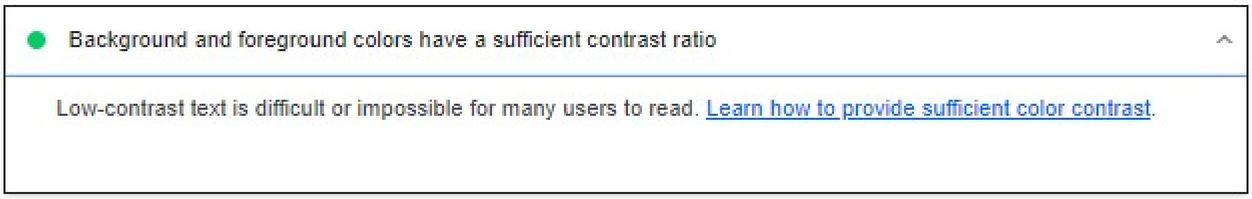
● Heading elements appear in a sequentially-descending order

Properly ordered headings that do not skip levels convey the semantic structure of the page, making it easier to navigate and understand when using assistive technologies. Learn more about heading order.

2.4 Contrast

One audit was found applicable to this metric and passed the test.

These are opportunities to improve the legibility of the content.



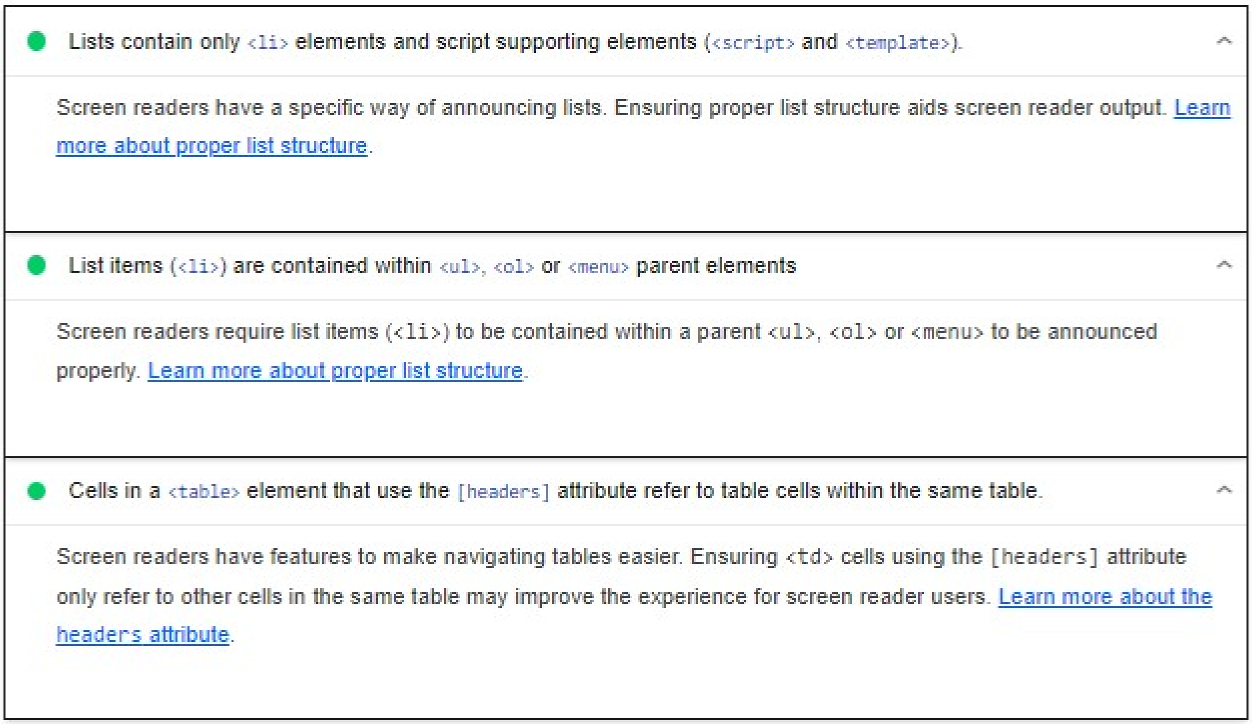
● Background and foreground colors have a sufficient contrast ratio

Low-contrast text is difficult or impossible for many users to read. Learn how to provide sufficient color contrast.

2.5 Tables and Lists

Three audits were found applicable to this metric and all of them passed the tests.

These are opportunities to improve the experience of reading tabular or list data using assistive technology, like a screen reader.



● Lists contain only `<li>` elements and script supporting elements (`<script>` and `<template>`).

Screen readers have a specific way of announcing lists. Ensuring proper list structure aids screen reader output. Learn more about proper list structure.

● List items (`<li>`) are contained within `<ul>`, `<ol>` or `<menu>` parent elements

Screen readers require list items (`<li>`) to be contained within a parent `<ul>`, `<ol>` or `<menu>` to be announced properly. Learn more about proper list structure.

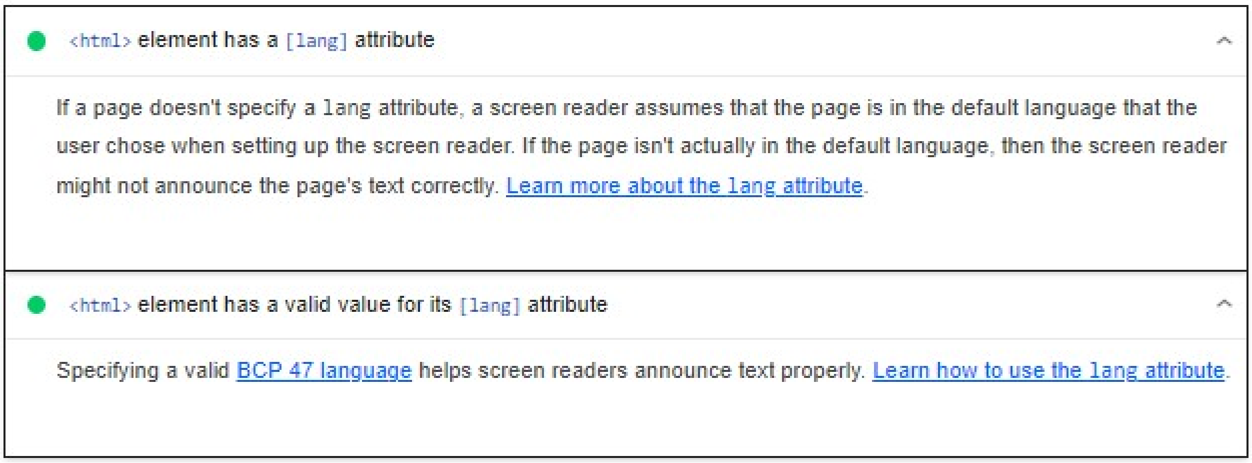
● Cells in a `<table>` element that use the `[headers]` attribute refer to table cells within the same table.

Screen readers have features to make navigating tables easier. Ensuring `<td>` cells using the `[headers]` attribute only refer to other cells in the same table may improve the experience for screen reader users. Learn more about the headers attribute.

2.6 Internationalization and Location

Two audits were found applicable to this metric and all of them passed the tests.

These are opportunities to improve the semantics of the controls in the application. This may enhance the experience for users of assistive technology, like a screen reader.



● `<html>` element has a `[lang]` attribute

If a page doesn't specify a `lang` attribute, a screen reader assumes that the page is in the default language that the user chose when setting up the screen reader. If the page isn't actually in the default language, then the screen reader might not announce the page's text correctly. Learn more about the lang attribute.

● `<html>` element has a valid value for its `[lang]` attribute

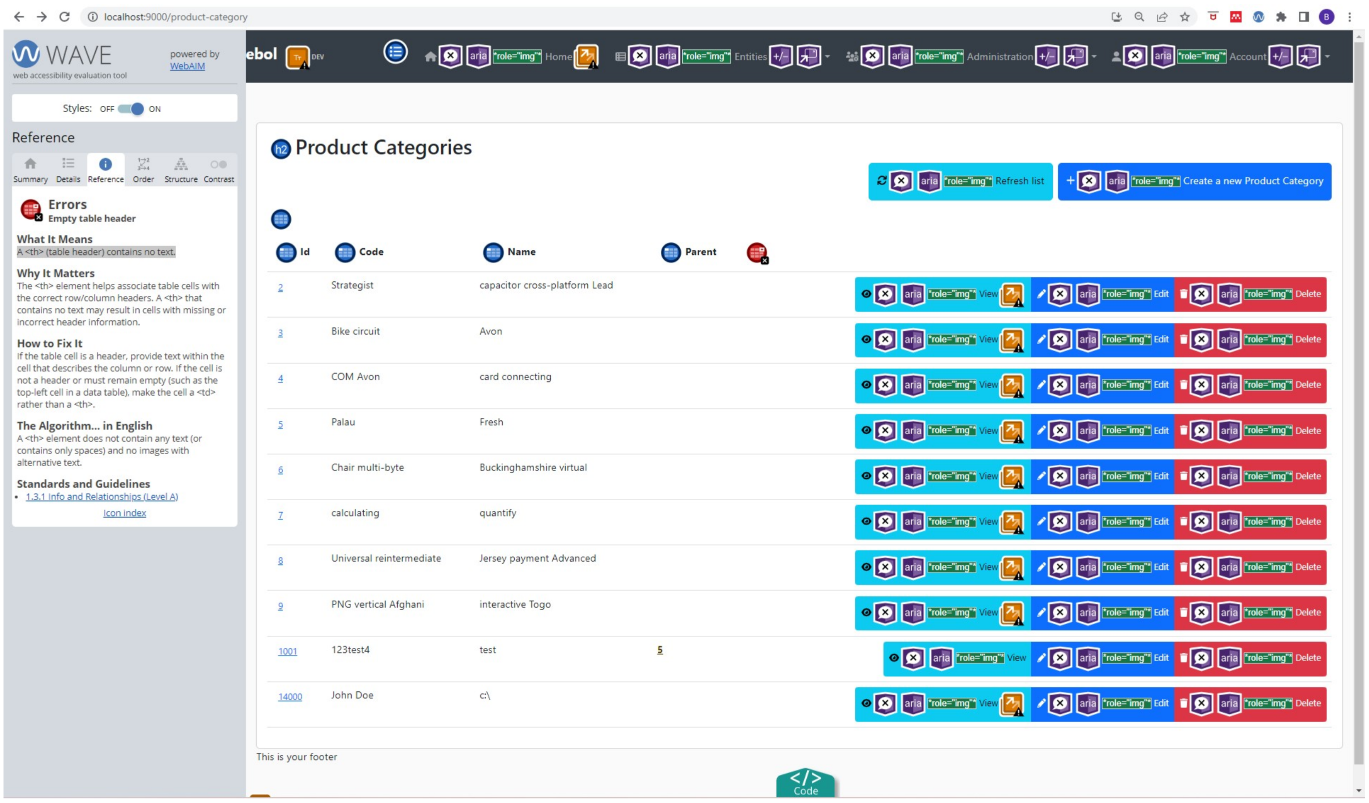
Specifying a valid BCP 47 language helps screen readers announce text properly. Learn how to use the lang attribute.

2.7 WCAG 2.1

To evaluate web accessibility and compare the results to WCAG 2.1 it was used a tool WAVE. [4]

WAVE is a web openness assessment instrument created by WebAIM.org. By incorporating icons and indicators into your page, it provides visual feedback regarding the accessibility of your website content.

Using this tool was possible to visualize the error Empty table header presented bellow:



Reference

Errors
Empty table header

What It Means
A <th> (table header) contains no text.

Why It Matters
The <th> element helps associate table cells with the correct row/column headers. A <th> that contains no text may result in cells with missing or incorrect header information.

How to Fix It
If the table cell is a header, provide text within the cell that describes the column or row. If the cell is not a header or must remain empty (such as the top-left cell in a data table), make the cell a <td> rather than a <th>.

The Algorithm... in English
A <th> element does not contain any text (or contains only spaces) and no images with alternative text.

