Front-End Test

SUMMARY

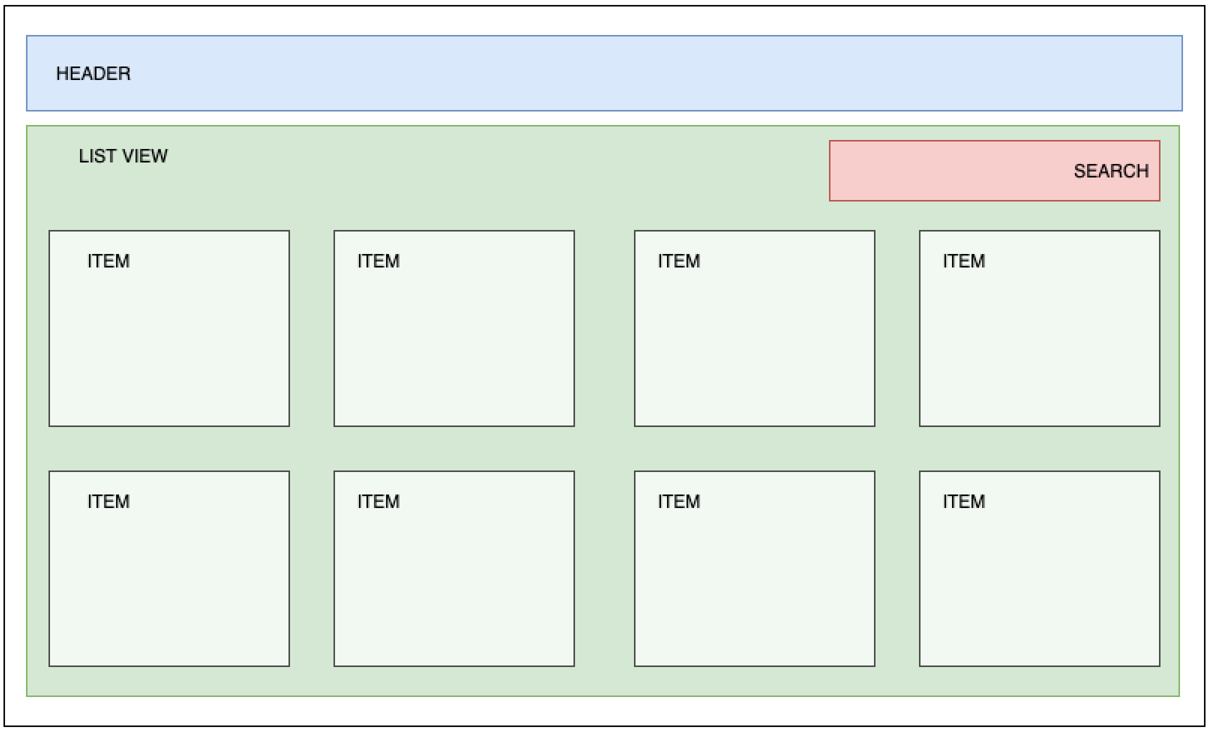
This test consists of the creation of a mini-application for purchasing mobile devices.

- The application will have only two views:
    1. Main view - List of products
    2. Product details
- The implementation of the designs is free choice, but must follow the structure defined in the screenshots. The level of detail of the proposal will be positively valued.
- It requires the use of React/Preact for application development and can be complemented with other JS libraries if deemed appropriate.
- The use of JS with ES6 is allowed, and preferably not testing with Typescript.
- A boilerplate template can be used to create the project structure.
- The application will be a SPA, where the routing of the views will be added to the client code, without being an MPA or using SSR.
- The project will have to contain the following scripts, in order to manage the application:
    1. START - Development mode
    2. BUILD - Compilation for Production mode
    3. TEST - Test launch
    4. LINT - Code check
- The project should be presented in an open source repository (Github, Gitlab, Bitbucket), with the solution to the problem. It is desired that the code can be uploaded in an evolutionary way in order to reach milestones.
- A README document must be included in the repository (preferably included in the first commit), where the explanation to execute the project will be included as well as any explanatory note or additional information considered necessary.

