GRADUAÇÃO EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

TP1 da disciplina Frameworks Front-end e conexão com Back-end apresentada ao curso de Análise e Desenvolvimento de Sistemas, do Instituto INFNET.

Aluno: Camila Nunes de Castro Cerqueira

Turma: GRLADS01C2-M1-L1

Professor: Vanessa Cristina Martins da Silva Frattini

16/02/2022

Sumário

TP1 – < Frameworks Front-end e conexão com Back-end >

Questão 1.a

O MVVM é um pattern design voltado para user interface, pois permite a separação da camada view de todas outras camadas. Esse padrão possui três camadas: Model, View e View Model. A camada View é a responsável por exibir as informações ao usuário, a model é responsável pela lógica de negócios e também como repositório para uma subcamada (acesso para Apis, por exemplo). Já o View Model é a responsável por carregar a lógica da aplicação de interface ao usuário, manter e observar os estados e ativar os eventos da View. A grande diferença desse modelo para os demais é a presença do View-Model que através do processo de desacoplamento facilita a alteração de classes, métodos e componentes, e assim permite a vinculação dos dados de forma reativa para visualização do usuário. Podemos ressaltar também que em aplicações pequenas é possível utilizar o View Model como um ponto de integração para acesso em bancos de dados e apis e assim todos os dados necessários serão encapsulados nessa mesma camada.

Questão 1.b

```
<script>
        Vue.createApp({
            data(){
                return{
                    items: []
                }
            },
            created(){
                fetch("/data.json")
                .then(resultado => resultado.json())
                .then(data =>{
                    this.items = data.items
                })

            }
        }).mount("#app")
    </script>
```

Questão 2

A principal diferença entre o MVC e MVVM é a presença da camada View Model no MVVM o que torna a aplicação mais organizada, ordenada e com facilidade no gerenciamento e apresentação das informações para o usuário. Ou seja, facilita a separação do desenvolvimento das interfaces gráficas e assim responde mais rápido as ações do usuário.

Questão 3

Para uma aplicação ser considerada reativa é necessário que ela consiga manipular um fluxo de dados assíncronos. Desse modo é imprescindível que ela tenha a capacidade de observar quando algum estado for modificado no objeto e que consiga reagir com essas mudanças imediatamente para que seja renderizado na tela do usuário naquele exato momento, ou seja, é possível que ele veja essas modificações em tempo real sem a necessidade de uma atualização de página.

Questão 4.a

V-if. A diretiva v-if atua como um controlador de fluxo de acordo com a condição ali imposta, desse modo o bloco somente será renderizado se aquela condição for verdadeira.

Questão 4.b

A diretiva v-show também consegue mostrar ou não um elemento, no entanto a sua lógica é diferente da v-if. O v-show, independentemente da condição, sempre será renderizado e estará presente no DOM, no entanto não aparece para o usuário pois altera a propriedade de estilo (CSS) do elemento

Questão 5

Através de interpolação utilizando a sintaxe “Mustache”: {{ variável }}.

Questão 6.a

Utilizamos o DOM para manipular os elementos presente nele, porém a manipulação em grande escala nos leva a problemas de performance e constantes atualizações, no entanto com as bibliotecas é possível usar o Virtual Dom que representa o DOM do browser e no permite alterar os elementos de uma forma mais rápida, pois não precisamos acessar diretamente o DOM Puro, ou seja, realizamos as modificações no Dom Virtual que por sua vez será renderizado e assim atualiza o DOM do browser em tempo real.

Questão 6.b

O loop de eventos é a etapa onde o Vue monitora os dados e eventos de acordo com as ações do usuário para realizar uma ação de acordo com o que foi definido.

Questão 7

A diretiva v-for é utilizada para a iteração de elementos de arrays e de propriedades de objetos que se encontram na parte do Vue.

Questão 8

Os filtros foram removidos do Vue 3.0. Segue exemplo na versão 2 com CDN.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <script src="https://cdn.jsdelivr.net/npm/vue@2.6.14"></script>
    <title>Exercício 8 - TP 1</title>
</head>
<body>
    <div id="app">
          <h1>{{ titulo | uppercase }} </h1>
      </div>
```

```
    <script>
        new Vue({
        el: "#app",
        data: {
            titulo: "exercício 8 - tp 1",

  },filters: {
            uppercase: function(str) {
            return str.toUpperCase();
        }
  }
});
    </script>
</body>
</html>
```

Questão 9

Pois é a única diretiva que renderiza conteúdos HTML, as demais tratam o conteúdo HTML como texto, logo é necessário ter certeza que o conteúdo utilizado no v-html é confiável para não injetar conteúdos que colocam em cheque a segurança da aplicação.

Questão 10

Index em anexo no rar.

Segue texto do código com CDN (Vue 3)

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <script src="https://unpkg.com/vue@next"></script>
    <title>Exercício 10 - TP 1</title>
```

```
    <style>
        .realizado {
            color: red;
            text-decoration: line-through;
        }
    </style>
</head>
<body>
    <div id="app">
    <ul>
        <li v-if="!tarefas.length">Lista Vazia</li>
        <li v-for="tarefa in tarefas" :class=" {'realizado':tarefa.estaCompleta}">
            {{tarefa.descricao}}
        </li>
    </ul>
</div>
    <script>
        Vue.createApp({
            data(){
                return{
                    tarefas: [
                        { id: 1 , descricao: "Realizar exercício 1", estaCompleta:
false},
                        { id: 2 , descricao: "Realizar exercício 2", estaCompleta:
true},
                        { id: 3 , descricao: "Realizar exercício 3", estaCompleta:
false},
                        { id: 4 , descricao: "Realizar exercício 4", estaCompleta:
false},
                        { id: 5 , descricao: "Realizar exercício 5", estaCompleta:
true},
                        { id: 6 , descricao: "Realizar exercício 6", estaCompleta:
false},
                        { id: 7 , descricao: "Realizar exercício 7", estaCompleta:
true},
                    ]
                }
            }

        }).mount("#app")
    </script>
</body>
</html>
```