



Unioeste - Universidade Estadual do Oeste do Paraná
CENTRO DE CIÊNCIAS EXATAS E TECNOLÓGICAS
Colegiado de Ciência da Computação
Curso de Bacharelado em Ciência da Computação

ATIVIDADE AVALIATIVA 04 – RELATORIO

Aluno: Luiz Eduardo Garzon de Oliveira

Docente: Leonardo Medeiros

Cascavel, agosto de 2023

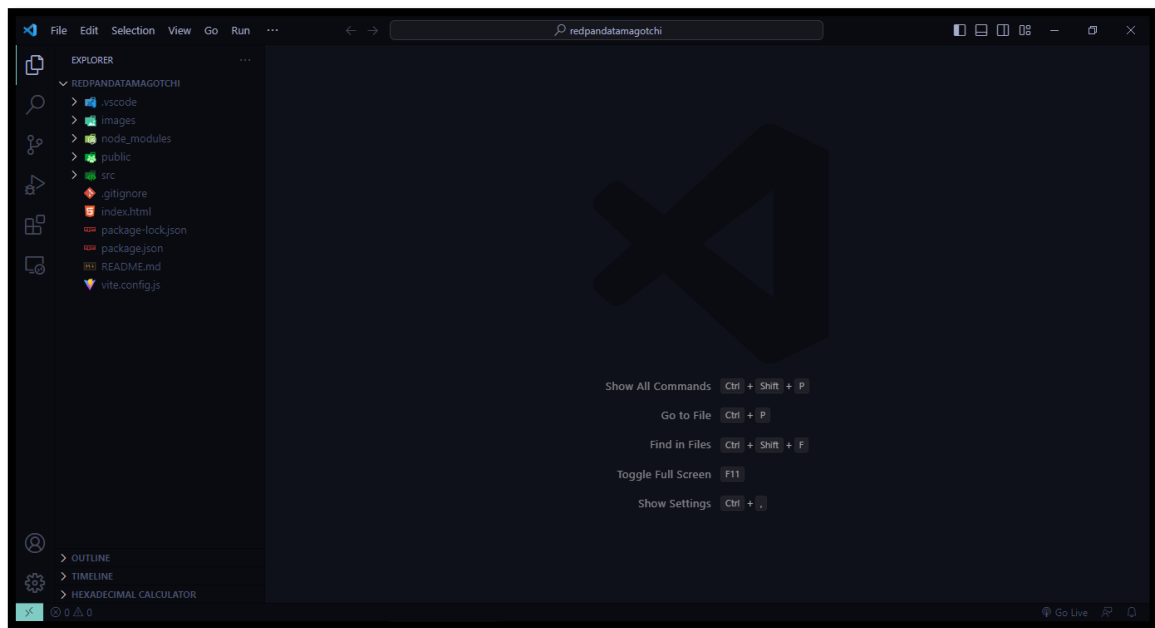
Descrição do projeto:

Para a avaliação 04 da disciplina TDS (Tecnologia para Desenvolvimento de Sistemas) foi escolhido a implementação de um Tamagotchi (bichinho virtual) utilizando os requisitos solicitados no documento de apresentação do trabalho requerido.

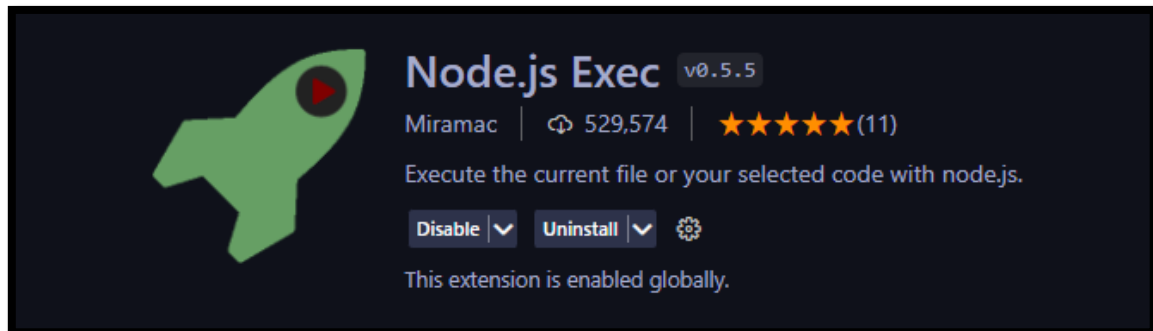
Parte 1:

Configuração do ambiente de trabalho e instalação das ferramentas necessárias:

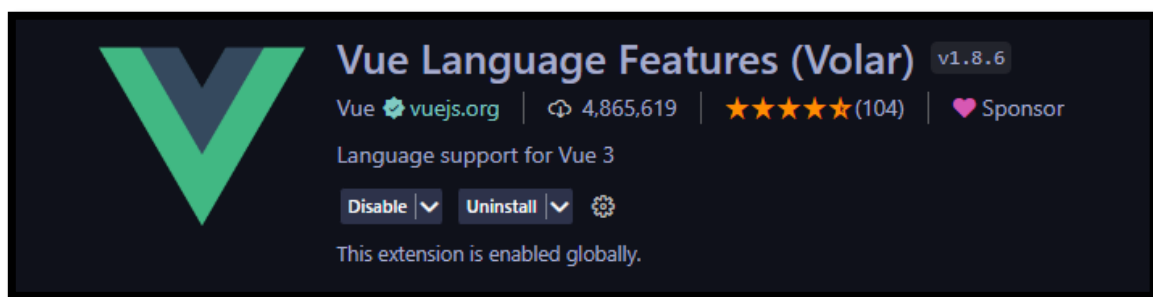
Configuração do ambiente de trabalho: VSCODE foi escolhido como ambiente de desenvolvimento no Windows 10 Home.



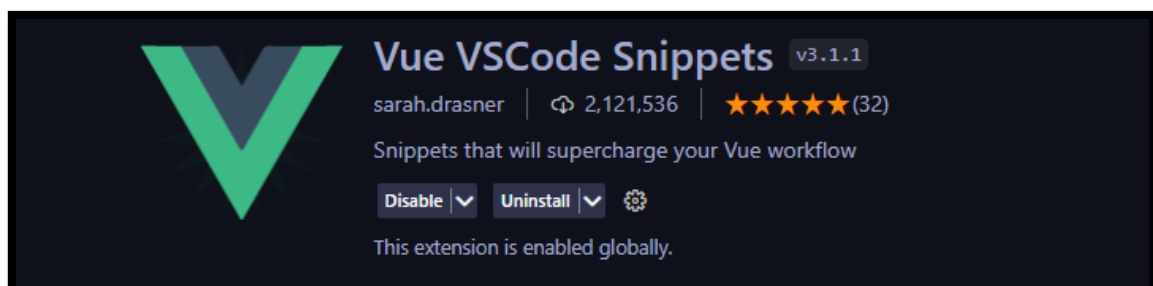
Extensões utilizadas para o desenvolvimento: Node.js Exec.: Para run no código atual com node.js. **Vue Language Features:** extensão de suporte para Vue, Vitepress, petite-vue implementar o desempenho de nível de serviço da linguagem TypeScript nativa. **Vue VSCode Snippets:** snippets para sobrecarregar um fluxo de trabalho. Concentra-se na ergonomia do desenvolvedor a partir do ponto de uso do Vue.



Node.js Exec v0.5.5
Miramac | 529,574 | ★★★★★ (11)
Execute the current file or your selected code with node.js.
Disable | Uninstall | ⚙️
This extension is enabled globally.



Vue Language Features (Volar) v1.8.6
Vue vuejs.org | 4,865,619 | ★★★★★ (104) | ❤️ Sponsor
Language support for Vue 3
Disable | Uninstall | ⚙️
This extension is enabled globally.



