Introduction:

Fever Pets

Welcome to our awesome project, *Fever Pets! Together*, we are gonna build the best place for our pets to be, so we can show them to the world. Isn't it fantastic?

We are thrilled to start working with you, so **the first step is to create a github repository, private to your account, that contains the text in this file as the standard Readme file, and invite the github account **FeverCodeReviews** as collaborator**, so we can check it out.

Once that's done, we would like you to perform a few tasks, but first let us introduce the Fever Pets® project.

Fever Pets will be the main place for our users to see data about pets. What pets you say? Well, ALL pets of course, we don't care about the owners, where those pets are based, or anything. In fact, we don't even have that information!

In order to have Fever pets working, we already had our backend team working, and they are offering two endpoints you can consume:
* https://my-json-server.typicode.com/Feverup/fever_pets_data/pets/?_page=<page_id> : To fetch the list of pets on the server, with the first <page_id> being 1.
    * Optionally, "_limit" can be used as query param to specify page size. Otherwise, it's 10 by default. Not specifying _page param will return the whole list of pets. This is NOT how the endpoint should be used.
* https://my-json-server.typicode.com/Feverup/fever_pets_data/pets/<pet_id> : To fetch the details about a particular pet with id <pet_id>

The endpoint for retrieving the pet list will return as body a list of Pet objects

Additionally, on the headers, a "link" header will be returned containing the following relations:
* "next": The url to get the next page, if any, won't come otherwise
* "prev": The url to get the previous page, if any, won't come otherwise
* "first": The url to get the first page
* "last": The url to get the last page

The details endpoint will return a Pet object, with the following structure:
* id: Int the pet's unique id
* name: String The pet's name
* kind: String The pet's kind (dog or cat on the first version)
* weight: Int The pet's weight, in grams
* height: Int The pet's height in centimeters
* length: Int The pet's length, in centimeters
* photo_url: String The pet's picture url
* description: String A small text about the pet
* number_of_lives: Int between 1 and 7, only for Cats

Now that the endpoints are clear, what will Fever Pets® do on its first version?
* It will have a home page that will list the pets returned by the list endpoint, with the name of the pet and it's basic traits, as well as the pet's thumbnail. This won't display the description inside the pet structure
* Home page will allow to sort the pets based on the quantifiable criteria: weight, length, height, name, or kind.

* There will be a detail page that will be accessible from the home page, by tapping on the cell/row for a particular pet
* The detail page will display the photo of the pet in a bigger size, and all the data, including the description
* Navigation back and forth between detail and home should be possible
* If any kind of sorting is applied, and detail is opened, navigating to home should preserve this order.
* The pets have a “health” assigned, weight / (height * length). The health has three tiers:
    * unhealthy: below 2 or over 5
    * healthy: between 3 and 5
    * very healthy: between 2 and 3
    * If the pet is a cat and the number of lives is 1, the pet is always “unhealthy”
* There should be a button that gives us the “pet of the day”. Every calendar day there should be a favourite pet for the whole day.

That’s basically our glorious Fever Pets® v1.0, and I’m sure you are already thrilled to start dealing with it!

In order for us to be able to see the progress, and give you feedback, we would ask for it to be done by making several PRs to the project (don't wait for them to be approved):
* PR #1: Create the project basic structure, this is just the skeleton of the project for whichever framework you have chosen to use (Angular, Vue, or whichever you fancy using). This PR should not contain any logic related to Fever Pets®
* After that, create as many pull requests as you want for the desired features. As we cannot review the PRs on time, just merge them to development or master and continue your code test, no need to wait for us. Every decision made during the development, should be written either on the body of the pull request or on the README.md

Steps:

1) **To create a github repository, private to your account, that contains the text in this file as the standard Readme file, and invite the github account **FeverCodeReviews** as collaborator.**

2) They will tell me a few tasks.

Goals:

1) Fever Pets will be the main place for our users to see data about pets.

We don’t care about the owners, where those pets are based, or anything. In fact, we don’t even have that information!

Notes:

- We already had our backend team working.

