Laravel + Vue Apollo Learning

Contents:

Create project

- Use command line: create-project –prefer-dist laravel/laravel name-project to create the project
- Configurations on .env file to connect to database

```
DB_CONNECTION= mysql
DB_HOST=ip address
DB_PORT=port
DB_DATABASE=database name
DB_USERNAME=username
DB_PASSWORD=password
```

- Open terminal in project and then type: php artisan key:generate to generate key for run project
- Create model Post that use command line: php artisan make:model name-model -a (use -a meaning onetime it to create model, controller, migration and factory)
- Go to migration file to add fields in table that path: /database/migrations/
- Go to model to filter field of table
- Go to path: routes\web.php

```
Route::get('{any}', function () { return view('welcome'); })->where('any', '.*');
```

- Go to path: routes\api.php to create route for each that use method from controller.
- Use command line to run project: php artisan serve and then use tool for test api for know it working or not.

Install VueJS on project

- Use command line for installing laravel/ui package: php artisan ui vue
- Use command line: npm install
- Use command line for create vue router: npm install vue-router vue-axios -save
  (use vue routes for routing to use different components and vue-axios for sending request to network server.
- Go to path: \resources\js\app.js
  import VueRouter from 'vue-router';
  import VueAxios from 'vue-axios';
  import axios from 'axios';
  Vue.component('example-component', require('./components/ExampleComponent.vue'));
- Go to laravel each file resources\welcome.blade.php:

```
<!DOCTYPE html>
<html lang="{{ str_replace('_', '-', app()->getLocale()) }}">
<head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>Post</title>
    <link rel="stylesheet" href="{{ asset('css/app.css')}}">
    <link rel="preconnect" href="https://fonts.gstatic.com">
    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css">
    <link href="https://fonts.googleapis.com/css2?family=Roboto+Mono:wght@600&display=swap" rel="stylesheet">
</head>
```

```
<body>
    <div id="app">
    </div>
    <script src="{{ asset('js/app.js') }}"></script>
    <script src="https://code.jquery.com/jquery-3.2.1.slim.min.js"></script>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.12.9/umd/popper.min.js"></script>
    <script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/js/bootstrap.min.js"></script>
</body>
</html>
```

- Go to path: \resources\js\views\posts to create file
    1. Create.vue
    2. Edit.vue
    3. Post.vue
    4. Show.vue
- Go to create file for routes on path: \resources\js\routes\index.js

```
import Vue from 'vue';
import VueRouter from 'vue-router';

import Post from '../views/posts/Post.vue';
import Create from '../views/posts/Create.vue';
import Edit from '../views/posts/Edit.vue';
import Show from '../views/posts/Show.vue';

Vue.use(VueRouter);

const routes = [
{
      path: '/',
      name: 'Post',
      component: Post
},
{
      path: '/create',
      name: 'Create',
      component: Create
},
{
      path: '/edit/:id',
      name: 'Edit',
      component: Edit
},
{
      path: '/show/:id',
      name: 'Show',
      component: Show
}];

const router = new VueRouter({
      mode:'hash',
      routes
});

export default reouter;
```

- Use command line: npm run watch for compiling and running the project.

Installing Vuex in Vue.js

- Go to Laravel vue project
- Use command line: npm install vuex –save for installing state management
- Go to path: \resources\js\app.js file to import store:

```
….
import store from './store';
…

const app = new Vue({
…
store,
…
});
```

- Go to create folder and file in path: \resources\js
    1. Create folder name ‘store’
    2. Create file name ‘index.js’ in store and configuration in file index.js

```
import Vue from ‘vue’;
import Vuex from ‘vuex’;

import posts from ‘./modules/posts’;

Vue.use(Vuex);

export default new Vuex.Store({
        state: {},
        getters: {},
        mutations: {},
        actions: {},
        modules:{
         posts
        }
});
```