Vue VSCode Snippets v3.1.1
sarah.drasner | 2,121,536 | ★★★★★ (32)
Snippets that will supercharge your Vue workflow
Disable | Uninstall | ⚙️
This extension is enabled globally.

Criação da aplicação Vue.js: Instalação e configuração do ambiente para o desenvolvimento em Vue.js feita a partir do site oficial do framework: [Quick Start | Vue.js \(vuejs.org\)](https://vuejs.org/guide/quick-start.html)

Make sure you have an up-to-date version of **Node.js** installed and your current working directory is the one where you intend to create a project. Run the following command in your command line (without the `>` sign):

```
> npm init vue@latest
```

This command will install and execute **create-vue**, the official Vue project scaffolding tool. You will be presented with prompts for several optional features such as TypeScript and testing support:

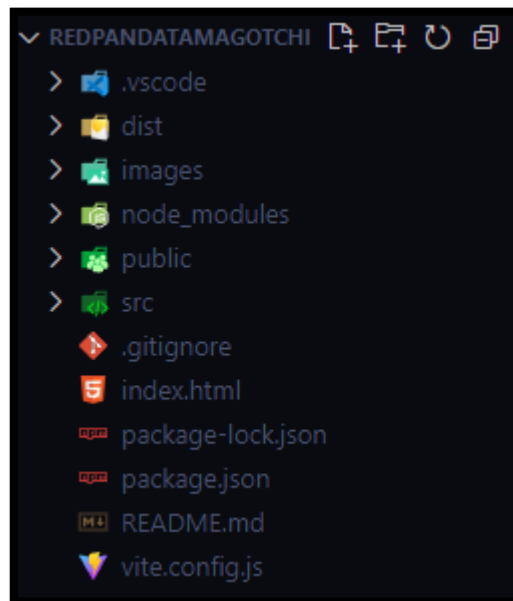
```
✓ Project name: ... <your-project-name>
✓ Add TypeScript? ... No / Yes
✓ Add JSX Support? ... No / Yes
✓ Add Vue Router for Single Page Application development? ... No / Yes
✓ Add Pinia for state management? ... No / Yes
✓ Add Vitest for Unit testing? ... No / Yes
✓ Add an End-to-End Testing Solution? ... No / Cypress / Playwright
✓ Add ESLint for code quality? ... No / Yes
✓ Add Prettier for code formatting? ... No / Yes
```

```
Scaffolding project in ./<your-project-name>...
Done.
```

If you are unsure about an option, simply choose **No** by hitting enter for now. Once the project is created, follow the instructions to install dependencies and start the dev server:

```
> cd <your-project-name>
> npm install
> npm run dev
```

Ambiente Final: Feito esses passos o ambiente VSCODE deve ficar como a imagem abaixo.



A seguir os principais diretórios comuns usados para organizar o código-fonte e os recursos da aplicação são estabelecidos:

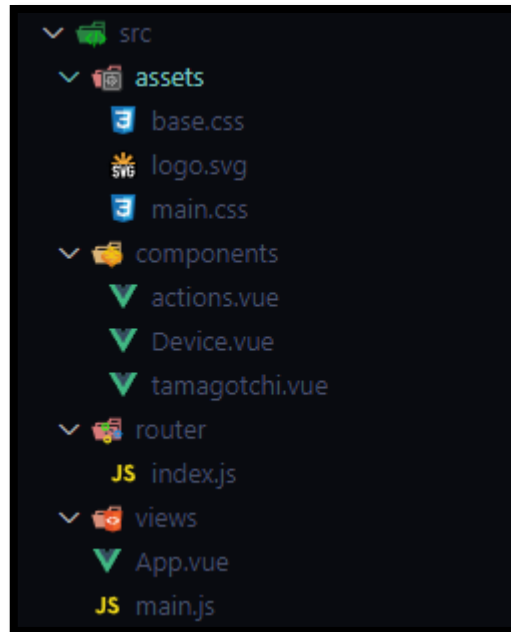
src: O diretório "src" é o diretório principal onde todo o código-fonte da sua aplicação Vue.js é colocado. Ele contém todos os componentes, arquivos JavaScript, CSS, imagens e outros recursos necessários para a construção da aplicação.

