Cute cats web app (Django + VueJs) — Get datas from API

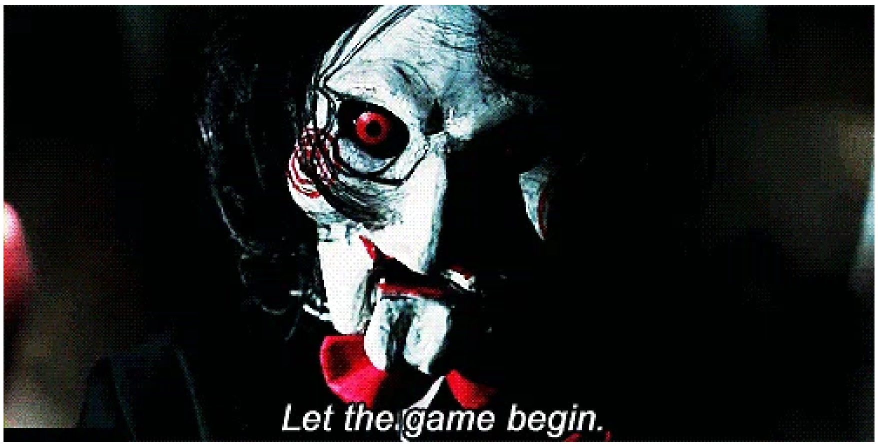
Jrmy Follow
Jul 9, 2018 · 11 min read

Summary

Introduction

In previous part we built a Rest API entrypoint to save some informations about cute cats. To build those API we use Django Rest Framework and create two entry points. By requesting URL '/api/cats' we get a list of cats saved by ourself. We can get more details about a cat by requesting '/api/cats/cat_id'.

In this part, we are going to see how to use VueJs and some tools to request the API and to display the retrieved datas.



Install some tools

To continue, we need to install some librairies

```
npm install --save axios vue-axios vuex
```

We use

- *axios* and *vue-axios*, to easily create requests
- *Vuex*, a state management pattern to code an ordered project

Vue-axios, a friend to make request

It wrap axios, a library for requesting our built REST API.

In *src* folder, create a *'services'* folder who will contain *api.service.js*. In this one, ApiService is an object containing method to request server. For the moment we just need get method.

```
# src/services/api.service.js

import Vue from 'vue'
import axios from 'axios'
import VueAxios from 'vue-axios'
import { API_URL } from '@/services/config'

const ApiService = {
  init () {
    Vue.use(VueAxios, axios)
    Vue.axios.defaults.baseURL = API_URL
  },

  get (resource, slug='') {
    return Vue.axios
            .get(`${resource}\${slug})
            .catch((error) => {
              throw new Error(`ApiService ${error}`)
            })
  },
}

export default ApiService

# src/services/config.js

export default {}
export const API_URL = 'http://localhost:8000/api'
```

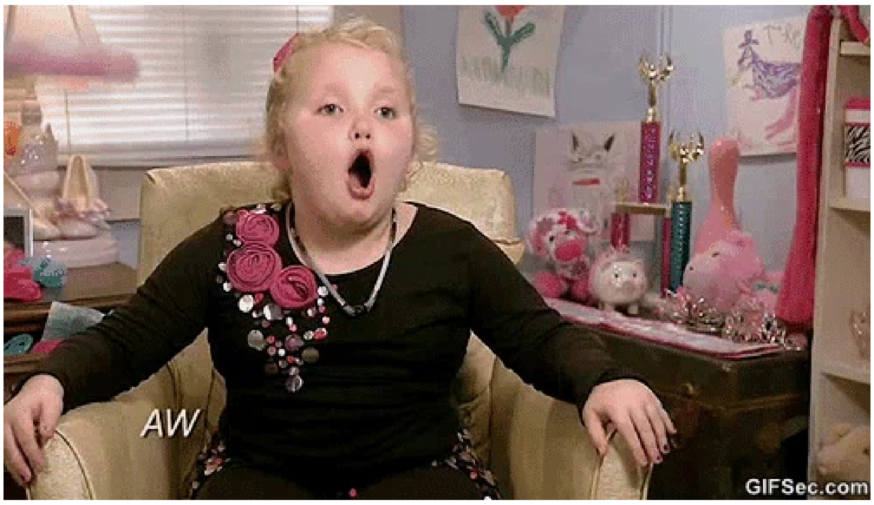
ApiService methods :

- init(), tell to Vue to use VueAxios and Axios, then define base, or the root, of all futur request.
- get(resource), resource param comes to complete the baseUrl. This method allow us to do a HTTP get request.

Before to use the ApiService, let's see what is Vuex and how to use it.