Standards and Guidelines
- 1.3.1 Info and Relationships (Level A)

Icon index

Product Categories

Id	Code	Name	Parent	
2	Strategist	capacitor cross-platform Lead		View Edit Delete
3	Bike circuit	Avon		View Edit Delete
4	COM Avon	card connecting		View Edit Delete
5	Palau	Fresh		View Edit Delete
6	Chair multi-byte	Buckinghamshire virtual		View Edit Delete
7	calculating	quantify		View Edit Delete
8	Universal reintermediate	Jersey payment Advanced		View Edit Delete
9	PNG vertical Afghani	interactive Togo		View Edit Delete
1001	123test4	test	5	View Edit Delete
14000	John Doe	c:\		View Edit Delete

This is your footer

It is possible to conclude that this page achieves the A grade.

3. Compatibility

To test the level of compatibility of the application, the group decided to use the metrics: Browser Compatibility, Platform Compatibility and Device Compatibility.

3.1 Browser Compatibility

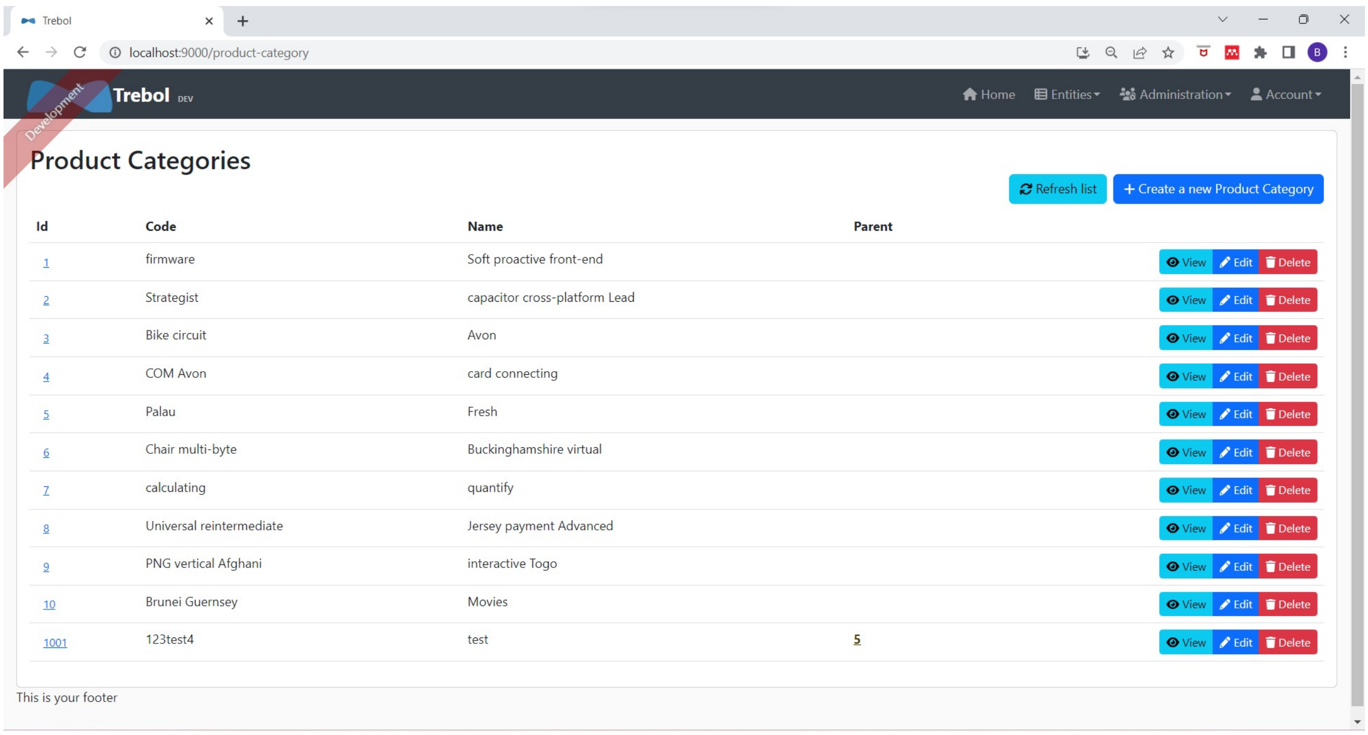
Browser Compatibility refers to the ability of a certain website or app to appear fully functional on different browsers that are available in the market. [5]

To make sure the website is compatible with different browsers, it is needed to follow a systematic process of addressing potential issues and providing a consistent user experience across different browsers. The process begins with utilizing standardized web technologies like HTML, CSS, and JavaScript, to ensure broader browser support.

For cross-browser testing, the browsers Google Chrome, Mozilla Firefox, and Microsoft Edge to ensure wide coverage.

All the browsers mentioned were tested using the Product Category page and all of them had the same behavior.

Google Chrome

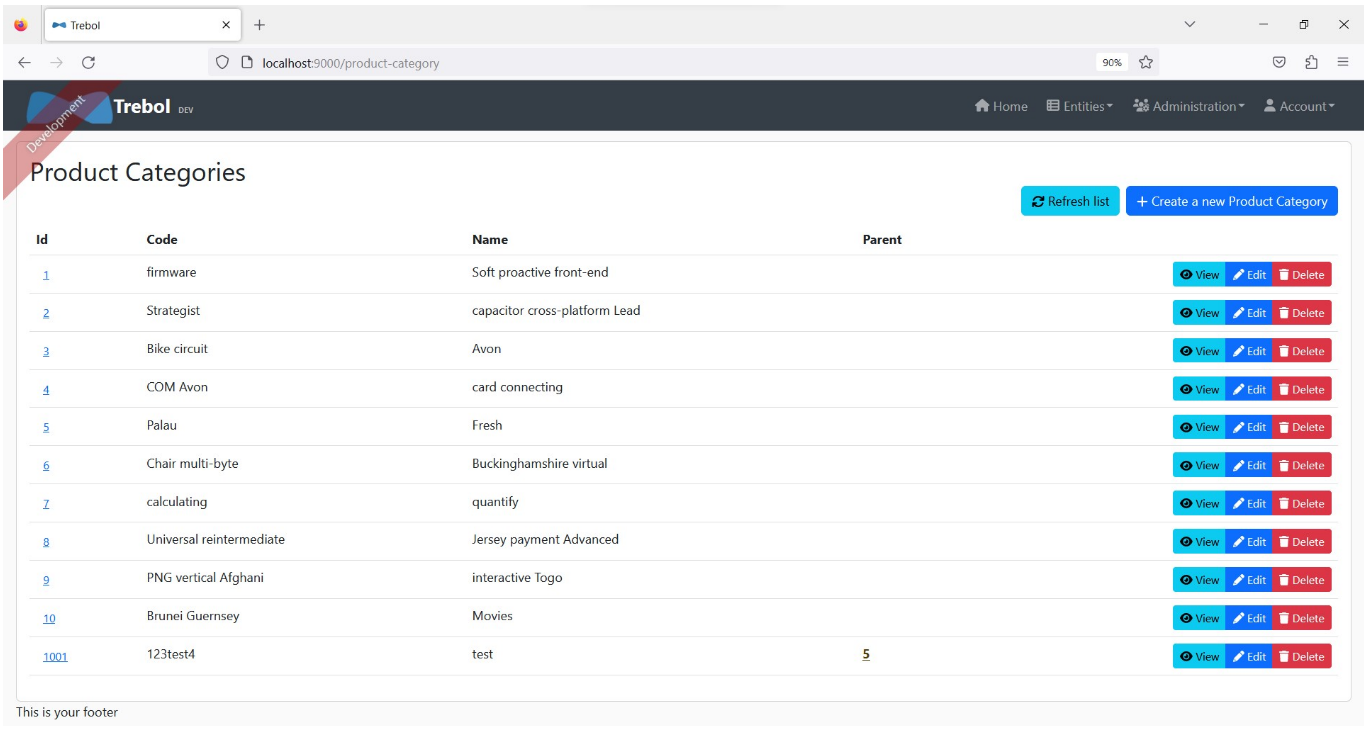


Product Categories

Id	Code	Name	Parent
1	firmware	Soft proactive front-end	
2	Strategist	capacitor cross-platform Lead	
3	Bike circuit	Avon	
4	COM Avon	card connecting	
5	Palau	Fresh	
6	Chair multi-byte	Buckinghamshire virtual	
7	calculating	quantify	
8	Universal reintermediate	Jersey payment Advanced	
9	PNG vertical Afghani	interactive Togo	
10	Brunei Guernsey	Movies	
1001	123test4	test	5

This is your footer

Mozilla Firefox

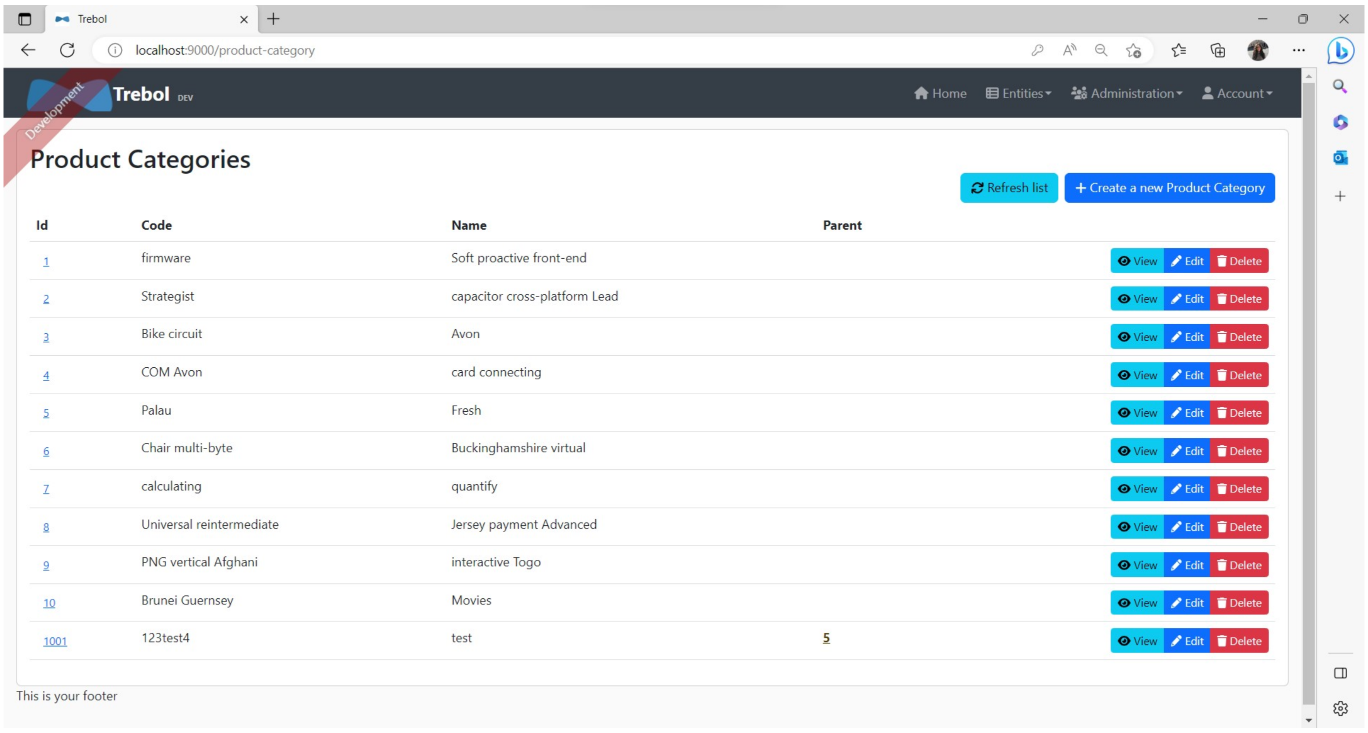


Product Categories

Id	Code	Name	Parent
1	firmware	Soft proactive front-end	
2	Strategist	capacitor cross-platform Lead	
3	Bike circuit	Avon	
4	COM Avon	card connecting	
5	Palau	Fresh	
6	Chair multi-byte	Buckinghamshire virtual	
7	calculating	quantify	
8	Universal reintermediate	Jersey payment Advanced	
9	PNG vertical Afghani	interactive Togo	
10	Brunei Guernsey	Movies	
1001	123test4	test	5

This is your footer

Microsoft Edge



Product Categories

Id	Code	Name	Parent
1	firmware	Soft proactive front-end	
2	Strategist	capacitor cross-platform Lead	
3	Bike circuit	Avon	
4	COM Avon	card connecting	
5	Palau	Fresh	
6	Chair multi-byte	Buckinghamshire virtual	
7	calculating	quantify	
8	Universal reintermediate	Jersey payment Advanced	
9	PNG vertical Afghani	interactive Togo	
10	Brunei Guernsey	Movies	
1001	123test4	test	5

This is your footer

It is possible to see that there are no differences between the three browsers analyzed.