assets: O diretório "assets" é usado para armazenar recursos estáticos, como imagens, fontes, arquivos CSS não processados, arquivos de dados ou qualquer outro recurso que não precise ser transformado ou compilado.

components: O diretório "components" é usado para armazenar componentes reutilizáveis da aplicação. Componentes no Vue.js são blocos de construção que encapsulam HTML, JavaScript e CSS, tornando-os fáceis de reutilizar em diferentes partes da aplicação.

router: O diretório "router" é usado quando se trabalha com Vue Router. O Vue Router é um roteador oficial para aplicações Vue.js que permite navegar entre diferentes páginas ou "rotas" sem a necessidade de recarregar a página. Neste diretório, você encontrará os arquivos relacionados à configuração do roteador da aplicação, como definir as rotas e suas respectivas configurações.

views: O diretório "views" é usado para armazenar componentes que representam as diferentes visualizações (ou páginas) da sua aplicação. Normalmente, cada rota definida no Vue Router corresponderia a uma vista dentro deste diretório.



Passo 2:

Desenvolvimento da aplicação Tamagotchi para vue.js. A seguir a explicação das principais partes do código:

Index.html: Contem a instancia html principal da aplicação. Linkagem do framework bootstrap.

O **header** define uma barra de navegação na página que exibe o texto "Tamagotchi" e um botão "Sobre" que quando clicado realiza um smoothscroll para o final da página na segunda section onde exibe informações extras sobre a história do tamagotchi e, ao clicar novamente no botão, agora em "voltar", ele retorna para o início da página.

O **body** contém todo o conteúdo principal, a **primeira section** contém a aplicação vue.js onde instancia tudo o que foi desenvolvido, no caso, toda a aplicação tamagotchi menos o header. A **segunda section** apenas exibe a informação extra como dito acima.

O **primeiro script main.js** vem como base do projeto em sua criação e não foi alterado. O **segundo script** configura a página para realização do smoothscroll e logica da mudança do botão quando é clicado para subir e para descer a página. O **terceiro script** realiza a importação do script para bootstrap.

```
index.html X
index.html > html > body > script
1 <!DOCTYPE html>
2 <html lang="en">
3
4 <head>
5   <meta charset="UTF-8">
6   <link rel="icon" href="/favicon.ico">
7   <!-- Link para o Bootstrap-->
8   <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css"
9     rel="stylesheet"
10     integrity="sha384-9ndCyUaIbzAi2FUVXJi0CjmCapSm07SnpJef0486qhLnuZ2cdeRh002iuK6FUUVM"
11     crossorigin="anonymous">
12   <meta name="viewport" content="width=device-width, initial-scale=1.0">
13   <title>TAMAGOTCHI VUE</title>
14 </head>
15
16 <!--Header para definicao da Nav-->
17 <header class="header-sobre" class="header-voltar">
18   <h1>TAMAGOTCHI</h1>
19   <nav>
20     <!-- Adicionamos a classe toggle-link ao link "Sobre" -->
21     <a href="#page2" class="toggle-link">Sobre</a>
22   </nav>
23 </header>
24
25 <!--Definicao do body que contem o conteudo principal da pagina-->
26 <body>
27   <!--Conteudo principal-->
28   <main style="padding-top: 10px;">
29     <!--Conteudo da aplicacao Vue.js-->
30     <section id="app"></section>
31     <!--Conteudo da segunda section html-->
32     <section id="page2">
33       <h1>Breve historia sobre Tamagotchi</h1>
34       <div>...
35     </div>
36     </section>
37   </main>
38
39 <!--JS base da aplicacao padrao-->
40 <script type="module" src="/src/main.js"></script>
```



```

68 <!--JS para configuracao do controle da pagina como smoothscroll-->
69 <script>
70     let num = 0;
71     // Função para rolagem suave ao clicar em links de âncoras
72     function smoothScroll(target) {
73         var targetElement = document.querySelector(target);
74         var targetPosition = targetElement.getBoundingClientRect().top;
75         var startPosition = window.pageYOffset;
76         var distance = targetPosition - startPosition;
77         var duration = 800; // Velocidade de rolagem em milissegundos (ajuste conforme necessário)
78         var startTime = null;
79
80         function animation(currentTime) {
81             if (startTime === null) startTime = currentTime;
82             var timeElapsed = currentTime - startTime;
83             var run = ease(timeElapsed, startPosition, distance, duration);
84             window.scrollTo(0, run);
85             if (timeElapsed < duration) requestAnimationFrame(animation);
86         }
87
88         // Função de interpolação para suavizar a animação (pode usar outras fórmulas também)
89         function ease(t, b, c, d) {
90             t /= d / 2;
91             if (t < 1) return c / 2 * t * t + b;
92             t--;
93             return -c / 2 * (t * (t - 2) - 1) + b;
94         }