Vuex

"Vuex is a state management pattern + library for Vue.js applications. It serves as a centralized store for all the components in an application, with rules ensuring that the state can only be mutated in a predictable fashion"—*VueJs documentation*



There are five keywords to know :

States—Getters—Actions—Mutations—Store

- **Store** is a collection of states and it exist only one by application
- **State** is a value or an object that you can reach everywhere in your code but you can't modify it directly
- **Getters** is the way to access and compute your *states*, but s*tates* must never change
- **Mutations** is the only way to change a *state*.
- **Actions** allow us to call, to commit, *mutations*. *Actions* can contain asynchronious operations.

Store is the place to be ! Let's create it. Begin to create a new *store* folder, then, an *index.js* file. The *index.js* file will be the entry point when *store* is imported, so this one will map to others *strore* modules.

```
#src/store/index.js

import Vue from 'vue'
import Vuex from 'vuex'

import cats from './cats.module'

Vue.use(Vuex)

export default new Vuex.Store({
  modules: {
    cats
  }
})
```

Below, we are importing *vuex* and cats module. We are creating a new store, and add cats in it. Cats module contains all states, getters, actions and mutations related to cats. In store folder, create a *cats.module.js* file

```
# src/store/cats.module.js

import ApiService from '@/services/api.service'
import { FETCH_CATS,
         FETCH_A_CAT
      } from './actions.type'
import { FETCH_START,
         FETCH_END,
         SET_CATS,
         SET_ERROR
       } from './mutations.type'

const state = {
  // TODO: a list of state related to cats
}

const getters = {
  // TODO: define method to access state value
}

const actions = {
  // TODO: define actions like FETCH_A_CAT
}

const mutations = {
  // TODO: define mutations to redefine state value
}
```

```
export default {
  state,
  getters,
  actions,
  mutations
}
```

Below is the skeleton of a store module. First, we import some actions and mutations related to the cats API. An action like FETCH_CATS will send a request to the API and get the list of cats. A mutation like SET_CATS will set the state ‘cats’ with the server response.

We need to define imported actions and mutations. To do that, create a *actions.type.js* file and a *mutations.type.js* file in *store* folder.

```
# src/store/actions.type.js

export const FETCH_A_CAT = "fetchACat"
export const FETCH_CATS = "fetchCats"

# src/store/mutations.type.js

// global
export const FETCH_START = "loadingOn"
export const FETCH_END = "loadingOff"
export const SET_ERROR = "setError"
// related to cats
export const SET_A_CAT = "setACat"
export const SET_CATS = "setCats"
```

Now everything it’s ok, start to define *states* in *cats.module*.

Variables we need should be :

- cats : to get cat list from the Rest Api
- cat : to get a specific cat
- errors : in case everything is not working well
- loading : to see the state of the request and be able to display a loading bar

```
#src/store/cats.module
...
const state = {
  cats: [],
  cat: {},
  errors: {},
  loading: false
}
...
```

What about getters ? Remember, getters are the only way to access to states. We want accessing to cats value and the loading state to know if user interface should display the loader or not.

```
#src/store/cats.module
...
const getters = {
  currentCat (state) {
    return state.cat
  },
  cats (state) {
    return state.cats;
  },
  isLoading (state) {
    return state.loading;
  }
}
...
```

Mutations are the only way to change states values.

```
#src/store/cats.module
...
const mutations = {
  [FETCH_START] (state) {
    state.loading = true
  },
  [FETCH_END] (state) {
    state.loading = false
  },
  [SET_CATS] (state, pCats) {
    state.cats = pCats
    state.errors = {}
  },
  [SET_A_CAT] (state, pCat) {
    state.cat = pCat
    state.errors = {}
  },
  [SET_ERROR] (state) {
    state.errors = errors
  }
}
...
```

Mutations must be called by actions. In our case, we have two actions. Getting a list of cats, by requesting "./api/cats" or a specific cat by requesting "./api/cats/cat_id". Those url was define in the previous story about how to create an Rest API entrypoint.

```
#src/store/cats.module
...
const actions = {
  [FETCH_CATS] (context, payload) {
    context.commit(FETCH_START)
    return ApiService
      .get('cats')
      .then(({data})) => {
        context.commit(SET_CATS, data.cats.results);
        context.commit(FETCH_END)
      }
      .catch(({response})) => {
        context.commit(SET_ERROR, 
response.data.errors)
      }
  },
  [FETCH_A_CAT] (context, payload) {
    context.commit(FETCH_START)
    const {cat_id} = payload
    return ApiService
      .get(`cats/${cat_id}`)
      .then(({data}) => {
        context.commit(SET_A_CAT, data.cats);
        context.commit(FETCH_END)
      })
      .catch(({response}) => {
        context.commit(SET_ERROR, 
response.data.errors)
      })
  }
}
...
```

FETCH_CATS action :

- context.commit(FETCH_START) call the action fetchStart to set loading state to true
- Using get method from our API service to request '.api/cats'

- Then, if the request respond with cats datas, we set *cats state* by calling the action *setCats* and set *loading state* to false when it's over.
- Catch, if something goes wrong the actions will return an errors object

FETCH_A_CAT action is pretty similar but we have to get the cat_id before making a request. To do that, we just pick cat_id in payload and set cats states with specific cat detail.