3.2 Device Compatibility

Testing software for device compatibility involves checking its behavior on several devices including mobile phones, tablets, laptops, and desktop systems. Although most of the users use mobile phones for internet access, this part of testing remains of critical importance as you can never anticipate how end-user preferences change. [6]

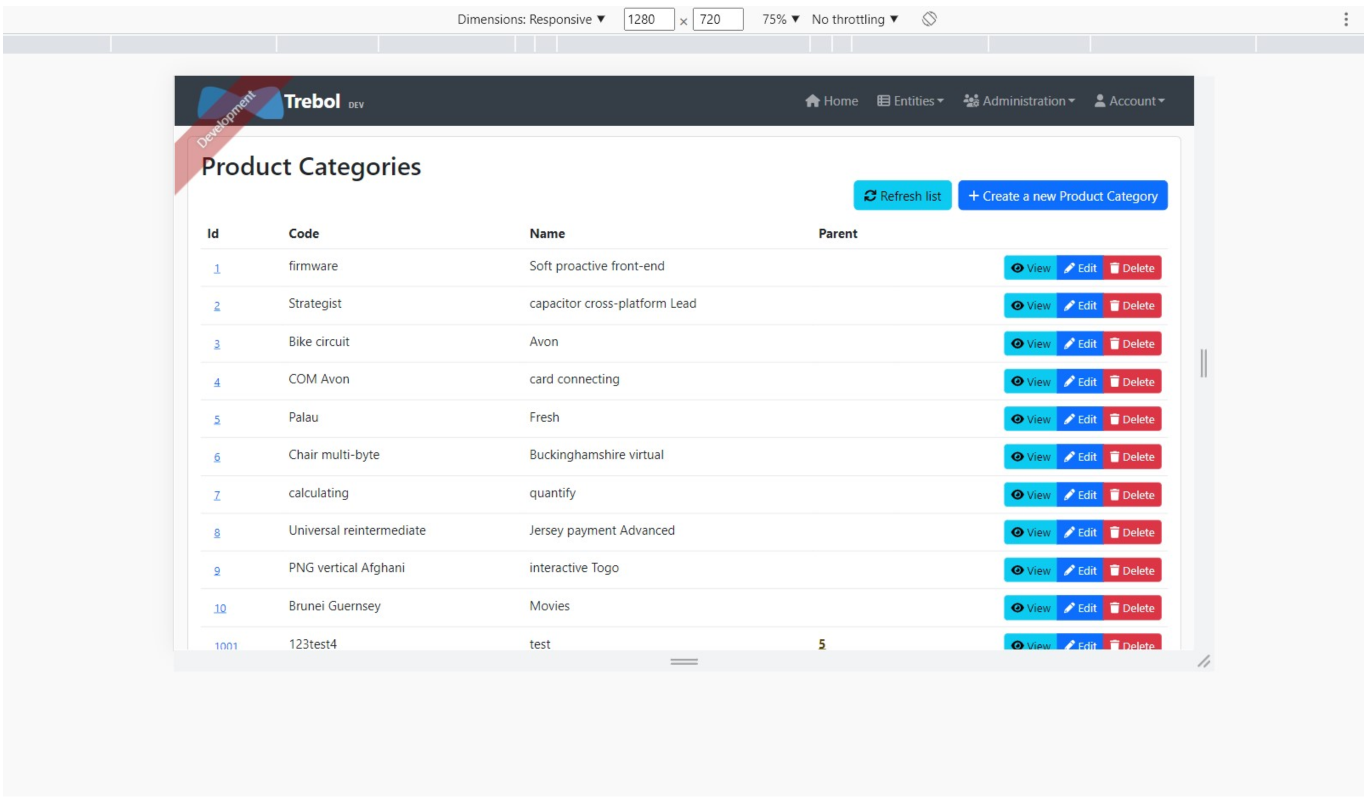
Therefore, to the website, it is necessary to run it on desktop systems, laptops, mobile phones, and tablets.

To test this metric it was used Chrome dev tools to change the dimensions of the page.

The values used were:

- Design for desktop displays from 1280×720 through 1920×1080
- Design for mobile displays from 360×640 through 414×896
- Design for tablet displays from 601×962 through 1280×800

Desktop



Dimensions: Responsive ▼ 1280 × 720 75% ▼ No throttling ▼

Trebol DEV Home Entities Administration Account

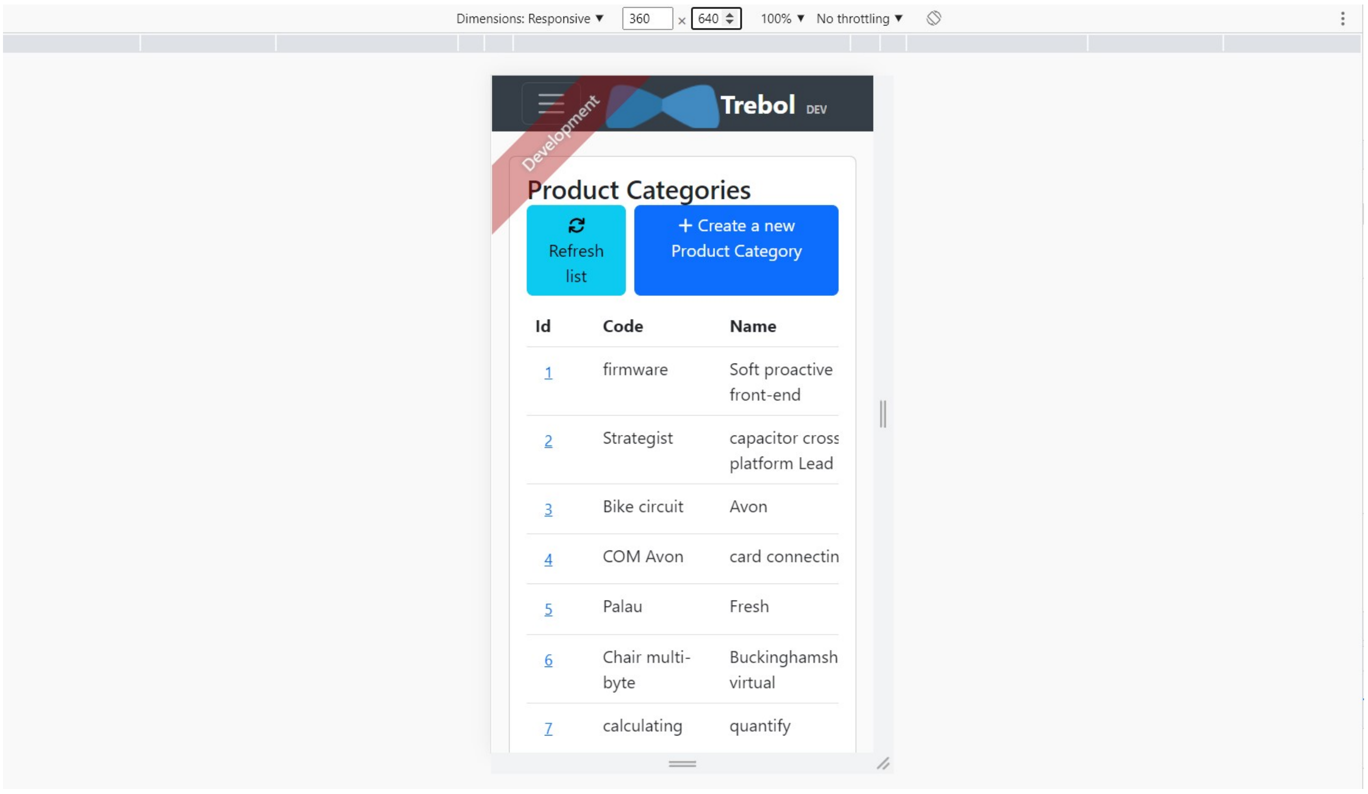
Development

Product Categories

Refresh list | + Create a new Product Category

Id	Code	Name	Parent	
1	firmware	Soft proactive front-end		View Edit Delete
2	Strategist	capacitor cross-platform Lead		View Edit Delete
3	Bike circuit	Avon		View Edit Delete
4	COM Avon	card connecting		View Edit Delete
5	Palau	Fresh		View Edit Delete
6	Chair multi-byte	Buckinghamshire virtual		View Edit Delete
7	calculating	quantify		View Edit Delete
8	Universal reintermediate	Jersey payment Advanced		View Edit Delete
9	PNG vertical Afghani	interactive Togo		View Edit Delete
10	Brunei Guernsey	Movies		View Edit Delete
1001	123test4	test	5	View Edit Delete

Mobile



Dimensions: Responsive ▼ 360 × 640 100% ▼ No throttling ▼

Trebol DEV

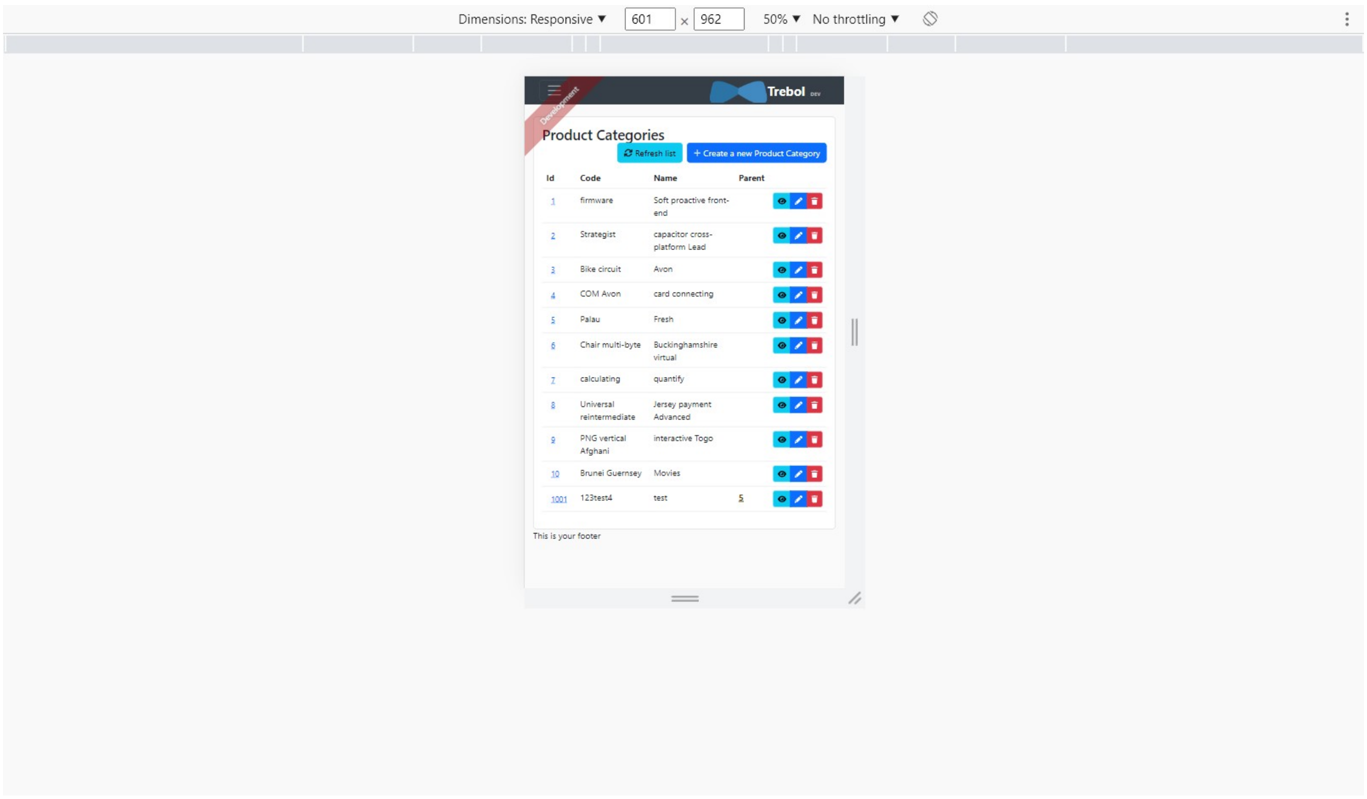
Development

Product Categories

Refresh list | + Create a new Product Category

Id	Code	Name
1	firmware	Soft proactive front-end
2	Strategist	capacitor cross platform Lead
3	Bike circuit	Avon
4	COM Avon	card connectin
5	Palau	Fresh
6	Chair multi-byte	Buckinghamsh virtual
7	calculating	quantify