DESCRIPTION OF THE VIEWS

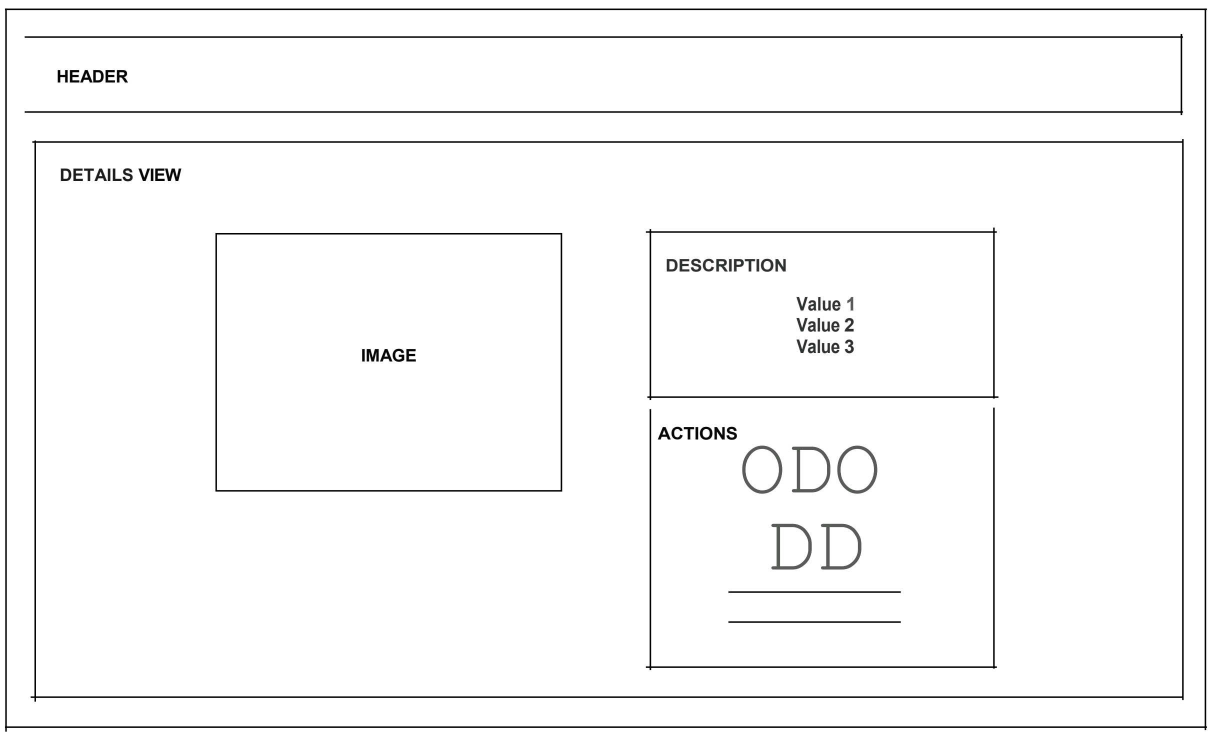
PLP - Product List Page

- Page where the list of products will be displayed.
- This page will show all the elements returned by the API request.
- It will allow content filtering based on the search criteria entered by the user.
- When you select a product, you must navigate to the product details.
- A maximum of four elements per row will be displayed, and it will be adaptive according to the resolution.



PDP - Product Details Page

- This page will be divided into two columns:
    - In the first one, the product image component will be displayed.
    - In the second one, the product details and actions will be shown.
- It should display a link to navigate back to the product list.


HEADER
DETAILS VIEW
IMAGE
DESCRIPTION
Value 1
Value 2
Value 3
ACTIONS
ODO
DD

DESCRIPTION OF THE COMPONENTS

HEADER

- The title or icon of the application will act as a link to the main view.
- A breadcrumbs will be displayed, showing the page where the user is located as well as a link for navigation.
- On the right side of the header, the number of items that have been added to the cart will be displayed.

Search Bar (SEARCH)

- An input will be displayed to the user, which will allow the introduction of a text string.
- The user must filter the products based on the text entered, and it will be compared with the Make and Model of the products.
- The filtering will be in real time, i.e. a search will be launched each time the user changes the search criteria.

List item (ITEM)

- The following product information will be displayed:
  - Image
  - Brand
  - Model
  - Price

Product Image (IMAGE)

- The product image will be displayed

Product Description (DESCRIPTION)

- The details associated with the products will be displayed. At least the following attributes will be displayed:
  - Brand
  - Model
  - Price
  - CPU
  - RAM
  - Operating System
  - Screen resolution
  - Battery
  - Cameras
  - Dimensions
  - Weight

Product Actions (ACTIONS)

- Two types of selectors will be displayed, where the user will be able to select the type of product he/she wants to add to the cart. The option selectors for the following attributes will be displayed:
  - Storage
  - Colors
- Even if there is only one option, the selector will be displayed with the information. For this use case, it should be selected by default.
- An Add button will be displayed, where the user, after selecting the options, will add the product to the cart.
- When adding a product via the API, the following information is required to be sent:
  - The product identifier
  - The selected color code
  - The code of the selected storage capacity
- The add request returns, in the response, the number of products in the basket. This value must be shown in the application header in any view of the application. This requires persisting the data.

RESOURCES

API Integration

To be able to perform the test, it is required to integrate with an API for data management.

The API domain will be the same for all Endpoints, and will be as follows:

```
https://front-test-api.herokuapp.com/
```

Endpoint definitions are as follows:

- Obtain the list of products

  Path

  ```
  GET /api/product
  ```

  Response

  ```
  [
      {
          id: 0001,
          ...
  ```

- Get Product Detail

  Path

  ```
  GET /api/product/:id
  ```

  Response

  ```
  {
      id: 0001,
      ...
  ```

- Add product to cart

  Path

  ```
  POST /api/cart
  ```

Body

```
{
    id: 0001
```

Response

```
{
    count: 1
}
```

Data persistence

It is required to add a client storage of the data received from the API. What we want to offer is a caching system, so that requests to the API are not made every time. For them, it is required to define the following functionality:

- The information will be stored each time the API service is requested.
- This information will be saved, and will have an expiration of 1 hour, once this time is exceeded, the information must be revalidated.
- Any storage method can be used to store this information, either in the browser or in memory, but always on the client.