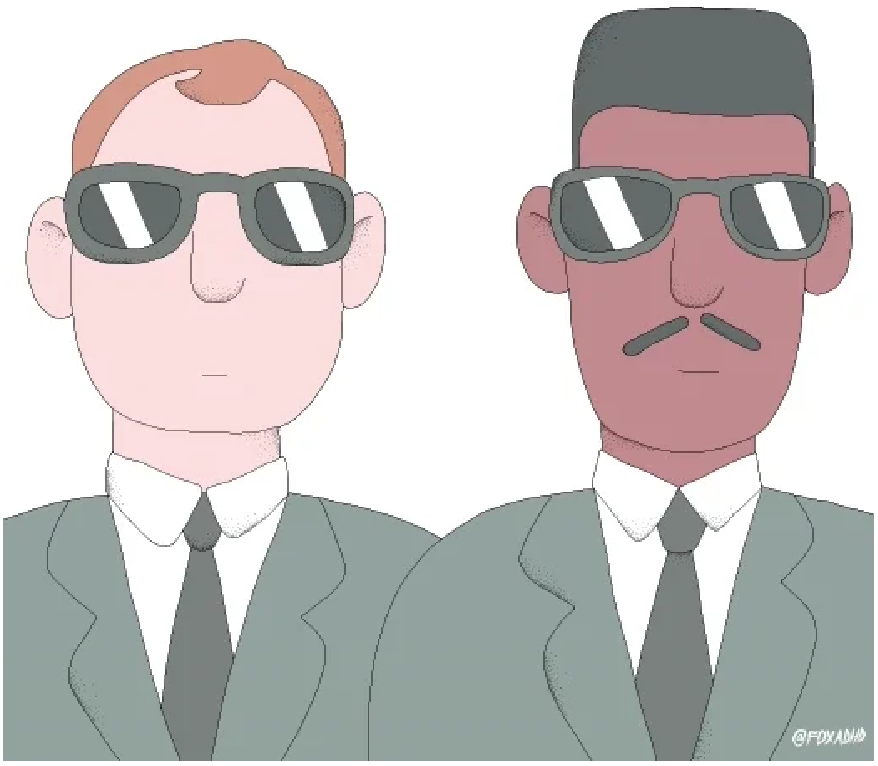
We are now able to request the Rest API and get some values thanks to *cats.module* we have been created. But how to display those datas ?

Display cats list page

We are going to update Home.vue to display the cats list from the Rest API. Current Home.vue file is the following.

```
# src/views/Home.vue

<template>
  <section class="container">
    <h1> Welcome to your APP</h1>
    <redirectButton pathname="cutecat">
</redirectButton>
  </section>
</template>

<script>
  import redirectButton from
'@/components/buttons/redirect'

export default {
    components: {
      redirectButton
    }
  }
</script>
```

The goal now is to replace the redirectButton by a catsList component.



So let’s create this component.

```
# src/components/CatList.vue

<template>
  <ul id="catList">
    <li v-for="cat in cats">
        <div v-if="cat.photo" class="photo-container">
          <img :src="cat.photo"
               alt="cat.name"
               height="200px"
               width="200px"/>
        </div>
        <div v-else class="photo-container">
          <img
src="http://icons.iconarchive.com/icons/sonya/swarm/25
6/Cat-icon.png"
               alt="cat.name"
               height="200px"
               width="200px"/>
        </div>
      <div class="description">
        <p><span class="grey"> Name </span>{{ cat.name
}}</p>
        <p><span class="grey"> Owner </span>{{
cat.owner }}</p>
        <p><span class="grey"> Age </span>{{ cat.age
}}</p>
```

```
      </div>
    </li>
  </ul>
</template>

<script>
  import { mapGetters } from 'vuex'

export default {
    name: 'catsList',
    computed: {
      ...mapGetters([
        'cats',
        'isLoading'
      ])
    }
  }
</script>
```

CatList component is a list of all cats find in our API. Each li tag will contain :

- Cat photo or a default image
- Cat name
- Owner name
- Age

To access to those data, we have to get state from store. To do that, just use mapGetters with the name of whised getter.