Tablet



Product Categories

Id	Code	Name	Parent
1	firmware	Soft proactive front-end	
2	Strategist	capacitor cross-platform Lead	
3	Bike circuit	Avon	
4	COM Avon	card connecting	
5	Palau	Fresh	
6	Chair multi-byte	Buckinghamshire virtual	
7	calculating	quantify	
8	Universal reintermediate	Jersey payment Advanced	
9	PNG vertical Afghani	interactive Togo	
10	Brunei Guernsey	Movies	
1001	123test4	test	5

This is your footer

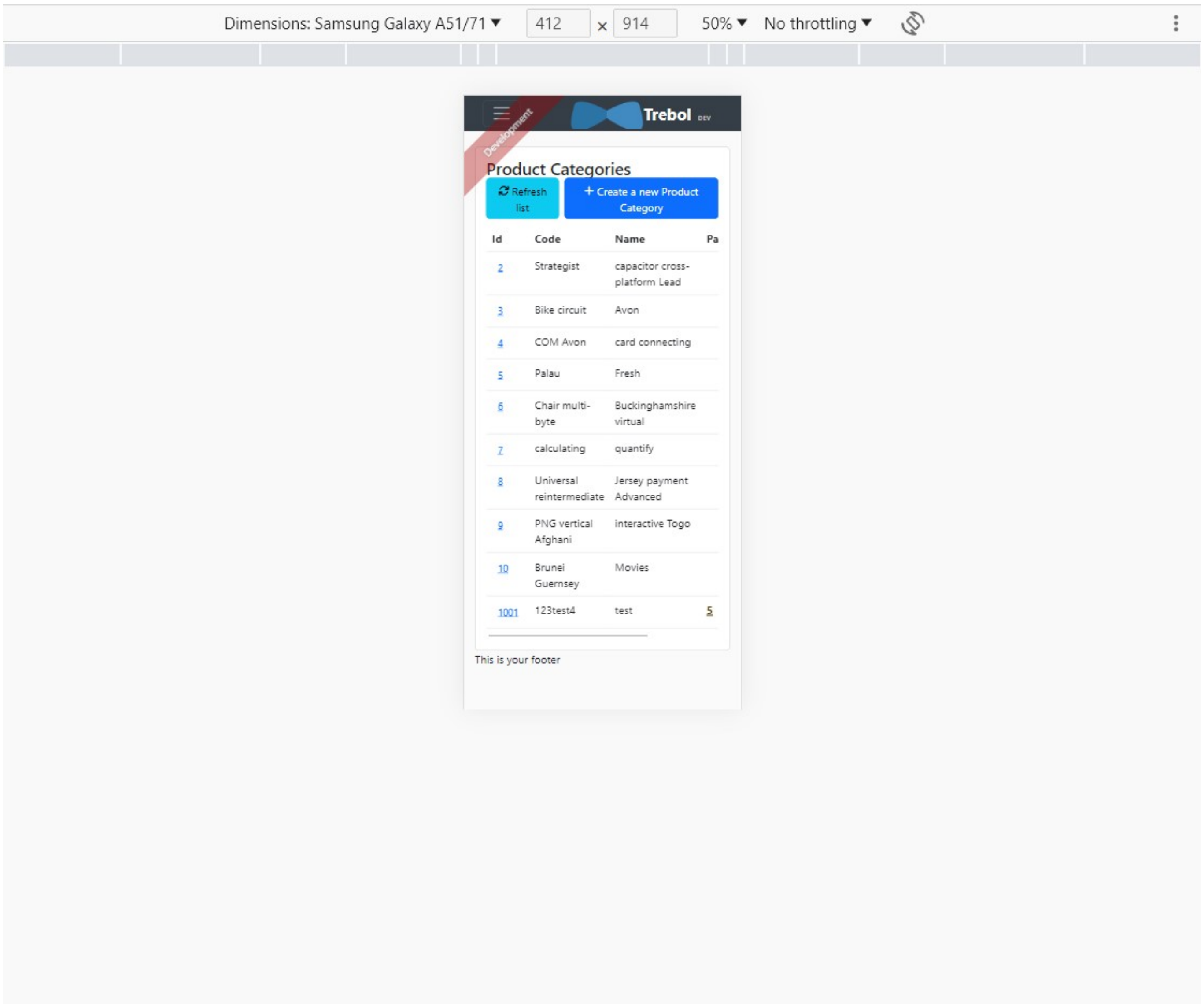
It is possible to see that there are no differences between the three devices analyzed.

3.3 Platform Compatibility

Cross Platform Compatibility testing measures how well pages display on different software and hardware platforms, including different browser versions, different operating systems and different machines.

Platform compatibility testing addresses the difference between operating systems such as an Android and an iOS to ensure consistent and reliable performance. [7]

Android

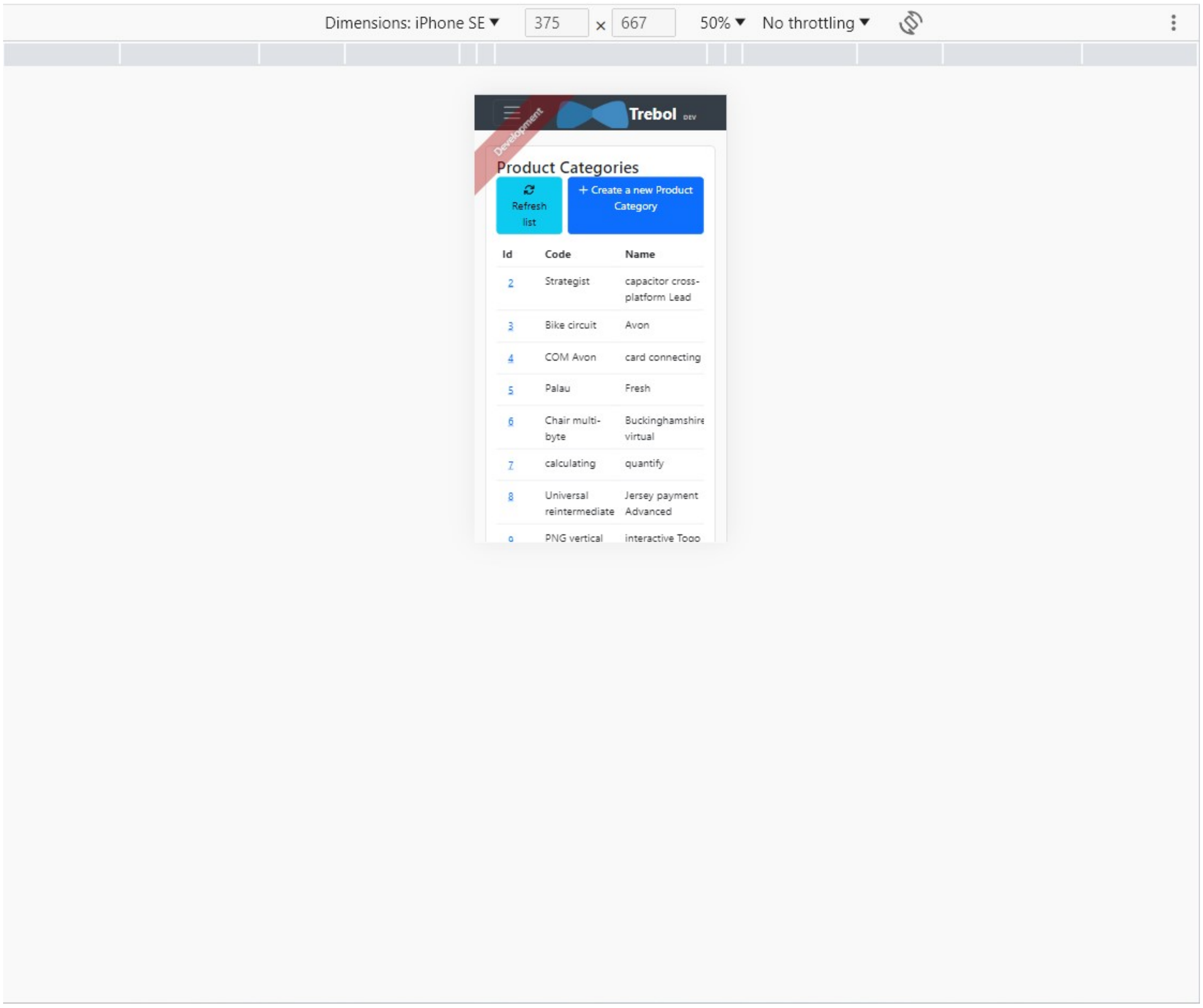

Dimensions: Samsung Galaxy A51/71
412 × 914
50%
No throttling
Development
Trebol DEV
Refresh list
+ Create a new Product Category

Product Categories

Id	Code	Name	Pa
2	Strategist	capacitor cross-platform Lead	
3	Bike circuit	Avon	
4	COM Avon	card connecting	
5	Palau	Fresh	
6	Chair multi-byte	Buckinghamshire virtual	
7	calculating	quantify	
8	Universal reintermediate	Jersey payment Advanced	
9	PNG vertical Afghani	interactive Togo	
10	Brunei Guernsey	Movies	
1001	123test4	test	5

This is your footer

iOS



Product Categories

Refresh list

+ Create a new Product Category

Id	Code	Name
2	Strategist	capacitor cross-platform Lead
3	Bike circuit	Avon
4	COM Avon	card connecting
5	Palau	Fresh
6	Chair multi-byte	Buckinghamshire virtual
7	calculating	quantify
8	Universal reintermediate	Jersey payment Advanced
9	PNG vertical	interactive Togo

It is possible to see that there are no differences between the two platforms analyzed.

4. Maintainability

It was used the IntelliJ plugin MetricsTree and Sonargraph to measure Coupling and Structural Erosion (CSE) and Size and Complexity (SC).