    3. Create folder name ‘modules’ in folder ‘store’
    4. Create file name ‘posts.js’ in modules and configuration in file

```
import { apolloClient } from "../../apollo/index";

import Vue from "vue";
import Vuex from "vuex";
// import axios from 'axios'

import { POSTS,FETCH_POST,CREATE_POST,UPDATE_POST,DELETE_POST, } from
'../../graphql/post'

Vue.use(Vuex)

const state = {
   posts: [],
   post: []
};
const getters = {
   posts: state => {
      return state.posts
   },
   post: state => {
      return state.post
   }
};
const actions = {
   loadPosts({
      commit
   }){
      apolloClient.query({
         query: POSTS
      }).then(response => {
         commit('setPosts',response.data.posts)
         console.log('🚀🚀 loadPosts from actions');
      });
   },

   getPostByID({ commit }, id){
      apolloClient.query({
         query: FETCH_POST,
         variables:{
            id: id
         }
      }).then(response => {
         commit('setPost',response.data.post)
      });
   },
};
```

```
const mutations = {
    setPosts(state, posts){ state.posts = posts },
    setPost(state, post){ state.post = post },
    addPost(state, post){
        state.posts.push(post);
    },
    updatePost(state, data){
        state.posts = state.posts.map(post => {
            if (post.id === data.id) {
              return Object.assign({}, post, data.data)
            }
            return post
        })
    },
    deletePost(state, id){
        var index = state.posts.findIndex(post => post.id === id)
        state.posts.splice(index, 1)
    }
};

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

7. Go to “resources/js/views/posts/Post.vue”

```
<template>
  <div class="m-3">
    <h1 class="text-center m-3">
      Posts List 🖌️
      <router-link :to="{ name: 'Create' }" class="text-center"
        >Create</router-link
      >
    </h1>
    <div class="row mt-2 justify-content-center alert alert-success">
      <div class="col-sm-10">
        <div class="form-group">
          <label for="search">🔍 Search</label>
          <input
            type="text"
            class="form-control"
            placeholder="Search ..."
            v-model="search"
          />
        </div>
      </div>
    </div>
    <div class="row justify-content-center">
      <div class="col-10">
        <ul
          class="list-group list-group-flush"
          v-for="post in filterByPost"
          :key="post.id"
        >
          <li class="list-group-item">
            <div class="float-left">
              <router-link
                :to="{ name: 'Show', params: { id: post.id } }"
                class="text-dark"
                style="text-decoration: none"
              >
                <h5>
                  <span v-if="post.status">✅</span>
                  <span v-if="!post.status">❎</span>
                  📖 {{ post.title }}
                </h5>
              </router-link>
              <p
                v-html="post.description.substring(0, 100) + ' ...'"
                class="text-primary"
              ></p>
            </div>
```

```
      <div class="float-right">
          <router-link
            :to="{ name: 'Edit', params: { id: post.id } }"
            class="card-link"
            >Edit 📝</router-link
          >
          <a
            href="javascript:;"
            class="card-link"
            @click.prevent="deletePost(post.id)"
            >Delete 🔥</a
          >
        </div>
      </li>
    </ul>
  </div>
 </div>
</div>
</template>
<script>
import { mapGetters } from "vuex";
export default {
 name: "Post",
 data() {
  return {
   search: "",
  };
 },
 computed: {
  ...mapGetters(["posts"]),
  filterByPost() {
   return this.posts.filter((post) => {
    return (
     post.title.includes(this.search) ||
     post.description.includes(this.search)
    );
   });
  },
 },
 created() {
  this.$store.dispatch("loadPosts");
 },
 methods: {
  deletePost(id) {
   this.$store.dispatch("remove", parseInt(id));
   this.$store.dispatch("loadPosts");
  },
 },
};
</script>
```

8. Go to “resources/js/views/posts/Create.vue” add methods

```
<template>
 <div>
  <h1 class="text-center">
   Create Post 🖌️ <router-link :to="{ name: 'Post' }">List</router-link>
  </h1>

  <div class="row justify-content-center">
   <div class="col-sm-10">
    <form @submit.prevent="createPost">
     <div class="form-group">
      <label>Title<span class="text-danger">*</span></label>
      <input
       type="text"
       class="form-control"
       placeholder="Enter title"
       v-model="title"
       required
      />
     </div>
     <div class="form-group">
      <label>Description<span class="text-danger">*</span></label>
      <quill-editor
       ref="QuillEditor"
       v-model="description"
       :options="editorOption"
      />
     </div>
```