Let’s make a little update in Home.vue

```
# src/views/Home.vue

<template>
  <section class="container">
    <h1> Our cuttest cats </h1>
    <catList></catList>
  </section>
</template>

<script>
  import catList from '@/components/CatsList'

  export default {
    components: {
      redirectButton,
      catList
    }
  }
</script>
```

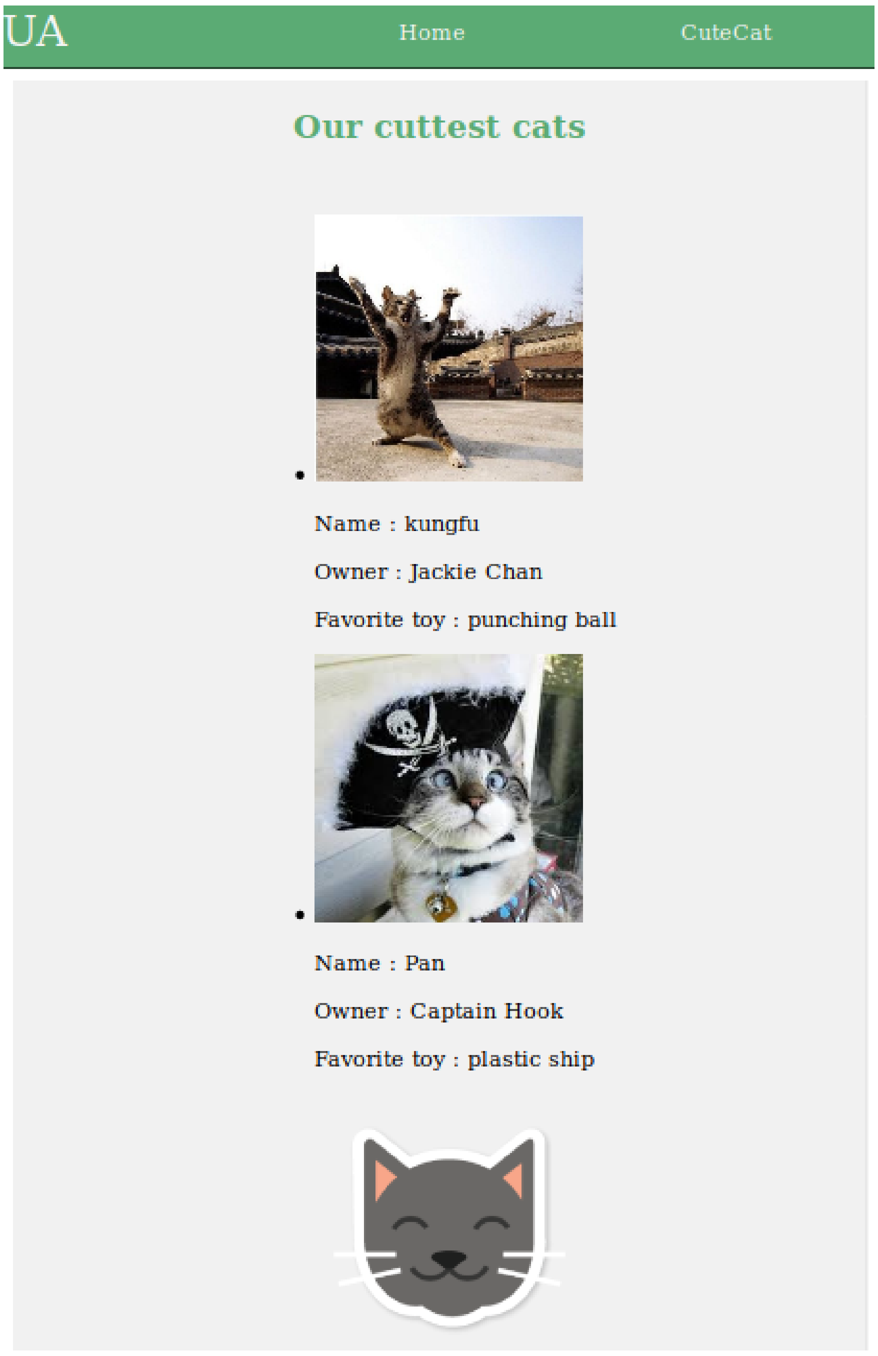
Don't try to run this code, nothing will be display. In fact, even if we call *cats* getter, this one is empty because it's the initial value. We have to update this value by requesting the API. The code already exist in the store, we just have to call this one.

```
<template>
  <section class="container">
    <h1> Our cuttest cats </h1>
    <catList></catList>
  </section>
</template>

<script>
  import catList from '@/components/CatsList'
  import { FETCH_CATS } from '@/store/actions.type'
  import { mapGetters } from 'vuex'

export default {
    components: {
      catList
    },
    mounted () {
      this.$store.dispatch(FETCH_CATS)
      .then(() => {
        console.log("YOUPI");
      })
      .catch((err) => {
        console.log("ERR : ", err);
      })
    }
  }
</script>
```

Remember, we defined FETCH_CATS action before. To call a specific action we use *dispatch* store method and we do this in mounted.