MetricsTree is an IDE extension that helps to evaluate quantitative properties of java code. It supports the most common sets of metrics at the project, package, class, and method levels. [8]

Sonargraph is a powerful static code analyzer that allows you to monitor a software system for technical quality and enforce rules regarding software architecture, metrics and other aspects in all stages of the development process. The Sonargraph platform supports Java, Kotlin (the JVM version of it), C#, Python 3 and C/C++ out of the box and includes powerful features like a Groovy based scripting engine and a DSL (domain specific language) to describe software architecture. [9]

System used in Sonargraph: Trebol-jhispter.sonargraph

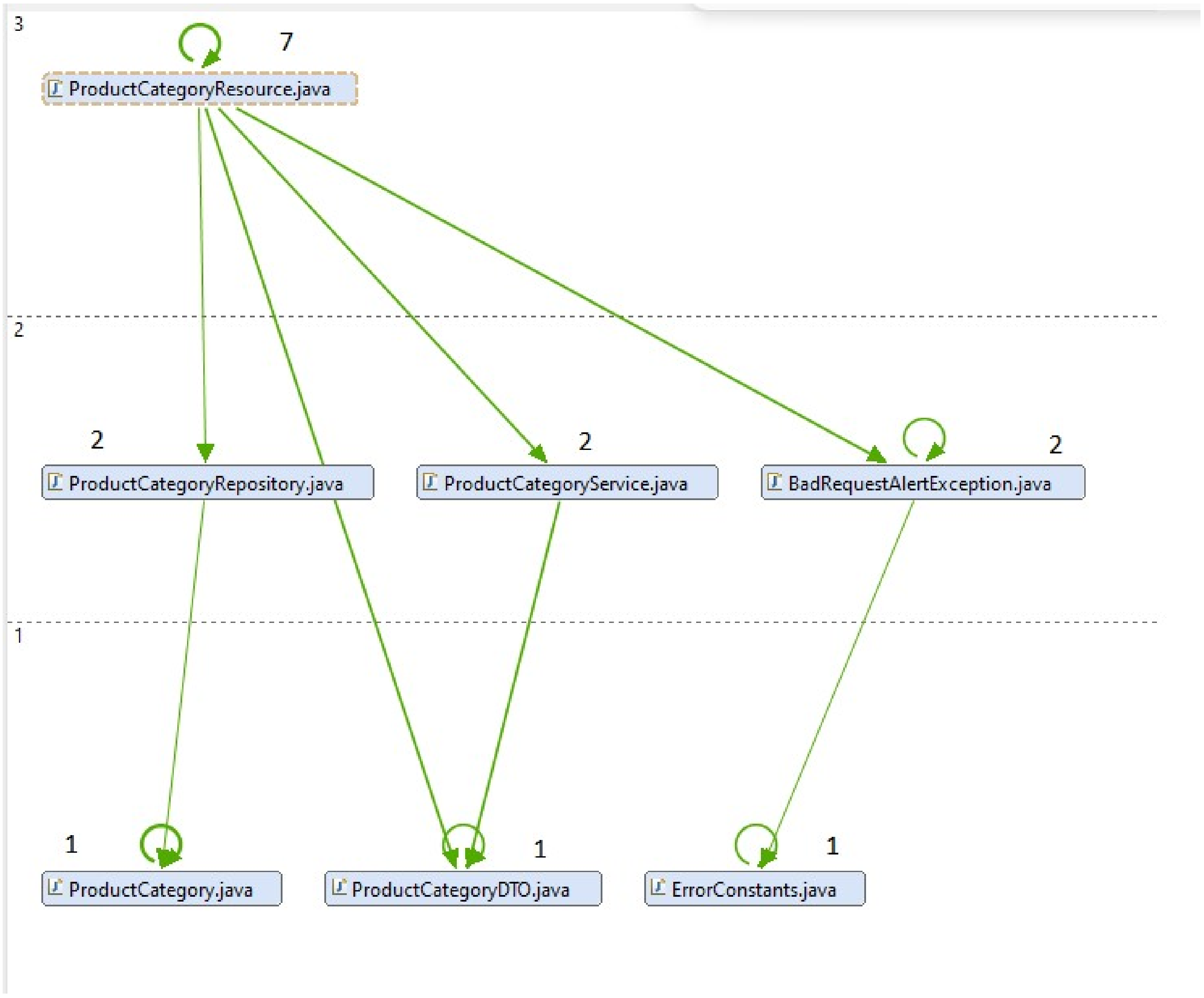
To measure CSE were used the metrics: average component dependency, propagation cost, cyclicity and relative cyclicity and maintainability level.

To measure SC were used the metrics: size metric and cyclomatic complexity.

4.1 Average Component Dependency (ACD)

ACD can be obtained by dividing CCD by the number of components to get the ACD.

This way, it was used the tool Sonargraph to calculate the components to get the ACD and the value of CCD. To do this, in Sonargraph it was selected the class ProductCategoryResource and then selected the option to show in graph view. The graph obtained is in the next image.


3
7
ProductCategoryResource.java
2
2
ProductCategoryRepository.java
2
ProductCategoryService.java
2
BadRequestAlertException.java
1
1
ProductCategory.java
1
ProductCategoryDTO.java
1
ErrorConstants.java

The number above each class represents the value from the metric Depends Upon. This metric was also available in the tool Sonargraph.

Having all the values from the Depends Upon metric of all components of the class ProductCategoryResource, the value CCD can be calculated as the sum of these values. It was determined that CCD has a value of 16.

As it was said before, ACD can be obtained by dividing CCD by the number of components. This way:

```
ACD = 16/7 = 2,28
```

The maximum value is equal to the number of components, in this case it that would be 7 On average, a component depends on 2,28 components. Comparing to the maximum value, it can be said that in terms of coupling, this number is fairly low.

4.2 Propagation Cost (PC)

The metric propagation cost measures the potential impact a change in a component will cause on other components. A higher propagation cost will likely lead to an increase in complexity of the system therefore making it difficult to maintain.

PC can also be calculated by dividing the ACD once more by the number of nodes (components).

```
PC = 2,28/7 = 0,32
```

Since we are working with a small part of the system, this value shouldn't be concerning.

4.3 Maintainability Level

The Maintainability Index first appeared in 1992 when it was proposed by Paul Oman and Jack Hagemeister at the International Conference on Software Maintenance with the goal of establishing automated software development metrics to guide software related decision making. The Maintainability Index tries to give a holistic view of the relative maintenance burden for different sections of a project by blending together a series of different metrics. [10] Here we use a more modern adaptation of the original formula proposed in 2011 by Microsoft:

```
Maintainability Index = MAX(0,(171 - 5.2 * ln(Halstead Volume) - 0.23 * (Cyclomatic Complexity) - 16.2 * ln(Lines of Code))*100 / 171)
```

The ProductCategoryResource class has a Maintainability Index of 32,09

About the methods of this class, their maintainability levels are:

Method	Maintainability Index
ProductCategoryResource()	78,0773
createProductCategory()	55,7516
deleteProductCategory()	61,1356
getAllProductCategory()	68,5218
getProductCategory()	65,166
partialUpdateProductCategory()	49,6253
updateProductCategory()	49,9261

4.4 Size metric

Lines of Code (LoC) per file counts every line that contains actual code and skips empty lines and comment lines.

Total Lines metric counts every single line, including empty lines and comment lines.

Number of Statements verifies the statements, i.e., s a single complete action performed in the source code, usually terminated by a special character or a newline.

In the class ProductCategoryResource it was possible to determine the following values:

- Lines of Code: 108
- Total Lines: 175
- Number of Statements: 40

These values can indicate of how much the component is doing and how complex it is. So, we can say that this component is not complex.

4.5 Cyclomatic complexity

Cyclomatic complexity was proposed in 1976, and it computes the number of different possible execution paths through a method or function, which is also a floor for the number of test cases needed to achieve 100% test coverage.

For this metric it was analyzed the metric Average Complexity that is described as the weighted average modified extended cyclomatic complexity for fully analyzes code.

- Average Complexity: 2,95

As the value calculated is very low we can say that the class ProductCategoryResource is easy to understand and don't have a big risk associated when modifying the class.

4.6 Frontend Coding Issues

In this section, code complexity and unused code will be described.

Code Complexity

Complexity refers to the intricacy and convoluted nature of code. High complexity can make code harder to understand, modify, and maintain. Complex code often contains deeply nested conditionals, excessive loops, and long functions or methods.

To measure code complexity it was used the framework ESLint.

ESLint statically analyzes the code to quickly find problems. It is built into most text editors and can be extended with plugins and rules to measure code complexity. [11]

It was used the plugin eslint-plugin-complexity. This plugin is based on the Cyclomatic Complexity metric.

To install this plugin it was used the following command:

```
npm install eslint-plugin-complexity --save-dev
```

Then, the file .eslintrc.json was updated with the lines:

```
{
  "plugins": [
    "complexity"
  ],
  "rules": {
    "complexity": ["error", 0] // Set the maximum allowed complexity value
  }
}
```

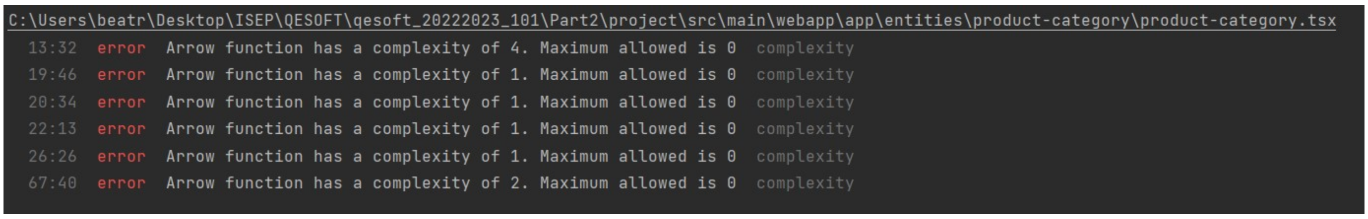
With these lines, the "complexity" rule is enabled, and the maximum allowed complexity is set to 0. This way is possible to observe the complexity for each method.

Finally to analyzed the complexity in product-categories, the next command was executed:

`eslint .\src\main\webapp\app\entities\product-category`

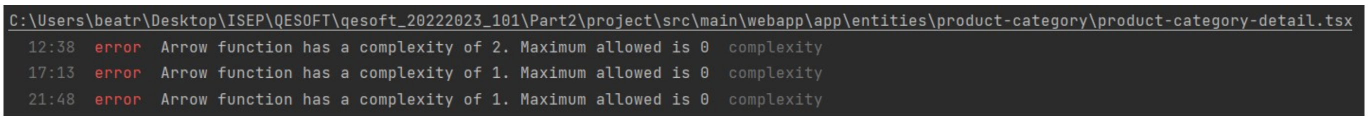
In the console was presented the complexity of each function of each file in the path above.

Example of page list of product categories:



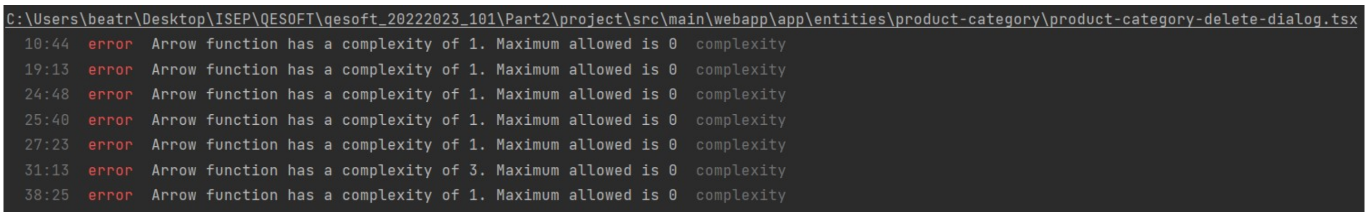
```
C:\Users\beatr\Desktop\ISEP\QESOFT\qesoft_20222023_101\Part2\project\src\main\webapp\app\entities\product-category\product-category.tsx
  13:32  error  Arrow function has a complexity of 4. Maximum allowed is 0  complexity
  19:46  error  Arrow function has a complexity of 1. Maximum allowed is 0  complexity
  20:34  error  Arrow function has a complexity of 1. Maximum allowed is 0  complexity
  22:13  error  Arrow function has a complexity of 1. Maximum allowed is 0  complexity
  26:26  error  Arrow function has a complexity of 1. Maximum allowed is 0  complexity
  67:40  error  Arrow function has a complexity of 2. Maximum allowed is 0  complexity
```

Example of page detail of product categories:



```
C:\Users\beatr\Desktop\ISEP\QESOFT\qesoft_20222023_101\Part2\project\src\main\webapp\app\entities\product-category\product-category-detail.tsx
  12:38  error  Arrow function has a complexity of 2. Maximum allowed is 0  complexity
  17:13  error  Arrow function has a complexity of 1. Maximum allowed is 0  complexity
  21:48  error  Arrow function has a complexity of 1. Maximum allowed is 0  complexity
```

Example of page delete of product categories:



```
C:\Users\beatr\Desktop\ISEP\QESOFT\qesoft_20222023_101\Part2\project\src\main\webapp\app\entities\product-category\product-category-delete-dialog.tsx
  10:44  error  Arrow function has a complexity of 1. Maximum allowed is 0  complexity
  19:13  error  Arrow function has a complexity of 1. Maximum allowed is 0  complexity
  24:48  error  Arrow function has a complexity of 1. Maximum allowed is 0  complexity
  25:40  error  Arrow function has a complexity of 1. Maximum allowed is 0  complexity
  27:23  error  Arrow function has a complexity of 1. Maximum allowed is 0  complexity
  31:13  error  Arrow function has a complexity of 3. Maximum allowed is 0  complexity
  38:25  error  Arrow function has a complexity of 1. Maximum allowed is 0  complexity
```

As it is possible to see the complexity of each function isn't too high. Some functions that have a complexity of 4 and 3 need to improve their complexity.

