

LAB 9 – Programação com o framework VueJS

[parte 2]

O objectivo deste laboratório é repetir a funcionalidade do site desenhado nos LAB5, LAB6, e LAB7¹, mas agora construído com o framework **VueJS**

Assume-se aqui que já realizou com sucesso o LAB8.

¹ NOTA: a funcionalidade "REMEMBER ME" é OPCIONAL: a API `users.php` NÃO está configurada para enviar e receber o cookie "rememberMe"

1. REGISTER

Construa o template adaptando o template `register_template.tpl` realizado no LAB5

```
<template>
<div>

    <Menu/>

    <br/>

    <div v-if="!userLoggedIn" id="register-form" class="container">
        <h1 style="text-align: center">Register</h1>
        <form @submit.prevent="handleSubmit">
            .
            .
            .

            <button @click="cancel()" class="btn btn-warning">Cancel</button>
            <p style="float:right;">
            <button type="submit" class="btn btn-primary">Register</button>
            </p>

        </form>
    </div>
    <div v-else>
        <h3 style="text-align: center;">Logout first </h3>
    </div>

    <Footer />
</div>
</template>
```

Realize o código do controlador

```
<script>

import Footer from '@/components/Footer.vue'
import Menu from '@/components/Menu.vue'

export default {
  components: {
    Footer,
    Menu,
  },
  data() {
    return {
      user: {
        name: '',
        email: '',
        password: '',
      },
      passwordConfirmation: '',
      submitting: true,
      error: true,
    }
  },
  methods: {
```

```

    },
    computed: {
    },
    directives: {
    },
    created() {
    }
  }
}
</script>

```

O controlador da acção valida os dados introduzidos.

Adicione o código necessário para:

- Em caso de insucesso
 - o formulário apresenta os campos correctos já preenchidos e o motivo do insucesso no registo
- Em caso de sucesso
 - o controlador utiliza as acções
`this.$store.dispatch('user/userExists',user)` para verificar se o utilizador já existe na base de dados e, se não existir,
`this.$store.dispatch('user/addUser')` para inserir na base de dados
 - o controlador redireciona para a "vista" `Message.vue`, que mostra a mensagem "Welcome! You can now login". A "vista" `Message.vue` redireciona automaticamente para a página de entrada do site passados 2 segundos

O template e o controlador devem ser colocados no ficheiro

`C:\XAMPP\htdocs\LAB8_10\src\views\Register.vue`

Faça o mapeamento deste controlador no ficheiro

`C:\XAMPP\htdocs\LAB8_10\src\router\index.js`

```

import Vue from 'vue'
import VueRouter from 'vue-router'
import Register from '../views/Register.vue'

Vue.use(VueRouter)

const routes = [
  {
    path: '/register',
    component: Register
  },
]

```

2. LOGIN

Adapte o template `login_template.tpl` realizado no LAB6 para construir o template de login

```
<template>
<div>

  <Menu/>

  <br/>

  <div v-if="!userLoggedIn" id="login-form" class="container">
    <h1 style="text-align: center">Login</h1>
    <form @submit.prevent="handleSubmit">
      .
      .
      .
      <button @click="cancel()" class="btn btn-warning">Cancel</button>
      <p style="float:right;">
        <button type="submit" class="btn btn-primary">Login</button>
      </p>
    </form>
  </div>
  <div v-else>
    <h3 style="text-align: center;">Logout first </h3>
  </div>

  <Footer />

</div>
</template>
```

Realize o controlador responsável pela página de login

```
<script>

import Footer from '@/components/Footer.vue'
import Menu from '@/components/Menu.vue'

export default {
  components: {
    Footer,
    Menu,
  },
  data() {
    return {
      user: {
        email: '',
        password: '',
      },
      submitting: false,
      error: false,
    }
  },
  methods: {

  },
  computed: {

  },
  directives: {
```

```

    },
    created() {
    }
  }
</script>

```

Adicione o código necessário para:

- Utilizar a "acção" `this.$store.dispatch('user/loginUser', user)` e, em caso de sucesso, guardar na "store" no objecto "user" o id, nome, email e session_id do utilizador validado
- Em caso de sucesso utilizar a "vista" `Message.vue`, com a mensagem `"Welcome back " + this.$store.getters['user/getUser'].name + "!"`
- Em caso de insucesso no formulário de login é apresentada a mensagem de erro "Login failed: wrong email or password".