Guess what ? We can build VueJs project and check the result.

```
# as always

npm run build
python manage.py collectstatic
python manage.py runserver
```



Our cuttest cats

- Name : kungfu

  Owner : Jackie Chan

  Favorite toy : punching ball
- Name : Pan

  Owner : Captain Hook

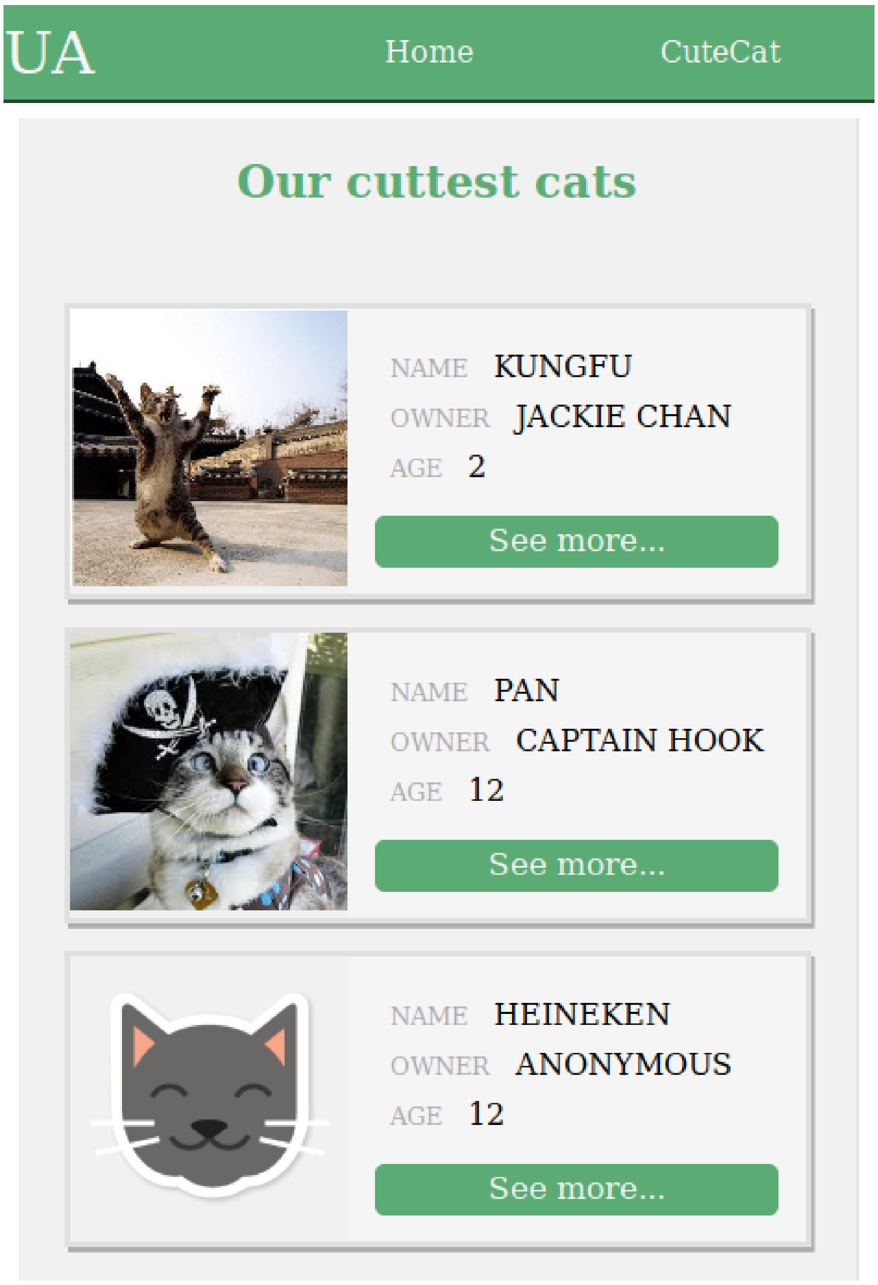
  Favorite toy : plastic ship

The job is done ! As we can see, some styling stuff are required.

```
# src/components/CatList.vue

<template>
 ...
</template>
```

```
<script>
 ...
</script>
```

```
<style scoped lang='sass'>
  #catList
    min-width: 85%
    li
      list-style: none
      display: flex
      justify-content: space-between
      border: solid 3px #e0e0e0
      margin: 15px 0
      box-shadow: 2px 3px #adadad
      .description
        flex: 2
        padding: 10px 15px
        background: #f5f5f5
        p
          text-transform: uppercase
          span
            font-size: 18px
            color: darkgrey
            padding: 0 10px 0 0
</style>
```