Unused Code

Unused code is another significant frontend coding issue that can impact maintainability. It refers to pieces of code that are present in the codebase but are not actively used or executed. Having unused code can lead to various problems, such as increased code complexity, decreased performance, and difficulties in code maintenance.

To measure unused code it was used the framework ESLint. It was used the plugin eslint-plugin-unused-imports.

To install this plugin it was used the following command:

`npm install eslint eslint-plugin-unused-imports --save-dev`

Then, the file .eslintrc.json was updated with the lines:

```
{
  "plugins": [
    "unused-imports"
  ],
  "rules": {
    "unused-imports/no-unused-imports": "error",
  }
}
```

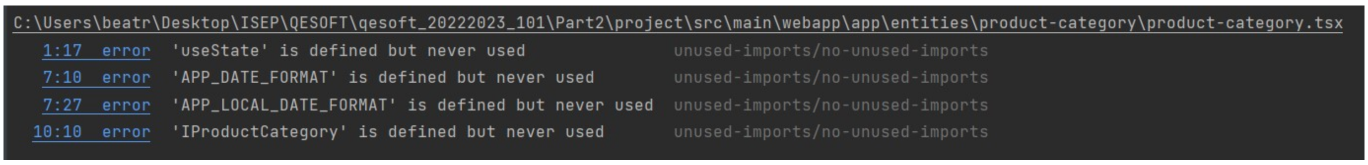
With these lines, the "unused-imports/no-unused-imports" rule is enabled. This way is possible to observe the imports not used in the files.

Finally to analyzed the unused code in product-categories, the next command was executed:

`eslint .\src\main\webapp\app\entities\product-category`

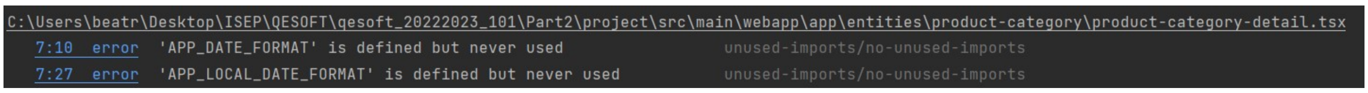
In the console was presented the unused code of each function of each file in the path above.

Example of page list of product categories:



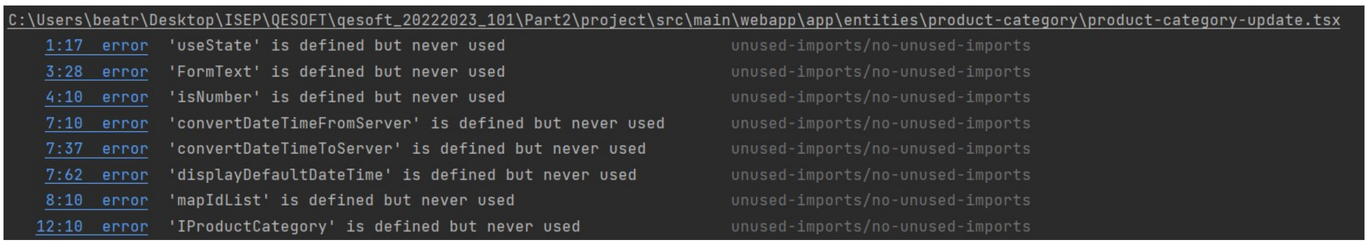
```
C:\Users\beatr\Desktop\ISEP\QESOFT\qesoft_20222023_101\Part2\project\src\main\webapp\app\entities\product-category\product-category.tsx
   1:17  error  'useState' is defined but never used              unused-imports/no-unused-imports
   7:10  error  'APP_DATE_FORMAT' is defined but never used       unused-imports/no-unused-imports
   7:27  error  'APP_LOCAL_DATE_FORMAT' is defined but never used unused-imports/no-unused-imports
  10:10  error  'IProductCategory' is defined but never used      unused-imports/no-unused-imports
```

Example of page detail of product categories:



```
C:\Users\beatr\Desktop\ISEP\QESOFT\qesoft_20222023_101\Part2\project\src\main\webapp\app\entities\product-category\product-category-detail.tsx
  7:10  error  'APP_DATE_FORMAT' is defined but never used               unused-imports/no-unused-imports
  7:27  error  'APP_LOCAL_DATE_FORMAT' is defined but never used         unused-imports/no-unused-imports
```

Example of page update of product categories:



```
C:\Users\beatr\Desktop\ISEP\QESOFT\qesoft_20222023_101\Part2\project\src\main\webapp\app\entities\product-category\product-category-update.tsx
   1:17  error  'useState' is defined but never used                    unused-imports/no-unused-imports
   3:28  error  'FormText' is defined but never used                    unused-imports/no-unused-imports
   4:10  error  'isNumber' is defined but never used                    unused-imports/no-unused-imports
   7:10  error  'convertDateTimeFromServer' is defined but never used   unused-imports/no-unused-imports
   7:37  error  'convertDateTimeToServer' is defined but never used     unused-imports/no-unused-imports
   7:62  error  'displayDefaultDateTime' is defined but never used      unused-imports/no-unused-imports
   8:10  error  'mapIdList' is defined but never used                   unused-imports/no-unused-imports
  12:10  error  'IProductCategory' is defined but never used            unused-imports/no-unused-imports
```

As it is possible to see the unused code of each file isn't too high. All the identified variables can be deleted.

5. Performance

Performance testing is the practice of evaluating how a system performs in terms of responsiveness and stability under a particular workload. Performance tests are typically executed to examine speed, robustness, reliability, and application size.

Performance testing gathers all the tests that verify an application’s speed, robustness, reliability, and correct sizing. It examines several indicators such as a browser, page and network response times, server query processing time, number of acceptable concurrent users architected, CPU memory consumption, and number/type of errors which may be encountered when using an application. [12]

These tests highlight improvements you should make to your applications relative to speed, stability, and scalability before they go into production.

To measure the performance it was used Apache JMeter. Apache JMeter is a pure Java, open-source application that enables load testing and performance measurement of functional behavior. Initially developed for testing web applications, JMeter has evolved to accommodate other types of testing as well.

JMeter can evaluate the performance of both static and dynamic resources, including web-based applications. It is capable of simulating a heavy load on a server, a group of servers, or an object to assess its strength or to analyze overall performance under various load types. [13]

In the GQM presented in the global report were identified the metrics First Contentful Paint, Total Blocking Time, Speed Index, Largest Contentful Paint and Cumulative Layout.

To measure these metrics, it was used Lighthouse. Lighthouse is an open-source, automated tool for improving the quality of web pages. You can run it against any web page, public or requiring authentication. It has audits for performance, accessibility, progressive web apps, SEO, and more. [2]

With Lighthouse the performance audits were checked. The performance audits ensure that your page is optimized for users to be able to see and interact with page content. [14]

Report generated by LightHouse: Lighthouse Report - Performance

5.1 Performe Tests based on User Experience - Lighthouse

The metrics scores and the perf score are colored according to these ranges:

- 0 to 49: Poor
- 50 to 89: Needs Improvement
- 90 to 100: Good

Since the global score was 85, it is considered an acceptable value and it is concluded that the performance needs some improvements.

5.1.1 First Contentful Paint

First Contentful Paint marks the time at which the first text or image is painted.

First Contentful Paint measures how long it takes the browser to render the first piece of DOM content after a user navigates to your page. Images, non-white canvas elements, and SVGs on your page are considered DOM content; anything inside an iframe isn't included. [15]

The First Contentful Paint score is a comparison of the page's First Contentful Paint time and First Contentful Paint times for real websites, based on data from the HTTP Archive.

These scores can be interpreted using the following values:

- 0–1.8: Fast
- 1.8–3: Moderate
- Over 3: Slow

The score obtained from this metric was 0.2s, which is evaluated as fast.

This means that the page takes 0.2s to render the first piece of DOM content after a user navigates to the page associated to the entity Product Category.

5.1.2 Total Blocking Time

Total Blocking Time expresses the sum of all time periods between First Contentful Paint and Time to Interactive, when task length exceeded 50ms, expressed in milliseconds.

Total Blocking Time measures the total amount of time that a page is blocked from responding to user input, such as mouse clicks, screen taps, or keyboard presses. The sum is calculated by adding the blocking portion of all long tasks between First Contentful Paint and Time to Interactive. Any task that executes for more than 50 ms is a long task. The amount of time after 50 ms is the blocking portion. [16]

Total Blocking Time score is a comparison of the page's Total Blocking Time time and Total Blocking Time times millions of real sites when loaded on mobile devices.

These scores can be interpreted using the following values:

- 0–200: Fast
- 200–600: Moderate
- Over 600: Slow

The score obtained from this metric was 220ms, which is evaluated as moderate.

This means that the page associated to the entity Product Category is blocked by 220ms from responding to the user input.

5.1.3 Speed Index

Speed Index shows how quickly the contents of a page are visibly populated.

Speed Index measures how quickly content is visually displayed during page load. Lighthouse first captures a video of the page loading in the browser and computes the visual progression between frames. [17]

Speed Index score is a comparison of the page's speed index and the speed indices of real websites, based on data from the HTTP Archive.

These scores can be interpreted using the following values:

- 0–3.4: Fast
- 3.4-5.8: Moderate
- Over 5.8: Slow

The score obtained from this metric was 1.4s, which is evaluated as fast.

This means that the page associated to the entity Product Category takes 1.4s to visually display the content during page load.

5.1.4 Largest Contentful Paint

Largest Contentful Paint marks the time at which the largest text or image is painted.

Largest Contentful Paint measures when the largest content element in the viewport is rendered to the screen. This approximates when the main content of the page is visible to users. [18]

Largest Contentful Paint scores are extracted from the data from Chrome's tracing tool.

These scores can be interpreted using the following values:

- 0–2.5: Fast
- 2.5-4: Moderate
- Over 4: Slow

The score obtained from this metric was 1.6s, which is evaluated as fast.

This means that the page associated to the entity Product Category takes 1.6s to make the main content of the page visible to users.

5.1.5 Cumulative Layout Shift

Cumulative Layout Shift measures the movement of visible elements within the viewport.

Cumulative Layout Shift is a measure of the largest burst of layout shift scores for every unexpected layout shift that occurs during the entire lifespan of a page. A layout shift occurs any time a visible element changes its position from one rendered frame to the next. [19]

The scores of Cumulative Layout Shift can be interpreted using the following values:

- 0–0.1: Good
- 0.1-0.25: Needs Improvement
- Over 0.25: Poor

The score obtained from this metric was 0.065, which is evaluated as good.

To provide a good user experience, sites should strive to have a Cumulative Layout Shift score of 0.1 or less. This is verified in the page associated to the entity Product Category.

5.2 Backend Performance Tests - Apache JMeter

There are three main types of performance tests commonly used:

- **Load Tests:** These tests are designed to measure the performance of a system under high levels of user traffic or load. They simulate a large number of users or transactions to determine how the system responds.
- **Stress Tests:** Stress testing is similar to load testing but pushes the system beyond normal usage scenarios to test its limits. This involves testing how the system performs under extreme conditions.
- **Soak Tests:** Soak testing is used to evaluate how a system performs over an extended period of time. This type of testing involves subjecting the system to sustained load for several hours, days, or even weeks to observe its performance. Soak tests are especially useful for detecting memory leaks.

