BarBooks Coding Assessment

Develop a solution that allows the user to search for any game he wants.

Instructions

1. Use your time to deliver a solution that showcases your coding skills and the level of quality you expect (but no need to gold plate it); we are interested in the internal quality of your solution as much as in that it works.

2. In a real-life scenario, you would ask questions to clarify any doubts but for this assessment, document the questions you would ask and provide your answers in the readme file.

3. Once complete, publish your code on GitHub and let us know through the email talent@barbooksaustralia.com

4. Make sure your package contains a README file with all relevant information necessary to run your solution, including:

    a. What kind libraries your project is using?
    b. What could you do better in your code next iteration?
    c. Any other notes you feel relevant for the evaluation of your solution.

5. When doing a code test or a pairing exercise we want you to follow the development best practices and make sure you put as much emphasis on code quality as possible. But don't try to over-engineer it. Clean, scalable, well-tested code is what we're after.

6. Thanks for taking the time to complete the coding assessment. We look forward to receiving your solution! **And please, keep this document confidential.**

Constraints

- You must use JavaScript technology for most of the solution and we expect you to do it using **React** or **Vue**.

- The solution must be implemented with an acceptable level of automated tests; we should be able to verify it from the command line (a.k.a. npm test)

- Your back-end component must integrate with the **API Endpoints**. The service will return a JSON containing all the information you need to build your app.

- You are allowed to use a code generator (e.g. create-react-app, @vue/cli etc) to create your initial setup only.

- You are free to use libraries (eg. Vue, VueX, React, Redux, Jest, ExpressJS, React-Select, ReactQuery, useSWR, and others) but you cannot use anything that creates the structure/scaffolding for you like frameworks.

- Please DO NOT use any UI/UX framework like React-Bootstrap, ReactStrap, MaterialUI, Vuetify, BootstrapVue, Vue Material, etc. Create your visual components on your own using your CSS skills.

- The solution must run via the command-line and we should be able to boot it with a single command. The fewer dependencies on the operating system, the better. Once the minimum requirements are met, we must be able to boot it with a one-liner. (e.g. npm start)

- A database server isn't required, if needed, mock the data in any application layer.

API Documentation

API URL

https://api.dev.cloud.barbooksaustralia.com/code-challenge/api

Endpoints

1. Games list

   GET **/games**

2. Games by platform

   GET **/games?platform=pc**

   - Insert platform, eg: pc, browser

3. Games by category or tag

   GET **/games?category=shooter**

   - Insert game category or tag, eg: mmorpg, shooter, strategy, moba, racing, sports, etc.

4. Sort games by release date, alphabetical or relevance

   GET **/games?sort-by=alphabetical**

   - Insert sort by, eg: release-date, alphabetical or relevance

5. Games by platform & category & sorted

   GET **/games?platform=browser&category=mmorpg&sort-by=release-date**

6. Filter Games by multiple tags for personalized results

   GET **/filter?tag=3d.mmorpg.fantasy.pvp&platform=pc**

   - Insert tag, eg: mmorpg, shooter, pvp, mmofps, etc...

   - Optionally you can also use the "platform" and "sort" parameters

7. Categories List

   GET **/categories**

8. Return details from a specific game

   GET **/game?id=452**

API CORS Support

For local development, we would recommend using a proxy middleware.

1. https://create-react-app.dev/docs/proxying-api-requests-in-development/#configuring-the-proxy-manually

2. https://cli.vuejs.org/config/#devserver-proxy

Rate Limits

Please avoid doing more than 4 requests per second.

Responses

200: Success

404: Object not found: Game or endpoint not found

429: Too many requests

500: Something wrong on our end (unexpected server errors) v1.0.0

Frontend Guidelines

General

- You **must** use CSS Modules. (https://github.com/css-modules/css-modules)
- Prefer to use functional components instead of classes.
- You can use Less, SSCS, CSS.
- Keep your code clean and most importantly, keep it readable.
- Using lazy load images is a plus but not mandatory.
- Thinking about usability is also a plus.

Home page

- The search box must run in memory and filtering by name only.
- The user must be able to type the category (like an auto-complete).
- Multiple categories can be added (eg. tags). *Check the API documentation for more information.*
- The filters are complementary and must be sent to the API generating a new request on every change.
- The state of the filter must remains when navigating between pages.

Details page

- You must list all the information that comes from the API.
- The "back" button must return to the home page.

Wireframes

See the document attached.