UA
Home
CuteCat
Our cuttest cats
NAME KUNGFU
OWNER JACKIE CHAN
AGE 2
See more...
NAME PAN
OWNER CAPTAIN HOOK
AGE 12
See more...
NAME HEINEKEN
OWNER ANONYMOUS
AGE 12
See more...

Well done ! Now it could be cool to have a clickable button linked to cat details page. Let’s update CatsList.vue

```
# src/components/CatsList.vue

<template>
  <ul id="catList">
    <li v-for="cat in cats">
        <div v-if="cat.photo" class="photo-container">
          <img :src="cat.photo"
               alt="cat.name"
               height="200px"
               width="200px"/>
        </div>
        <div v-else class="photo-container">
          <img
src="http://icons.iconarchive.com/icons/sonya/swarm/25
6/Cat-icon.png"
               alt="cat.name"
               height="200px"
               width="200px"/>
        </div>
      <div class="description">
        <div>
          <p><span class="grey"> Name </span>{{
cat.name }}</p>
          <p><span class="grey"> Owner </span>{{
cat.owner }}</p>
          <p><span class="grey"> Age </span>{{ cat.age
}}</p>
        </div>
        <div class="more">
          <router-link :to="{ name: 'cutecat', params:
{cat_id: cat.id} }">See more...</router-link>
```

```
          </div>
        </div>
      </li>
    </ul>
</template>

<script> ... </script>

<style>
 #catList
    min-width: 85%
    li
      list-style: none
      display: flex
      justify-content: space-between
      border: solid 3px #e0e0e0
      margin: 15px 0
      box-shadow: 2px 3px #adadad
      .description
        flex: 2
        padding: 10px 15px
        background: #f5f5f5
        display: flex
        flex-direction: column
        justify-content: space-around
        div
          p
            text-transform: uppercase
            span
              font-size: 18px
              color: darkgrey
              padding: 0 10px 0 0
        .more
          height: 35px
          text-align: center
          padding-top: 8px
          font-size: 19px
          background: #5bab74
          border-radius: 5px
          &:hover
            background-color: #3e8353
          a
            text-decoration: none
            color: #f5f5f5

</style>
```

and also router/index.js file

```
# src/router/index.js

import Vue from 'vue'
import Router from 'vue-router'
import Home from '@/views/Home'
import CuteCat from '@/views/CuteCat'

Vue.use(Router)

export default new Router({
  routes: [
    {
      path: '/',
      name: 'home',
      component: Home
    },
    {
      path: '/cutecate/@:cat_id',
      name: 'cutecat',
      component: CuteCat
    }
  ]
})
```

Display cat details page

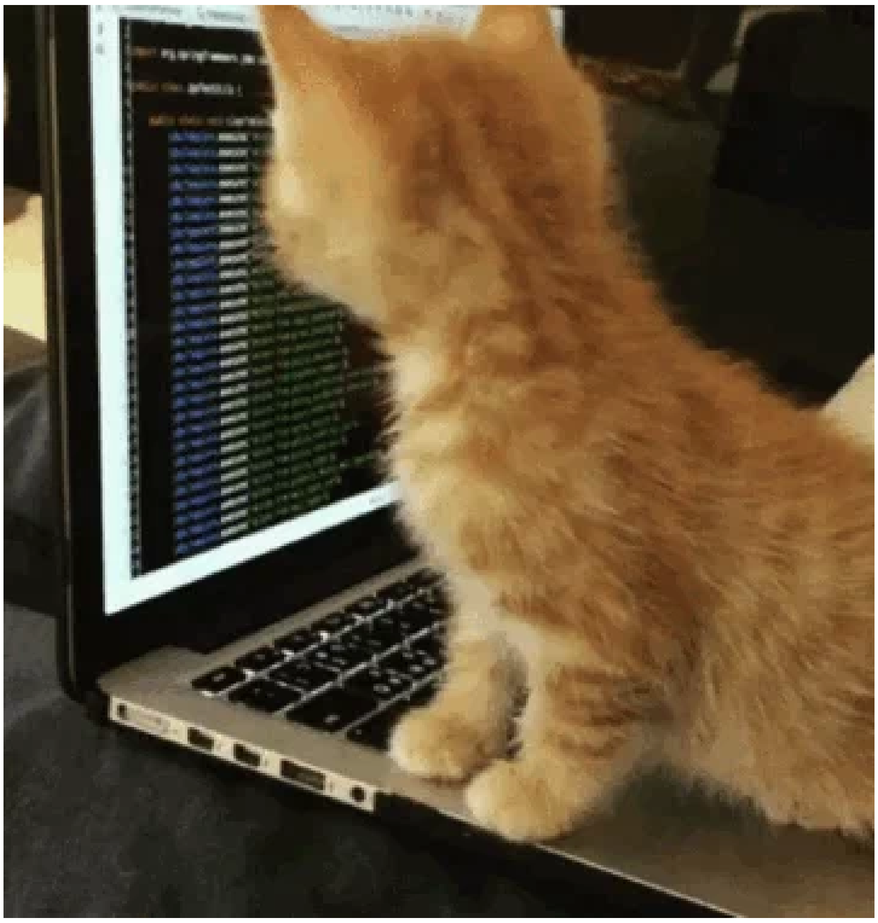
A router is now available to reach specific cat details by the cat ID. The *cat_id* param will be used to request *'./api/cats/cat_id'*.