To run JMeter tests, a test plan was designed with the following configuration:

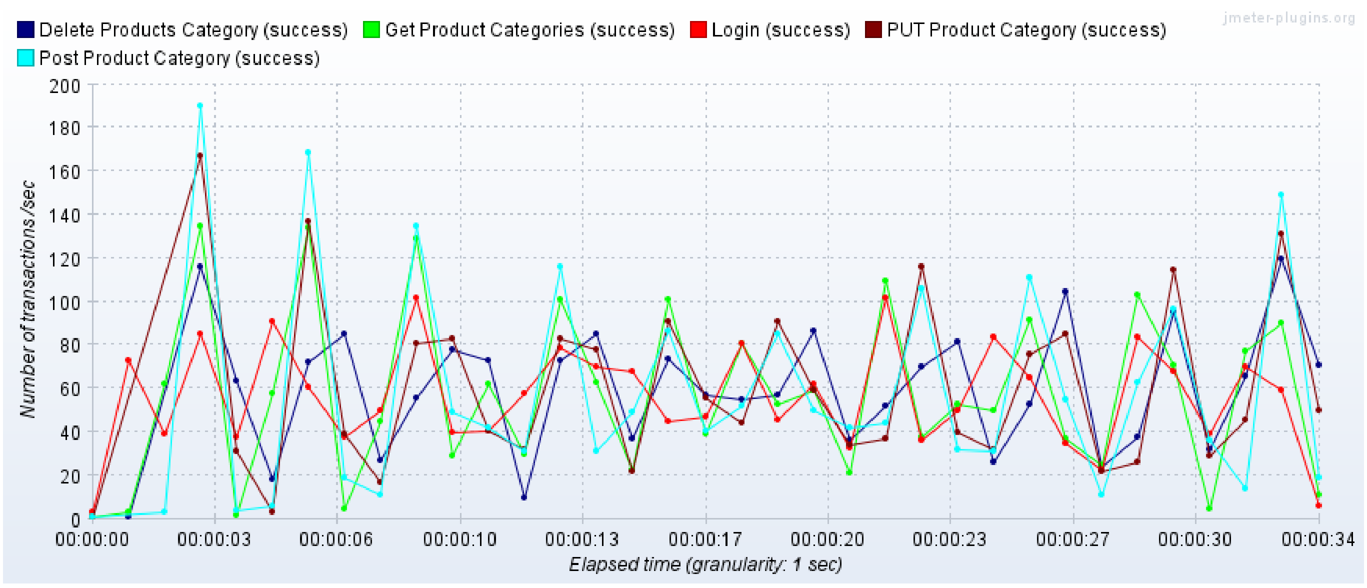
```
.
└── Thread Group/
    ├── Http Request (Login)/
    │   ├── Http Header Manager
    │   ├── View Results Tree
    │   └── JSR223 PostProcessor
    ├── Http Request (Get Product Categories)/
    │   ├── Http Header Manager
    │   └── View Results Tree
    ├── Http Request (Post Product Category)/
    │   ├── Http Header Manager
    │   ├── View Results Tree
    │   ├── CSV Data Set Config
    │   └── JSR223 PostProcessor
    ├── Http Request (Put Product Category)/
    │   ├── Http Header Manager
    │   └── View Results Tree
    ├── Http Request (Delete Product Category)/
    │   ├── Http Header Manager
    │   └── View Results Tree
    ├── View Results Tree
    ├── jp@gc - Transactions per Second
    └── Aggregate Report
```

5.2.1 Load Test

To do a load test for the application, we'll simulate many users accessing the product categories page. The thread properties defined will be:

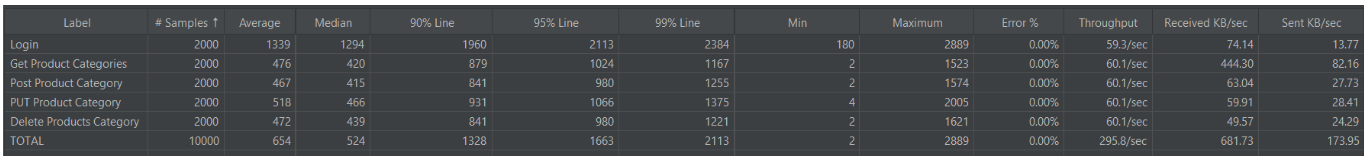
- **Number of Threads (users):** 200
- **Ramp-up period:** 1
- **Loop Count:** 10

Only 1 thread was defined for each test, so it'll only take 1 second to ramp it up. To see the results, a View Results Tree Listener and a Aggregate Report was used. It was also used a jp@gc - Transactions per Second. These were possible after the download of these plugins.



By this image (jp@gc - Transactions per Second), it is possible to see that in the same second several requests are being handled. Besides this, we can also see the that all requests were a success.

To show the results from the test made, we used an Aggregate Report :



Label	# Samples ↑	Average	Median	90% Line	95% Line	99% Line	Min	Maximum	Error %	Throughput	Received KB/sec	Sent KB/sec
Login	2000	1339	1294	1960	2113	2384	180	2889	0.00%	59.3/sec	74.14	13.77
Get Product Categories	2000	476	420	879	1024	1167	2	1523	0.00%	60.1/sec	444.30	82.16
Post Product Category	2000	467	415	841	980	1255	2	1574	0.00%	60.1/sec	63.04	27.73
PUT Product Category	2000	518	466	931	1066	1375	4	2005	0.00%	60.1/sec	59.91	28.41
Delete Products Category	2000	472	439	841	980	1221	2	1621	0.00%	60.1/sec	49.57	24.29
TOTAL	10000	654	524	1328	1663	2113	2	2889	0.00%	295.8/sec	681.73	173.95

Based on these results, it is concluded:

- **Average Response Time:** 654ms. This means that the application is responding quickly.
- **Median Response Time:** 524ms. Most of the requests are being processed in a short time.
- **Throughput:** 295.8 transactions per second. This value is not very high since we are simulating the test with 200 users.
- **Errors:** There were no errors recorded during the test.

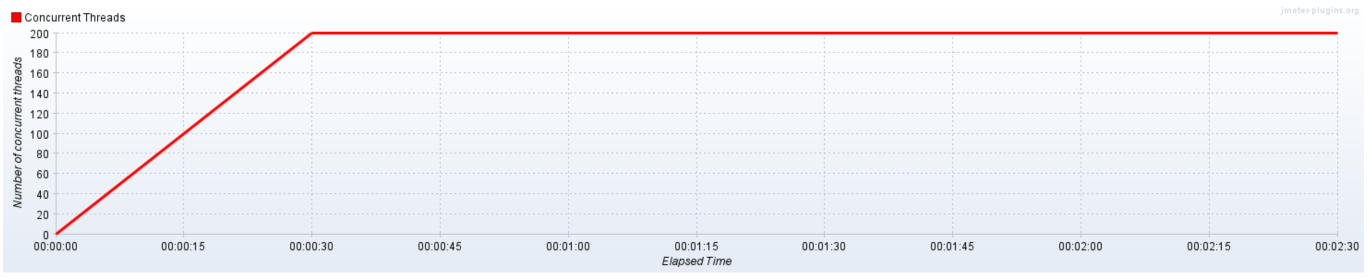
5.2.2 Stress Test

For this type of test, it was used the bzm - concurrency thread group plugin.

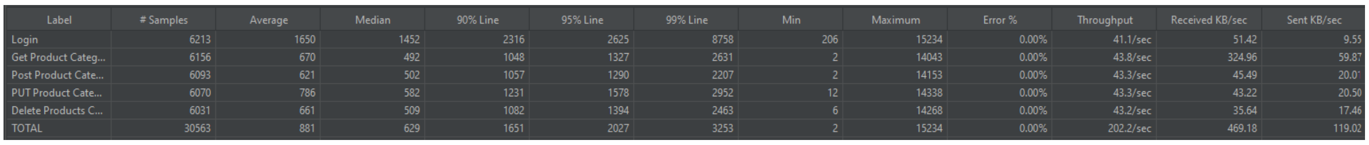
The thread properties defined will be:

- **Target Concurrency:** 200
- **Ramp-Up time (min):** 0.5
- **Ramp-Up Steps Count:**
- **Hold Target Rate Time (min):** 2

Having this configurations as well as the next image is possible to conclude that the 200 users will start after a 30 second period. There are no ramp-up steps, this means that the concurrency level will never increase. This test will last for 2 minutes.



To show the results from the test made, we used an Aggregate Report :



Label	# Samples	Average	Median	90% Line	95% Line	99% Line	Min	Maximum	Error %	Throughput	Received KB/sec	Sent KB/sec
Login	6213	1650	1452	2316	2625	8758	206	15234	0.00%	41.1/sec	51.42	9.55
Get Product Categ...	6156	670	492	1048	1327	2631	2	14043	0.00%	43.8/sec	324.96	59.87
Post Product Cate...	6093	621	502	1057	1290	2207	2	14153	0.00%	43.3/sec	45.49	20.0
PUT Product Cate...	6070	786	582	1231	1578	2952	12	14338	0.00%	43.3/sec	43.22	20.50
Delete Products C...	6031	661	509	1082	1394	2463	6	14268	0.00%	43.2/sec	35.64	17.46
TOTAL	30563	881	629	1651	2027	3253	2	15234	0.00%	202.2/sec	469.18	119.02

Based on these results, it is concluded:

- **Average Response Time:** 881ms. This means that the application is responding quickly.
- **Median Response Time:** 629ms. Most of the requests are being processed in a short time.
- **Throughput:** 202.2 transactions per second. This value is not very high since we are simulating the test with 200 users.
- **Errors:** There were no errors recorded during the test.

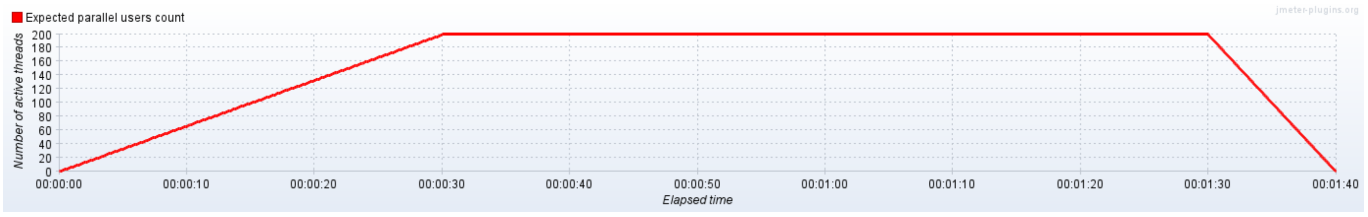
5.2.3 Soak Test

For this type of test, it was used the jp@gc - ultimate thread group plugin.

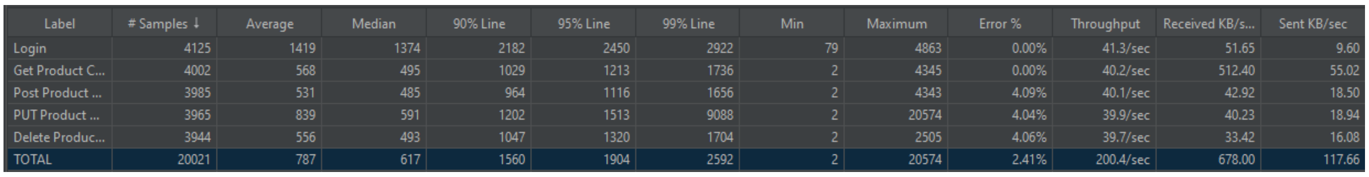
The thread properties defined will be:

- **Start Threads Count:** 200
- **Initial Delay (sec):** 0
- **Start Time (sec):** 30
- **Hold Load For (sec):** 60
- **Shutdown Time:** 10

Having this configurations as well as the next image is possible to conclude that the 200 users will start immediately. the thread groups will start after a delay of 30 seconds from the beginning of the test and the load will be sustained for 60 seconds. A 10 second shutdown period will be allocated to stop the threads and conclude the test.



To show the results from the test made, we used an Aggregate Report :



Label	# Samples ↓	Average	Median	90% Line	95% Line	99% Line	Min	Maximum	Error %	Throughput	Received KB/s...	Sent KB/sec
Login	4125	1419	1374	2182	2450	2922	79	4863	0.00%	41.3/sec	51.65	9.60
Get Product C...	4002	568	495	1029	1213	1736	2	4345	0.00%	40.2/sec	512.40	55.02
Post Product ...	3985	531	485	964	1116	1656	2	4343	4.09%	40.1/sec	42.92	18.50
PUT Product ...	3965	839	591	1202	1513	9088	2	20574	4.04%	39.9/sec	40.23	18.94
Delete Produc...	3944	556	493	1047	1320	1704	2	2505	4.06%	39.7/sec	33.42	16.08
TOTAL	20021	787	617	1560	1904	2592	2	20574	2.41%	200.4/sec	678.00	117.66

Based on these results, it is concluded:

- **Average Response Time:** 787ms. This means that the application is responding quickly.
- **Median Response Time:** 617ms. Most of the requests are being processed in a short time.
- **Throughput:** 200.4 transactions per second. This value is not very high since we are simulating the test with 200 users.
- **Errors:** There were 2.41% errors recorded during the test. These errors occurred in the requests POST, PUT and DELETE.