```

```

95
96         requestAnimationFrame(animation);
97     }
98
99     // Captura o link "Sobre" e adiciona um evento de clique para rolagem suave
100     var linkSobre = document.querySelector('.toggle-link'); // Alteramos o seletor para .
    toggle-link
101     linkSobre.addEventListener('click', function (e) {
102         e.preventDefault();
103         var target = this.getAttribute('href');
104         smoothScroll(target);
105         if (num == 0) {
106             this.textContent = 'Voltar';
107             document.querySelector('header').classList.remove('header-sobre');
108             document.querySelector('header').classList.add('header-voltar');
109             num = 1;
110         } else {
111             this.textContent = 'Sobre';
112             document.querySelector('header').classList.remove('header-voltar');
113             document.querySelector('header').classList.add('header-sobre');
114             num = 0;
115         }
116     });
117

```

```
118 // Captura todos os links de âncoras (que levam a seções) e adiciona um evento de clique para
    // rolagem suave
119 document.querySelectorAll('a[href^="#"]').forEach(anchor => {
120     anchor.addEventListener('click', function (e) {
121         e.preventDefault();
122         var target = this.getAttribute('href');
123         smoothScroll(target);
124     });
125 });
126
127 // Evento de carregamento para garantir que a rolagem suave funcione corretamente após o
    // carregamento da página
128 window.addEventListener('load', function () {
129     // Recarrega a página no topo
130     window.scrollTo(0, 0);
131 });
132
133 // Evento onbeforeunload para garantir que a página seja recarregada no topo ao dar um reload
134 window.onbeforeunload = function () {
135     window.scrollTo(0, 0);
136 };
137 </script>
138
139 <!--Link Bootstrap-->
140 > <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootstrap.bundle.min.js" ...
141
142 </script>
143 </body>
```