As we did before, we create a cat ID card component, let's say a idCard component. This card will be import in *CuteCat.vue.*

```
# src/components/idCard.vue

<template>
  <div id="idCard">
    <h1>{{ currentCat.name }}</h1>
    <div id="idPhoto">
      <div v-if="currentCat.photo" class="photo-
container">
        <img :src="currentCat.photo"
             alt="currentCat.name"
             height="250px"
             width="250px"/>
      </div>
      <div v-else class="photo-container">
        <img
src="http://icons.iconarchive.com/icons/sonya/swarm/25
6/Cat-icon.png"
             alt="currentCat.name"
             height="250px"
             width="250px"/>
      </div>
    </div>
    <div class="description">
      <div>
        <p><span class="grey"> Owner </span>{{
```

```
currentCat.owner }}</p>
        <p><span class="grey"> Age </span>{{
currentCat.age }} yo</p>
        <p><span class="grey"> Color </span>{{
getColor }}</p>
        <p><span class="grey"> Mood </span>{{ getMood
}} </p>
        <p><span class="grey"> Favorite toy </span>{{
currentCat.toy }}</p>

</div>
    </div>
  </div>
</template>

<script>
import { mapGetters } from 'vuex'


export default {
  name: 'idCard',
  computed: {
    ...mapGetters([
      'currentCat',
      'isLoading'
    ]),
    getColor () {
      let color = '';
      if (this.currentCat.colors === "BLK") {
        color = 'Black'
      } else if (this.currentCat.colors === "GRY") {
        color = 'Grey'
      } else if (this.currentCat.colors === "RED") {
        color = 'Red'
      };
      return color;
    },
    getMood () {
      let mood = '';
      if (this.currentCat.mood === "HAY") {
        mood = 'Happy'
      } else if (this.currentCat.mood === "GRY") {
        mood = 'Grumpy'
      } else if (this.currentCat.mood === "MIC") {
        mood = 'Milkholic'
      };
      return mood;
    }
  }
}
</script>
```

Details data are placed in the state *currentCat*. For the moment, this state is an empty object. We need to retrieve datas by requesting the API. A little recap, on the home page, a list of cat is displayed. A button linked to cat details page is available for each cat. On click a new page will be show with cat_id as url parameter. When the content is mounted, a request will be send. Let's code it !



```
# src/views/CuteCat.vue

<template>
  <section class="container">
    <idCard></idCard>
  </section>
</template>

<script>
  import idCard from '@/components/idCard'
  import { FETCH_A_CAT } from '@/store/actions.type'
  import { mapGetters } from 'vuex'

export default {
    data () {
      return {
      }
    },
    components: {
      idCard
    },
    mounted () {
      this.$store.dispatch(FETCH_A_CAT,
this.$route.params)
      .then(() => {
        console.log('YOUPI');
      })
      .catch(() => {
        console.log('ERR : ', err);
      })
    }
  }
</script>
```