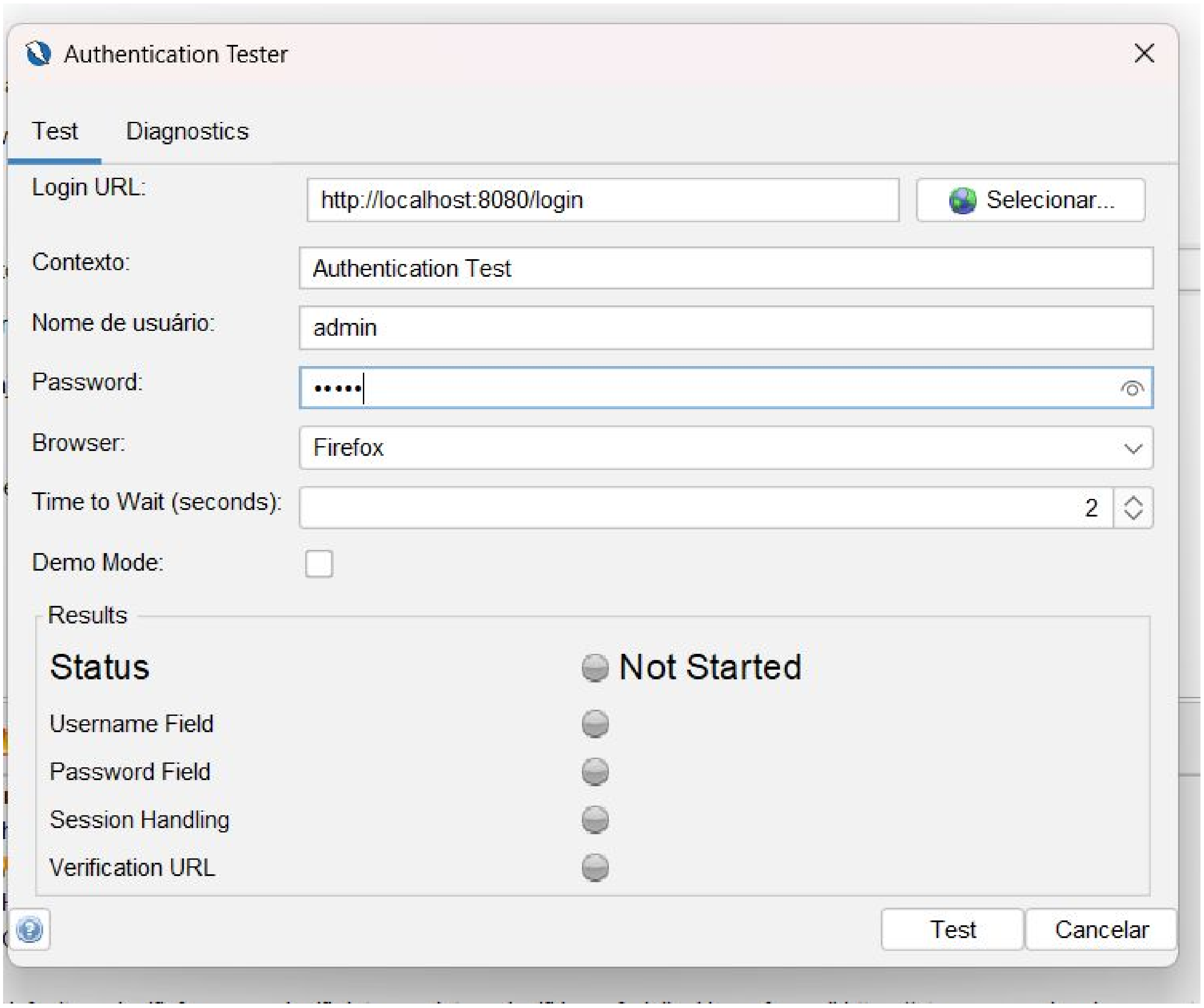
6. Security

Security testing determines whether software is susceptible to online assaults and evaluates the effects of unauthorized or unanticipated inputs on its functionality. Security testing gives proof that data and systems are secure, trustworthy, and do not allow unauthorized inputs. [20]

To measure security it was used the tool OWASP ZAP.

Zed Attack Proxy (ZAP) is a free, open-source penetration testing tool being maintained under the umbrella of the Open Web Application Security Project (OWASP). ZAP is designed specifically for testing web applications and is both flexible and extensible. [21]

In order to obtain a report from OWASP ZAP, in this tool, a add-on was installed (Authentication Helper). Then, is was executed the following authentication tester:



Authentication Tester

Test Diagnostics

Login URL: http://localhost:8080/login

Contexto: Authentication Test

Nome de usuário: admin

Password: •••••

Browser: Firefox

Time to Wait (seconds): 2

Demo Mode: ☐

Results

Status	● Not Started
Username Field	●
Password Field	●
Session Handling	●
Verification URL	●

However most of the requests failed due to authentication issues and all the requests made to http://localhost:8080/api/product-categories had the response code 401.

The reports generated:

- OWASP ZAP Report
- OWASP ZAP Risk and Confidence Report

Taken into account the OWASP Top 10 - 2021, in this section will be mentioned A04:2021 – Insecure Design and A10:2021 – Server-Side Request Forgery (SSRF).

6.1 A04:2021 – Insecure Design

As it is described in [22], this is new category for 2021. It focuses on risks related to design and architectural flaws, with a call for more use of threat modeling, secure design patterns, and reference architectures. As a community we need to move beyond "shift-left" in the coding space to pre-code activities that are critical for the principles of Secure by Design. Notable Common Weakness Enumerations (CWEs) include CWE-209: Generation of Error Message Containing Sensitive Information, CWE-256: Unprotected Storage of Credentials, CWE-501: Trust Boundary Violation, and CWE-522: Insufficiently Protected Credentials.

6.1.1 Description

A secure design can be described as a culture and methodology that constantly evaluates threats and ensures that code is robustly designed and tested to prevent known attack methods. Threat modeling should be integrated into refinement sessions (or similar activities). In the development of the user story it is recommended the determination of the correct flow and failure states, in a way that they are well understood by everyone involved. The analysis of assumptions and conditions for expected and failure flows is also recommended.

6.1.2 Solutions

There are several ways to prevent these kinds of errors, such as:

- With AppSec professionals, establish and utilize a secure development lifecycle to assist in the evaluation and design of security and privacy-related controls.
- Create and make use of a library of safe design patterns or ready-to-use components for paved roads.
- Critical authentication, access control, business logic, and key flows can all benefit from threat modeling.
- Integrate security language and controls into user stories
- Coordinate security language and controls into client stories
- Make sure that all of the critical flows are resistant to the threat model by writing unit and integration tests. For each tier of your application, compile misuse and use cases.
- Isolate level layers on the framework and organization layers relying upon the openness and assurance needs
- Design strong tenant segmentation across all tiers
- Limit asset utilization by client or administration

6.2 A10:2021 – Server-Side Request Forgery (SSRF)

As it is described in [23], this classification is added from the Top 10 community study. The data show that the incidence rate is relatively low, that testing coverage is above average, and that the potential for exploit and impact is above average. The hope is that new entries, which are likely to be a single or small group of Common Weakness Enumerations (CWEs) for attention and awareness, can be rolled into a larger category in a subsequent edition.

6.2.1 Description

When a web application fetches a remote resource without validating the user-supplied URL, SSRF flaws occur. It permits an aggressor to force the application to send a created solicitation to an unforeseen objective, in any event, when safeguarded by a firewall, VPN, or one more kind of organization access control list. Fetching a URL is becoming more and more common in modern web applications, which offer users a variety of convenient features

6.2.2 Solutions

Developers can implement any or all of the following defense in depth controls to stop SSRF:

- From Network layer
    - To lessen the impact of SSRF, separate networks should be used for remote resource access functionality.

- Deny by default firewall policies or network access control rules should be used to block all but essential intranet traffic.

- From Application layer:
  - Use a positive allow list to enforce the URL schema, port, and destination.
  - Do not send clients raw responses.
  - Disable HTTP redirections.
  - Be aware of URL consistency to avoid attacks like DNS rebinding and "time of check, time of use" (TOCTOU) race conditions.
  - Sanitize and validate all client-supplied input data.

7. Conclusions

It is possible to draw the conclusion that the application cannot be used as the foundation for a new project based on the previous analysis.

In what concerns accessibility, the page Product Categories comparing to WCAG 2.1 it had the grade A which is the lowest and easiest level of conformance to obtain. This means that it does not achieve broad accessibility for many situations.

This application is compatible with different browsers, devices, and platforms.

The project has a low level of maintainability (32.09).

In terms of performance, all the tests gave good results.

In terms of security it was not possible to obtain results due to some problems with the tool used by the group.

Concerning investigation, we can presume that this application can't be utilized.

8. **References**

[1] Accessibility - Developer Mozilla. (https://developer.mozilla.org/en-US/docs/Learn/Accessibility/What_is_accessibility#so_what_is_accessibility)

[2] Lighthouse. (https://developer.chrome.com/docs/lighthouse/overview/)

[3] Lighthouse - Accessibility Audits. (https://developer.chrome.com/docs/lighthouse/accessibility/)

[4] WAVE Evaluation Tool. (https://chrome.google.com/webstore/detail/wave-evaluation-tool/jbbplnpkjmmeebjpijfedlgcdilocofh/related)

[5] Browser Compability. (https://www.omniconvert.com/what-is/browser-compatibility/)

[6] Devices Compability. (https://diffy.website/blog/compatibility-testing)

[7] Platform Compability. (https://xbosoft.com/software-testing-services/platform-compatibility/)

[8] MetricsTree plugin for IntelliJ IDEA. (https://plugins.jetbrains.com/plugin/13959-metricstree)

[9] Sonargraph. (https://www.hello2morrow.com/products/sonargraph)

[10] Coleman, D., Ash, D., Lowther, B., & Oman, P. (1994, August). Using metrics to evaluate software system maintainability. Computer, 27(8), 44-49. DOI: 10.1109/2.291294

[11] ESLint. (https://eslint.org/)

[12] Performance Testing. (https://www.tricentis.com/learn/performance-testing)

[13] Apache Software Foundation - Apache JMeter. (https://jmeter.apache.org/)

[14] Lighthouse - Performance Audits. (https://developer.chrome.com/docs/lighthouse/performance/)

[15] Lighthouse - First Contentful Paint. (https://developer.chrome.com/docs/lighthouse/performance/first-contentful-paint/?utm_source=lighthouse&utm_medium=devtools)

[16] Lighthouse - Total Blocking Time. (https://developer.chrome.com/docs/lighthouse/performance/lighthouse-total-blocking-time/?utm_source=lighthouse&utm_medium=devtools)

[17] Lighthouse - Speed Index. (https://developer.chrome.com/docs/lighthouse/performance/speed-index/?utm_source=lighthouse&utm_medium=devtools)

[18] Lighthouse - Largest Contentful Paint. (https://developer.chrome.com/docs/lighthouse/performance/lighthouse-largest-contentful-paint/?utm_source=lighthouse&utm_medium=devtools)

[19] Lighthouse - Cumulative Layout Shift. (https://web.dev/i18n/en/cls/)

[20] Security Testing. (https://brightsec.com/blog/security-testing/)

[21] OWASP ZAP. (https://www.zaproxy.org/getting-started/)

[22] OWASP ZAP - A04:2021 – Insecure Design. (https://owasp.org/Top10/A04_2021-Insecure_Design/)

[23] OWASP ZAP - A10:2021 - Server-Side Request Forgery. (https://owasp.org/Top10/A10_2021-Server-Side_Request_Forgery_%28SSRF%29/)