Test Plan

Test Strategy

The tests are categorized into sections according to the application's components. The tests will be structured in the Page Object model.

My approach is to implement inheritance relations if needed and to avoid duplication of code. Furthermore, I decided to keep layout and UI tests out of scope for now.

Tests and Tests Description

For each page in the application, test:

1. The header contains 3 cells with expected text displayed.
2. The table contains 10 rows. each row with an increasing value according to the number of the page (for page 1, numbers are 1-10).
3. For each cell in the table's body:
   a. The cell is not empty
   b. The cell contains valid values (no number or special characters)
4. For each member (name + surname in a row):
   a. There is no member with a name equal to surname.
   b. There are no duplicate members and duplicate Ids.
5. The footer contains the number count of the page.
6. Clicking on the right/left arrow leads to the next page.
7. Clicking on the right/left arrow when on the pages 1/10 does nothing.
8. Entering the number of the page by typing leads to the expected page.
9. Typing an invalid input does nothing/leads to the first page - verify with Product the expected behavior.
10. Verify the loading time of each page is less or equal 1-2 secs.

Not part of the iteration over the pages:

Add a member tests:
1. Verify the header.
2. Verify putting empty or invalid values is not allowed.
3. Add a valid member.

Server side testing:
Send a GET request to "**/pages?pages=1" and verify a json is returned with 10 first members.