```
<button type="submit" class="btn btn-primary">Submit</button>
      </form>
    </div>
   </div>
  </div>
</template>
<style>
.ql-editor {
  height: 40vh !important;
}
</style>
<script>
export default {
  name: "Create",
  data() {
    return {
      title: "",
      description: "",
      editorOption: {
        content: "html",
        contentType: "html",
        placeholder: "Write something...",
        theme: "snow",
      },
      errors: []
    };
  },
  methods: {
    createPost() {
      let post = {
        title: this.title,
        description: this.description,
        created_by: 1,
        updated_by: 1,
      };
      this.$store
        .dispatch("add", post)
        .then((responce) => {
          this.$router.push("/");
        })
        .catch(e => {
          console.error(e.message);
        });

    },
  },
  computed: {
    editor() {
      return this.$refs.QuillEditor.quill;
    },
  }
};
</script>
```

9. Go to “resources/js/views/posts/Edit.vue” add methods

```
<template>
  <div>
    <h1 class="text-center">Edit Post 🖌️ <router-link :to="{ name: 'Post' }">List</router-link></h1>

    <div class="row justify-content-center">
      <div class="col-sm-10">
        <form @submit.prevent="editPost">
          <div class="form-group">
            <label>Title<span class="text-danger">*</span></label>
            <input
              type="text"
              class="form-control"
              placeholder="Enter title"
              v-model="post.title"
            />
          </div>
          <div class="form-group">
            <label>Description<span class="text-danger">*</span></label>
            <quill-editor
            ref="QuillEditor"
            :options="editorOption"
            v-model="post.description"
          />
          </div>
```

```
          <div class="form-group">
            <label>Status</label>
            <select v-model="post.status" class="form-control">
              <option :value='false'>Draft</option>
              <option :value='true'>Published</option>
            </select>
          </div>
          <button type="submit" class="btn btn-primary">Submit</button>
        </form>
      </div>
    </div>
  </div>
</template>
<style>
.ql-editor{
  height: 40vh !important;
}
</style>
<script>
import {mapGetters} from 'vuex';
export default {
  name: "Edit",
  data () {
     return {
       editorOption: {
       },
       title: '',
       description: '',
       status: false
     }
   },
   methods: {
     editPost(){
       let data = {
         id: parseInt(this.$route.params.id),
         title: this.post.title,
         description: this.post.description,
         status: this.post.status,
         created_by: 1,
         updated_by: 1
       }
       console.log(data);
       this.$store.dispatch('update', data)
       .then(responce =>{
         this.$router.push('/');
       })
       .catch()
     }
   },
   computed: {
     editor() {
       return this.$refs.QuillEditor.quill
     },
     ...mapGetters(['post'])
   },
   mounted() {
     this.$store.dispatch("getPostByID", parseInt(this.$route.params.id));
   }
};
</script>
```

10. Go to “resources/js/views/posts/Show.vue” add method

```
<template>
  <div>
    <h1 class="text-center m-5">Show 🖌️<router-link :to="{ name: 'Post' }">List</router-link></h1>
    <div class="row">
      <div class="col-sm-12">
        <div class="card bg-white">
          <div class="card-body">
            <h1 class="card-title">📖 {{ post.title }}</h1>
            <p class="card-title" v-html="post.description"></p>
            <div>
              <p class="text-dark" v-if="post.created_by">
                <span class="h2">👨‍💼</span> {{ post.created_by.first_name }} {{ post.created_by.last_name }}<br>
                <span class="h4">📞</span> {{ post.created_by.tel }} <br>
                <span class="h4">📧</span> {{ post.created_by.email }}
              </p>
              <p class="text-dark">Created At: {{ post.created_at }}</p>
              <p class="text-dark">Updated At: {{ post.updated_at }}</p>
            </div>
          </div>
```

```
</div>
      </div>
    </div>
  </div>
</template>
<script>
import { mapGetters } from "vuex";
export default {
  name: "ShowPost",
  computed: {
    ...mapGetters(["post"]),
  },
  created() {
    this.$store.dispatch("getPostByID", parseInt(this.$route.params.id));
  },
};
</script>
```

Install Lighthouse on project

- Use command line: composer require nuwave/lighthouse for Laravel
- Use command line for get public lighthouse's configuration file: php artisan vendor:public –provider="Nuwave\Lighthouse\LighthouseServiceProvider"
- If we take a look at the config/lighthouse file you'll notice a setting used to register our schema file with Lighthouse:

```
'schema' => [
        'register' => base_path('graphql/schema.graphql')
],
```

- So let's go ahead and create our schema file and set up our post object type and query
    1. Create folder graphql in root project
    2. Create file name 'schema.graphql' in folder graphql
    3. Add code post object type and query

```
"A date string with format `Y-m-d`, e.g. `2011-05-23`."
scalar Date @scalar(class: "Nuwave\\Lighthouse\\Schema\\Types\\Scalars\\Date")

"A datetime string with format `Y-m-d H:i:s`, e.g. `2018-05-23 13:43:32`."
scalar DateTime @scalar(class: "Nuwave\\Lighthouse\\Schema\\Types\\Scalars\\DateTime")

type Query {
    posts: [Post!]! @all @orderBy(column: "created_at", direction: "DESC")
    post(id: Int! @eq): Post @find
}

type Post {
    id: ID!
    title: String!,
    description: String!,
    status: Boolean!,
    created_by: User @belongsTo
    updated_by: User @belongsTo
    created_at: DateTime!
    updated_at: DateTime!
}

type Mutation {
    createPost(input: CreatePostInput! @spread): Post @field(resolver: "PostMutator@create")
    updatePost(input: UpdatePostInput! @spread): Post @field(resolver: "PostMutator@update")
    deletePost(input: DeletePostInput! @spread): Post @field(resolver: "PostMutator@delete")
}

input CreatePostInput {
    title: String!,
    description: String!
}
input UpdatePostInput {
    id: Int!,
    title: String!,
    description: String!,
    status: Boolean!
}

input DeletePostInput {
    id: Int!
}
```

```
php artisan serve
```

5. Open browser to Testing Graphql API

```
http://localhost:8000/graphql-playground
```

Install Vue-Apollo in VueJS

- Use command line for apollo client full configuration
  npm install --save vue-apollo graphql apollo-client apollo-link apollo-link-http apollo-cache-inmemory graphql-tag

- After installed apollo client ready
  1. Go to path: resources/js to create folder name “apollo”
  2. Create file name “index.js” in folder apollo
  3. Write code for ApolloClient

```
import Vue from 'vue'
import VueApollo from 'vue-apollo'
import { ApolloClient } from 'apollo-client'
import { HttpLink } from 'apollo-link-http'
import { onError } from "apollo-link-error"
import { InMemoryCache } from 'apollo-cache-inmemory'

const httpLink = new HttpLink({
    uri: process.env.MIX_SOURCE_URL,
})

const defaultOptions = {
  watchQuery: {
      fetchPolicy: "no-cache",
      errorPolicy: "ignore",
  },
  query: {
      fetchPolicy: "no-cache",
      errorPolicy: "ignore",
  },
  mutate: {
    errorPolicy: 'ignore'
  }
};

const errorLink = onError(({ graphQLErrors, networkError }) => {
  if (graphQLErrors)
      graphQLErrors.map(({ message, locations, path }) =>
          console.log(
              `[GraphQL error]: Message: ${message}, Location: ${locations}, Path: ${path}`
          )
      )
  if (networkError) console.log(`[Network error]: ${networkError}`)
})

export const apolloClient = new ApolloClient({
  link: errorLink.concat(httpLink),
  cache: new InMemoryCache(),
  connectToDevTools: true,
  defaultOptions: defaultOptions
})

const apolloProvider = new VueApollo({
  defaultClient: apolloClient
})

Vue.use(VueApollo)

export default apolloProvider
```

4. Install the plugin into Vue

```
import Vue from ‘vue’;
import VueApollo from ‘vue-apollo’;

Vue.use(VueApollo);
```

- The provider holds the Apollo client instances that can then be used by all the child components.

```
const apolloProvider = new VueApollo({
  defaultClient: apolloClient
})
```

- Add it your app with the apolloProvider

```
const app = new Vue({
…
apolloProvider,
…
});
```