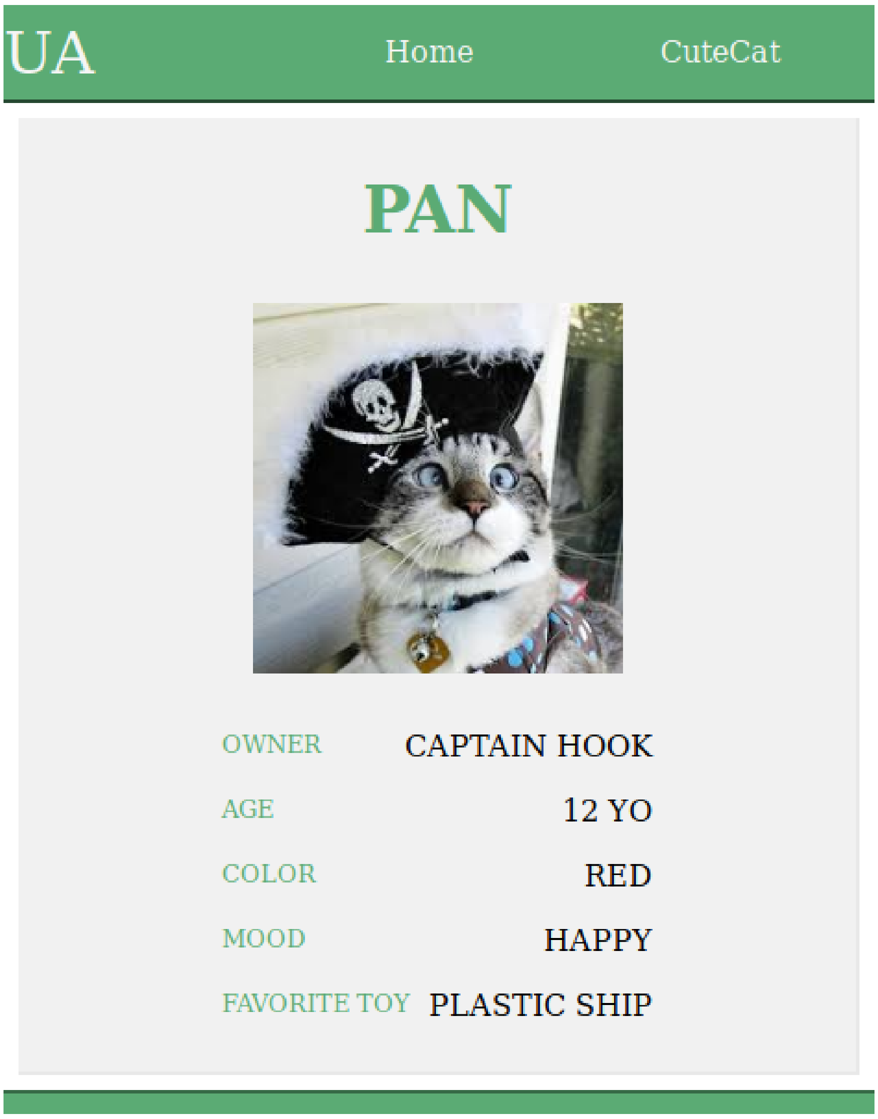
Import the idCard component and use it in template. In mounted, call the action FETCH_A_CAT by using *dispatch*. This time, the request need *cat_id* parameter. Use *this.$route.params* to get *cat_id* value show in url.

Add some style

```
# src/views/CuteCat.vue

<style scoped lang="sass">
  @import './../assets/styles/colors'
  @import './../assets/styles/global'

.container
    @include container-style
    min-height: 77vh
    h1
      color: $green
    p
      margin: 0
      color: lighten($green, 10%)
    img
      width: 100px
      height: 100px
</style>

# src/components/idCard.vue

<style scoped lang="sass">
  @import './../assets/styles/colors'
  @import './../assets/styles/global'

h1
    color: $green
    text-transform: uppercase
    text-align: center
    font-size: 40px
  #idPhoto
    text-align: center
```

```
      .description
        flex: 2
        padding: 10px 15px
        display: flex
        flex-direction: column
        justify-content: space-around
        div
          p
            text-transform: uppercase
            display: flex
            justify-content: space-between
            span
              font-size: 18px
              color: $green
              padding: 0 10px 0 0
  </style>
```

Build your project as usual and check the results. On home page click on See more button. I choose Pan !



PAN

OWNER	CAPTAIN HOOK
AGE	12 YO
COLOR	RED
MOOD	HAPPY
FAVORITE TOY	PLASTIC SHIP

The job is done ! congratulation !

Conclusion

In this part we have learnd how to get data from Rest API using VueJs, Axios and Vuex.

First we build an api service to make request easily with Vue-axios. For the moment we have defined get method only, to go further, in next part, it could be nice to define other request like post or put.

Then, with Vuex, we have created a cat module with states, getters, actions and mutations. It's a useful and secure way to access to data everywhere in the application.

Finally, we built a user interface to display those datas by using VueJs. We saw how to call, to dispatch, actions and get states values thankfull to mapGetter.

Next part could be about POST request. How to manage a form and to send datas to the server and save them.