App.vue: arquivo principal onde é instanciado toda a aplicação vue.js

Importamos os componentes Device e Tamagotchi, o device é a imagem do dispositivo que simula o dispositivo tamagotchi e o componente tamagotchi é o bichinho virtual em si que altera seus .svg conforme os humores que recebe do componente actions. Enquanto não for definido um nome para o tamagotchi a aplicação não prossegue em um alerta é disparado.

```
src > App.vue > {} template
1 <script>
2 import { RouterLink, RouterView } from 'vue-router'
3 import { reactive, toRefs } from 'vue';
4 import Device from './components/Device.vue'
5 import tamagotchi from './components/tamagotchi.vue'
6 import Actions from './components/actions.vue'
7
8 export default {
9   name: 'Home',
10  components: {
11    Device,
12    tamagotchi,
13    //Actions,
14  },
15  setup() {
16    const state = reactive({
17      petname: '',
18      userInput: '',
19      showCreateTamagotchi: true,
20    })
21  }
```

```
22
23   const createTamagotchi = () => {
24     if (state.userinput.length == 0) {
25       window.alert("Informe um nome para o Tamagotchi!")
26     } else {
27       state.petname = state.userinput,
28       state.showCreateTamagotchi = false
29     }
30   }
31   return {
32     ...toRefs(state),
33     createTamagotchi
34   },
35 }
```

```
39 <template>
40   <h1>{{ petname }}</h1>
41   <section :class="$style.wrapper">
42     <Device />
43     <tamagotchi v-if="petname" />
44   </section>
45
46   <form v-if="showCreateTamagotchi" @submit.prevent>
47     <input v-model="userinput" type="text" placeholder="Tamagotchi Name" id="petname">
48     <button @click="createTamagotchi" type="button" class="btn btn-success">Create PET
49   </button>
50 </form>
51 <RouterView />
52 </template>
```

```

54 <style module>
    1 reference
55 .wrapper {
56   position: relative;
57 }
58
59 h1 {
60   text-shadow: 0px 0px 1px white;
61   font-family: monospace;
62   font-weight: 700;
63   text-transform: capitalize;
64   text-align: center;
65 }
66
67 form {
68   margin-top: 20px;
69 }
70
71 form button {
72   margin-left: 10px;
73   text-shadow: 0px 0px 3px black;
74 }
75 </style>
76

```

Componentes: Implementação dos componentes Vue.

Device.vue: implementa através de um .svg o aparelho que simula o dispositivo virtual, é um svg estático que não muda de estado, apenas é posicionado na página.

```

index.html  App.vue  Device.vue x
src > components > Device.vue > {} template
1 <template>
2   
3 </template>
4
5 <script>
6 </script>
7
8 <style scoped>
    1 reference
9   .device {
10     width: 310px;
11   }
12 </style>

```

Tamagotchi.vue: O tamagotchi é definido aqui, são vários svgs que ocupam o caminho da tag de imagem conforme o humor que é emitido pelo componente actions, assim conforme o usuário interage com o tamagotchi e o componente actions altera o estado do tamagotchi, essa mudança de humor é passada para o componente tamagotchi que muda para o .svg correspondente.

```
index.html App.vue Device.vue tamagotchi.vue X
src > components > tamagotchi.vue > {} template
1 <template>
2   
3   <Actions @humorUpdate="currentlyHumor"></Actions>
4 </template>
5
6 <script>
7   import Actions from './actions.vue';
8
9   export default {
10     components: {
11       Actions,
12     },
13     data() {
14       return {
15         humor: '',
16       }
17     },
18     methods: {
19       currentlyHumor(newHumor) {
20         this.humor = newHumor;
21       }
22     },
23
24     computed: {
25       Humor() {
26         if (this.humor === 'Angry') {
27           return '../images/angrypanda.svg';
28         } else if (this.humor === 'Hungry') {
29           return '../images/angrypanda.svg';
30         } else if (this.humor === 'Tired') {
31           return '../images/sadpanda.svg';
32         } else if (this.humor === 'Happy') {
33           return '../images/happypanda.svg';
34         } else if (this.humor === 'Sad') {
35           return '../images/sadpanda.svg';
36         } else if (this.humor === 'Sleeping') {
37           return '../images/sleepingpanda.svg';
38         } else if (this.Humor === 'Playing') {
39           return '../images/playingpanda.svg';
40         } else {
41           return '../images/normalpanda.svg';
42         }
43       },
44     },
45   };
46 </script>
```

```

46
47 <style>
48 .panda {
49   max-width: 150px;
50   position: absolute;
51   left: 81px;
52   top: 125px;
53 }
54 </style>
55

