

LAB 10 – Programação com o framework VueJS

[parte 3]

Assume-se aqui que já realizou com sucesso o LAB9.

0. CRIAÇÃO DA TABELA COMMENTS

- no servidor de produção

Faça login no servidor de produção daw2 (IP 10.10.23.184) e adicione a tabela “comments” à sua base de dados¹

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

```
a12345@daw2:~$mysql -u a12345 -p -h 10.10.23.184 db_a12345
mysql> CREATE TABLE IF NOT EXISTS `comments` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `content` text CHARACTER SET utf8 COLLATE utf8_bin,
  `user_id` int(11) DEFAULT NULL,
  `micropost_id` int(11) DEFAULT NULL,
  `created_at` datetime NOT NULL,
  PRIMARY KEY (`id`),
  CONSTRAINT FOREIGN KEY (`user_id`) REFERENCES `users` (`id`),
  CONSTRAINT FOREIGN KEY (`micropost_id`) REFERENCES `microposts` (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8
```

Nota: se desejar pode inserir o comando no tab "SQL" na interface web em

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

¹ OPCIONAL: se está a utilizar uma cópia da base de dados no seu PC/portátil, utilize a interface web em <http://localhost/phpmyadmin/> para repetir a operação

1. MICROPOSTS - ATUALIZAÇÃO

Atualize o template que foi realizado no LAB8, de forma a ter

1. o texto “update post” se o utilizador que fez *log in* é o autor do post (já existente)
2. o texto “comment post” se o post *não pertence* ao utilizador (novidade)
3. um "link/botão" "show comments" no final do conteúdo do post

```
<template >
<div>
<Menu />
<!-- BEGIN MICROPOSTS -->
.
.
.
      <div class="panel-body">
        updated: {{micropost.updated_at}} <br>
        created: {{micropost.created_at}} <br>
        <div v-if=" userLoggedIn && user.id == micropost.user_id">
          <router-link :to="'/updatePost/' +
micropost.id">update post</router-link>
        </div>
        <div v-if=" userLoggedIn && user.id != micropost.user_id">
          <router-link :to="'/commentPost/' +
micropost.id">comment post</router-link>
        </div>
      </div>
.
.
.
      <div v-if="micropost.id!=show" >
        <button @click="showComments(micropost.id)" type="button"
class="btn btn-link">Show Comments</button>
      </div>
.
.
.
<!-- END MICROPOSTS -->
<Footer />
</div>
</template>
```

Atualize o código do controlador para utilizar

- a acção `this.commentsStore.getCommentsDB()` para fazer o download de todos os comentários existentes na base de dados,
- o "getter" `this.commentsStore.getPostComments(id)` para seleccionar os comentários de um post específico

Atualize o código do controlador para, quando o utilizador clicar no link "show comments" de um post específico, mostrar os comentários deste post imediatamente a seguir ao post.

Nota: se o utilizador clicar no link "show comments" de um outro post, os comentários do post anterior devem ficar escondidos

2. COMMENT

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;">Comment</h1><br><br>
    <form @submit.prevent="handleSubmit">
    <div class="form-group">
      <textarea
        v-model="comment.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'

import { useCommentsStore } from '@store/comments'
import { useUserStore } from '@store/user'

export default {
  setup() {
    const userStore = useUserStore()
    const commentsStore = useCommentsStore()
    return { userStore, commentsStore }
  },
  components: {
    Footer,
    Menu
  },
  data() {
    return {
      submitting: false,
      error: false,
      comment: {
        content: '',
      },
    },
  },
}
```

```

        user: {
          id: '',
          name: '',
          email: '',
          session_id: ''
        },
      },
    },
  },
  created() {
  },
  methods: {
  },
  computed: {
  },
  directives: {
  },
}
</script>

```

Adicione o código necessário para: utilizar a "acção"

```

this.commentsStore.addCommentDB(data)

```

- em caso de sucesso, guardar na base de dados o novo comment
- em caso de sucesso utilizar a "vista" `Message.vue`, com a mensagem "Success: Comment added"

O template e o controlador deverão encontrar-se em

```

C:\XAMPP\htdocs\LAB8_10\src\views\Comment.vue

```

Faça o mapeamento deste controlador no ficheiro

```

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

```

```

import Comment from '../views/Comment.vue'

```

```

const routes = [
  {
    path: '/comment/:id',
    component: Comment
  },
]

```

NOTA: O utilizador não pode introduzir comentários sobre o seu post (mas pode actualiza-lo...). Apenas utilizadores que fizeram login podem inserir comentários sobre posts!

O resultado deve ser o seguinte (ou equivalente):

- “Comentários” escondidos:



The screenshot shows a forum interface with a blue header containing 'home', 'logout', 'post blog', and 'Welcome Jose Bastos'. Below the header, a post by 'Jose Bastos' is shown with the text 'O portal do forum de DAW tem agora uma nova funcionalidade: post replies!' and a 'Show' button. A comment by 'Tester' with the text 'testing .' is visible below the post, with a 'Show' button. The comment has 1 like. The interface uses a blue and white color scheme.

- “Comentários” visíveis



The screenshot shows the same forum interface as above, but with the comment by 'Tester' visible. The comment has 1 like and a 'Hide' button. Below the comment, a yellow bar indicates 'Jose Bastos posts: 4' with a list of posts, including one with the text 'concordo! a testar é que se aprende...' and a timestamp '2017-11-17 17:56:47'. The interface uses a blue and white color scheme.

- “Comentários” escondidos:

<p>author: Jose Bastos</p> <p>updated: 2019-10-29 11:27:12 created: 2017-11-11 15:36:07</p>	<p>Mais um teste 456</p> <p>Show Comments</p>
---	---

- “Comentários” visíveis

<p>author: Jose Bastos</p> <p>updated: 2019-10-29 11:27:12 created: 2017-11-11 15:36:07</p>	<p>Mais um teste 456</p> <hr/> <p>author: abc date: 2022-10-19 13:40:57 test test test</p> <hr/> <p>author: abc date: 2022-10-19 13:41:24 mais um comentário coment?o</p>
---	---

3. UPLOAD

Considere o lab concluído quando obtiver a mesma funcionalidade dos exemplos

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

http://daw.deei.fct.ualg.pt/~a999993/LV_exam2/blog

Atualize a URL “base” no ficheiro `vite.config.js`

```
// vite.config.js
base: '/~a12345/LAB8_10/dist/',
```

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

Execute o comando²

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

Faça o upload com scp/WinSCP/FileZilla das pastas

- "src"
- "dist"

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

² OPCIONAL: apenas se utilizou a sua base de dados local, nos ficheiros "comments.js", "microposts.js", "users.js" dentro da pasta "src\store" reponha URL de

```
'http://localhost/LAB8_10/api/'
```

para

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

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

antes de fazer o "build" !!

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 `index_users_on_email` (`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;  
  
CREATE TABLE `comments` (  
  `id` int(11) NOT NULL AUTO_INCREMENT,  
  `content` text CHARACTER SET utf8 COLLATE utf8_bin,  
  `user_id` int(11) DEFAULT NULL,  
  `micropost_id` int(11) DEFAULT NULL,  
  `created_at` datetime NOT NULL,  
  PRIMARY KEY (`id`),  
  CONSTRAINT FOREIGN KEY (`user_id`) REFERENCES `users`  
(`id`),  
  CONSTRAINT FOREIGN KEY (`micropost_id`) REFERENCES  
`microposts` (`id`)  
) ENGINE=InnoDB DEFAULT CHARSET=utf8
```