O template e o controlador deverão encontrar-se em

`C:\XAMPP\htdocs\LAB8_10\src\views>Login.vue`

Faça o mapeamento deste controlador no ficheiro

`C:\XAMPP\htdocs\LAB8_10\src\router\index.js`

```

import Vue from 'vue'
import VueRouter from 'vue-router'
import Login from '../views/Login.vue'

Vue.use(VueRouter)

const routes = [
  {
    path: '/login',
    component: Login
  },
]

```

3. POST

Adapte o template `blog_template.tpl` realizado no LAB7 para realizar o template em VueJS:

```
<template>
<div>
<Menu />
<div id="post-form" class="container">
  <div v-if="!userLoggedIn" >
    <h3 style="text-align: center;">Login first </h3>
  </div>
  <div v-else>
    <h1 style="text-align: center;">New Post</h1><br><br>
    <form @submit.prevent="handleSubmit">
      <div class="form-group">
        <textarea
          v-model="post.content"
        >
      </textarea>
    </div>
    .
    .
    .
    <p style="float:left;">
      <button @click="cancel()" class="btn btn-warning">Cancel</button>
    </p>
    <p style="float:right;">
      <button type="submit" class="btn btn-primary">Add Post</button>
    </p>
  </form>
</div>
</div>
<Footer />
</div>
</template>
```

Realize o código do controlador responsável pelo novo post

```
<script>
import Footer from '@components/Footer.vue'
import Menu from '@components/Menu.vue'

export default {
  components: {
    Footer,
    Menu
  },
  data() {
    return {
      submitting: false,
      error: false,
      post: {
        content: '',
      },
      user: {
        id: '',
        name: '',
        email: '',
        session_id: ''
      }
    }
  }
}
```

```

    },
  },
  created: function () {

  },
  methods: {

  },
  computed: {
  },
  directives: {

  },
}
</script>

```

Adicione o código necessário para:

- Utilizar a "acção" `this.$store.dispatch('microposts/addMicropost', data)` e, em caso de sucesso, guardar na base de dados o novo post
- Em caso de sucesso utilizar a "vista" `Message.vue`, com a mensagem "Success: Post added"

O template e o controlador deverão encontrar-se em

`C:\XAMPP\htdocs\LAB8_10\src\views\Post.vue`

Faça o mapeamento deste controlador no ficheiro

`C:\XAMPP\htdocs\LAB8_10\src\router\index.js`

```

import Vue from 'vue'
import VueRouter from 'vue-router'
import Post from '../views/Post.vue'

Vue.use(VueRouter)

const routes = [
  {
    path: '/post',
    component: Post
  },
]

```

NOTA: apenas utilizadores que fizeram login podem inserir posts! o seu código tem de ser robusto contra tentativas de inserir posts em nome de outro utilizador!

4. UPDATE POST

Adapte o template `blog_template.tpl` realizado no LAB7 para realizar o template em VueJS:

```
<template>
<div>
<Menu />
<div id="post-form" class="container">
  <div v-if="!userLoggedIn" >
    <h3 style="text-align: center;">Login first </h3>
  </div>
  <div v-else>
    <h1 style="text-align: center;">Update Post</h1><br><br>
    <form @submit.prevent="handleSubmit">
      <div class="form-group">
        <textarea
          v-model="post.content"
        >
      </textarea>
    </div>
    .
    .
    .
    <p style="float:left;">
      <button @click="cancel()" class="btn btn-warning">Cancel</button>
    </p>
    <p style="float:right;">
      <button type="submit" class="btn btn-primary">Add Post</button>
    </p>
  </form>
</div>
</div>
<Footer />
</div>
</template>
```

Realize o código do controlador responsável por actualizar o post

```
<script>
import Footer from '@/components/Footer.vue'
import Menu from '@/components/Menu.vue'
export default {
  components: {
    Footer,
    Menu
  },
  data() {
    return {
      submitting: false,
      error: false,
      post: {
        id:"",
        content:"",
        user_id:"",
        author:"",
        created_at:"",
        updated_at:"",
        likes:"",
      },
      user: {
        id: '',
        name: '',
      }
    }
  }
}
```



```

        email: '',
        session_id: ''
      },
    },
  },
  created: function () {
  },
  methods: {
  },

  computed: {
  },
  directives: {
  },
}
</script>