```

Actions.vue: Aqui é implementada a logica principal da aplicação, todo o comportamento do tamagotchi é definido aqui. Após criar o tamagotchi inserindo seu nome os botões de interação ficam ativos assim como a caixa de mensagens das acoes e a caixa de status. O tamagotchi inicia em um estado neutro e a cada interação feita como alimentar, brincar ou dormir incrementa e desconta de seus atributos como energia, fome e felicidade e altera seu humor. Metricas foram definidas como parâmetros para as mudanças de humor, a partir de cada interação feita é disparada uma função check que avalia qual será o humor do tamagotchi após a acao feita, assim, os status são atualizados e a caixa de mensagem exibe o humor do tamagotchi. Mensagens Alert também são disparadas para ações que não são possíveis ou necessárias. As cores alteram conforme o objeto muda de estado.

```

index.html App.vue Device.vue tamagotchi.vue actions.vue X
src > components > actions.vue > {} style > .message p
1 <template>
2   <div class="message" v-bind:class="getClassColor()">
3     <h2>AÇÕES:</h2>
4     <p><span v-if="showMessage">{{ messageText }}</span></p>
5     <p><span>{{ messageText2 }}</span></p>
6   </div>
7   <div class="status">
8     <h2>STATUS:</h2>
9     <p>Energy: <span v-bind:class="getClassEnergia()">{{ energy }}</span> </p>
10    <p>Happiness: <span v-bind:class="getClassHappiness()">{{ happiness }}</span> </p>
11    <p>Hunger: <span v-bind:class="getClassHunger()">{{ hunger }}</span> </p>
12    <p>Humor: <span v-bind:class="getClassHumor()">{{ humor }}</span> </p>
13  </div>
14  <div class="actions">
15    <button type="button" class="btn btn-primary" @click="feed">Feed</button>
16    <button type="button" class="btn btn-primary" @click="play">Play</button>
17    <button type="button" class="btn btn-primary" @click="sleep">Sleep</button>
18  </div>
19 </template>

```

```

21 <script>
22 export default {
23   data() {
24     return {
25       energy: 60,
26       happiness: 60,
27       hunger: 40,
28       humor: "Neutro",
29       intervalID: null,
30       showMessage: true,
31       messageText: "Pet Criado!",
32     };
33   },

```

```

34   methods: {
35     getClassColor() {
36       if (this.humor == "Happy") {
37         return 'border-happy';
38       } else if (this.humor == "Angry") {
39         return 'border-angry';
40       } else if (this.humor == "Neutro") {
41         return 'border-neutro';
42       } else if (this.humor == "Sad") {
43         return 'border-sad';
44       } else if (this.humor == "Tired") {
45         return 'border-tired';
46       } else if (this.humor == "Sleeping") {
47         return 'border-sleeping';
48       } else if (this.humor == "Hungry") {
49         return 'border-hungry';
50       }
51     },

```

```

52     getClassEnergia() {
53       if (this.energy <= 30) {
54         return 'energia-baixa';
55       } else if (this.energy <= 70) {
56         return 'energia-media';
57       } else {
58         return 'energia-alta';
59       }
60     },
61     getClassHappiness() {
62       if (this.happiness < 30) {
63         return 'hapinnes-baixo';
64       } else if (this.happiness < 50) {
65         return 'hapinnes-medio';
66       } else if (this.happiness < 70) {
67         return 'hapinnes-ok';
68       } else {
69         return 'hapinnes-alto';
70       }
71     },

```

```

72     getClassHunger() {
73         if (this.hunger <= 20) {
74             return 'hunger-baixo';
75         } else if (this.hunger <= 60) {
76             return 'hunger-medio';
77         } else if (this.hunger <= 70) {
78             return 'hunger-alto';
79         } else {
80             return 'starving';
81         }
82     },
83     getClassHumor() {
84         if (this.humor == "Angry") {
85             return 'angry-humor';
86         } else if (this.humor == "Neutro") {
87             return 'normal-humor';
88         } else if (this.humor == "Happy") {
89             return 'happy-humor';
90         } else if (this.humor == "Hungry") {
91             return 'hungry-humor';
92         } else if (this.humor == "Tired") {
93             return 'tired-humor';
94         } else if (this.humor == "Sad") {
95             return 'sad-humor';
96         } else if (this.humor == "Sleeping") {
97             return 'sleeping-humor';
98         }
99     },