Endpoints:

1) https://my-json-server.typicode.com/Feverup/fever_pets_data/pets/?_page=<page_id>


```
[
  {
    "id": 1,
    "name": "Jade",
    "kind": "dog",
    "weight": 2741,
    "height": 20,
    "length": 35,
    "photo_url": "https://cdn2.thedogapi.com/images/rJH9KQoEX.gif",
    "description": "Bacon ipsum dolor amet ham pork belly cupim rump salami"
  },
  {
    "id": 2,
    "name": "Stinky",
    "kind": "cat",
    "weight": 6712,
    "height": 25,
    "length": 52,
    "photo_url": "https://cdn2.thecatapi.com/images/2lt.jpg",
    "description": "I shall purr myself to sleep pee in the shoe scratch at fleas",
    "number_of_lives": 5
  },
...

  {
    "id": 10,
    "name": "Oscar",
    "kind": "cat",
    "weight": 3612,
    "height": 25,
    "length": 49,
```

```
    "photo_url": "https://cdn2.thecatapi.com/images/b93.png",
    "description": "Meow meow mama bite off human's toes",
    "number_of_lives": 1
  }
]
```


Wit the enpoints without page params, it lists all pets(30):

https://my-json-server.typicode.com/Feverup/fever_pets_data/pets/

It lists 10 pets per page.

For example, to show the page 2 (from id: 11 to id: 20):

https://my-json-server.typicode.com/Feverup/fever_pets_data/pets/?_page=2

Optionally, "_limit" can be used as query param to specify page size. Otherwise, it's 10 by default. Not specifying _page param will return the whole list of pets. This is NOT how the endpoint should be used.

Important: Don't return the whole list of pets not specifying _page param

2) https://my-json-server.typicode.com/Feverup/fever_pets_data/pets/<pet_id>

To fetch the details about a particular pet with id <pet_id>

For example:
https://my-json-server.typicode.com/Feverup/fever_pets_data/pets/6
returns:


```
{
  "id": 6,
  "name": "Snap",
  "kind": "cat",
  "weight": 4623,
  "height": 26,
  "length": 50,
  "photo_url": "https://cdn2.thecatapi.com/images/8k7.jpg",
  "description": "I hide behind curtain when vacuum cleaner is on scratch strangers and poo on owners food but meow",
  "number_of_lives": 7
}
```


Additionally, on the headers, **a "link" header will be returned containing the following relations**:
* "**next**": The url to get the next page, if any, won't come otherwise
* "**prev**": The url to get the previous page, if any, won't come otherwise
* "**first**": The url to get the first page
* "**last**": The url to get the last page

For example:with the endpoint:
https://my-json-server.typicode.com/Feverup/fever_pets_data/pets/?_page=2

we've the response in Headers:

link →<http://my-json-server.typicode.com/Feverup/fever_pets_data/pets/?_page=1>; rel="first", <http://my-json-server.typicode.com/Feverup/fever_pets_data/pets/?_page=1>; rel="prev", <http://my-json-server.typicode.com/Feverup/fever_pets_data/pets/?_page=3>; rel="next", <http://my-json-server.typicode.com/Feverup/fever_pets_data/pets/?_page=3>; rel="last"

Pet object details:

The details endpoint will return a Pet object, with the following structure:

* **id**: Int the pet's unique id

* **name**: String The pet's name

* **kind**: String The pet's kind (**dog or cat** on the first version)

* **weight**: Int The pet's weight, in grams

* **height**: Int The pet's height in centimeters

* **length**: Int The pet's length, in centimeters

* **photo_url**: String The pet's picture url

* **description**: String A small text about the pet

* **number_of_lives**: Int **between 1 and 7, only for Cats**

Fever Pets v 1.0

1) It will have a home page that will list the pets returned by the list endpoint, with the name of the pet and it’s basic traits, as well as the pet’s thumbnail. This won’t display the description inside the pet structure

Home page

Home **Details**

Pets list:

name	kind	weight	height	length	thumbnail

2) Home page will allow to **sort the pets based on the quantifiable criteria**: weight, length, height, name, or kind.

Sorting columns (kind) and the rest by asc/desc criteria

3) There will be a detail page that will be accessible from the home page, by **tapping on the cell/row for a particular pet**

Detail page (<===== Home)

The detail page will display the **photo** of the pet in a **bigger size**, and **all the data**, including the description
Navigation back and forth between detail and home should be possible.

Home

Jade

Dog

Weight: 2741 grms.
Height: 20 cm.
length: 35 cm.

Description:

Bacon ipsum dolor amet ham pork belly cupim rump salami

Important:

- If **any kind of sorting** is applied, and detail is opened, **navigating to home should preserve this order**.

Health = $\frac{\text{Weight}}{\text{Height} * \text{Length}}$

unhealthy		very healthy	healthy		unhealthy	
0	1	2	3	4	5	6

The pets have a “health” assigned, weight / (height * length). The health has three tiers:

* **unhealthy**: below 2 or over 5
* **healthy**: between 3 and 5
* **very healthy**: between 2 and 3

* **If the pet is a cat and the number of lives is 1, the pet is always “unhealthy”**

Pet of the day

There should be a button that gives us the “pet of the day”. **Every calendar day there should be a favorite pet for the whole day.**