```

Adicione o código necessário para:

- Utilizar a "acção" `this.$store.dispatch('microposts/updateMicropost', data)` e, em caso de sucesso, guardar na base de dados o novo conteúdo do post
- Em caso de sucesso utilizar a "vista" `Message.vue`, com a mensagem "Success: Post updated"

O template e o controlador deverão encontrar-se em

`C:\XAMPP\htdocs\LAB8_10\src\views\UpdatePost.vue`

Faça o mapeamento deste controlador no ficheiro

`C:\XAMPP\htdocs\LAB8_10\src\router\index.js`

```

import Vue from 'vue'
import VueRouter from 'vue-router'
import UpdatePost from '../views/UpdatePost.vue'

Vue.use(VueRouter)

const routes = [
  {
    path: '/updatePost/:id',
    component: UpdatePost
  },
]

```

NOTA: o seu código tem de ser robusto contra tentativas de alterar posts que pertencem a outro utilizador!

5. LOGOUT

Construa o controlador que destroi a sessão quando o utilizador faz logout. O controlador utiliza a vista `'Message.vue'` (sugere-se “See you back soon!”), que ao fim de 2 segundos redirecciona para a página de rosto.

6. UPLOAD

Considere o lab concluído quando obtiver a mesma funcionalidade que foi requerida no LAB7, mas agora realizada em VueJS

(<http://daw.deei.fct.ualg.pt/~a555550/vue-app.forum/dist/>)

Atualize o “publicPath” no ficheiro

C:\XAMPP\htdocs\LAB8_10\vue.config.js

com o conteúdo

```
// vue.config.js
module.exports = {
  publicPath: '/~a12345/LAB8_10/dist/'
}
```

(substitua '12345' pelo seu nº de aluno!)

OPCIONAL: se utilizou a sua base de dados local, nos ficheiros "comments.js", "microposts.js", "users.js" dentro da pasta "src\store" atualize o URL

para `'http://localhost/LAB8_10/api/'`
para `'http://daw.deei.fct.ualg.pt/~a12345/LAB8_10/api/'`

(substitua '12345' pelo seu nº de aluno!)

Execute o comando

C:\XAMPP\htdocs\LAB8_10> npm run build

Faça o upload com WinSCP/FileZilla/scp das pastas

- "public"
- "src"
- "dist"

e dos ficheiros

- package.json
- vue.config

para a pasta “LAB8_10” no seu site web pessoal no servidor de produção

Teste o funcionamento do site no URL

http://daw.deei.fct.ualg.pt/~a12345/LAB8_10/dist

(substitua '12345' pelo seu nº de aluno...)

REFERÊNCIAS

- <https://vuejs.org/>
- <http://intranet.deei.fct.ualg.pt/IPM/labVueJS>
- http://intranet.deei.fct.ualg.pt/~a555550/LAB8_10/api/
- http://intranet.deei.fct.ualg.pt/~a555550/LAB8_10/api/api.html

ANEXO 1. Estrutura da base de dados

A estrutura da base de dados pode ser consultada em

<http://daw.deei.fct.ualg.pt/phpMyAdmin>

```
CREATE TABLE `users` (  
  `id` int(11) NOT NULL auto_increment,  
  `name` varchar(255) default NULL,  
  `email` varchar(255) default NULL,  
  `created_at` datetime NOT NULL,  
  `updated_at` datetime NOT NULL,  
  `password_digest` varchar(255) default NULL,  
  `remember_digest` varchar(255) default NULL,  
  `admin` tinyint(1) default NULL,  
  `activation_digest` varchar(255) default NULL,  
  `activated` tinyint(1) default NULL,  
  `activated_at` datetime default NULL,  
  `reset_digest` varchar(255) default NULL,  
  `reset_sent_at` datetime default NULL,  
  PRIMARY KEY (`id`),  
  UNIQUE KEY (`email`)  
) ENGINE=InnoDB DEFAULT CHARSET=utf8;  
  
CREATE TABLE `microposts` (  
  `id` int(11) NOT NULL auto_increment,  
  `content` text,  
  `user_id` int(11) default NULL,  
  `created_at` datetime NOT NULL,  
  `updated_at` datetime NOT NULL,  
  PRIMARY KEY (`id`),  
  CONSTRAINT FOREIGN KEY (`user_id`) REFERENCES `users` (`id`)  
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```