```

```

100     checkStatusFeed() {
101         // pode estar angry, hungry, tired, sad e happy
102         if (this.hunger > 70) {
103             this.humor = "Angry";
104             this.setMessage("O Tamagotchi esta Bravo!");
105         } else if (this.hunger <= 70 && this.hunger >= 60) {
106             this.humor = "Hungry";
107             this.setMessage("O Tamagotchi esta com Fome!");
108         } else if (this.energy <= 30) {
109             this.humor = "Tired";
110             this.setMessage("O Tamagotchi esta Cansado!");
111         } else if (this.happiness <= 30) {
112             this.humor = "Sad";
113             this.setMessage("O Tamagotchi esta Triste!");
114         } else {
115             this.humor = "Happy";
116             this.setMessage("O Tamagotchi esta Feliz!");
117         }
118         this.$emit('humorUpdate', this.humor);
119     },

```

```

120     feed() {
121         if (this.hunger > 0) {
122             if (this.energy == 0) {
123                 window.alert("TAMAGOTCHI ESTA SEM ENERGIA PARA COMER, FACA-O DORMIR!");
124             }
125             else {
126                 if (this.hunger - 20 < 0) {
127                     this.hunger = 0;
128                 } else {
129                     this.hunger -= 20;
130                 }
131                 this.energy -= 10;
132             }
133             this.checkStatusFeed();
134         } else {
135             window.alert("TAMAGOTCHI NAO ESTA COM FOME!");
136         }
137     },

```



```

138     checkStatusPlay() {
139         // pode ficar neutro, feliz, com fome e cansado
140         if (this.hunger <= 70 && this.hunger >= 60) {
141             this.humor = "Hungry";
142             this.setMessage("O Tamagotchi esta com Fome!");
143         } else if (this.energy <= 30) {
144             this.humor = "Tired";
145             this.setMessage("O Tamagotchi esta Cansado!");
146         } else if (this.happiness >= 70) {
147             this.humor = "Happy";
148             this.setMessage("O Tamagotchi esta Feliz!");
149         }
150
151         this.$emit('humorUpdate', this.humor);
152     },

```

```

153     play() {
154         if (this.happiness == 100) {
155             window.alert("TAMAGOTCHI NAO PRECISA BRINCAR!");
156         } else if (this.energy > 30) {
157             if (this.hunger >= 60) {
158                 window.alert("TAMAGOTCHI ESTA COM FOME!");
159             }
160             else {
161
162                 if (this.happiness + 20 > 100) {
163                     this.happiness = 100;
164                 } else {
165                     this.happiness += 20;
166                 }
167                 this.energy -= 20;
168                 this.hunger += 20;
169                 if (this.energy <= 30) {
170                     window.alert("TAMAGOTCHI ESTA SEM ENERGIA APÓS BRINCAR!");
171                 }
172             }
173             this.checkStatusPlay();
174         } else {
175             window.alert("TAMAGOTCHI ESTA SEM ENERGIA, FACA-O DORMIR!");
176         }
177     },

```

```

178     checkStatusSleep() {
179         // pode acordar neutro, triste, feliz ou com fome
180         if (this.hunger > 70) {
181             this.humor = "Angry";
182             this.setMessage("O Tamagotchi esta Bravo!");
183         } else if (this.hunger > 50) { // Hungry
184             this.humor = "Hungry";
185             this.setMessage("O Tamagotchi esta com Fome!");
186         } else if (this.happiness <= 30) { // Sad
187             this.humor = "Sad";
188             this.setMessage("O Tamagotchi esta Triste!");
189         } else if (this.happiness >= 70) { // Happy
190             this.humor = "Happy";
191             this.setMessage("O Tamagotchi esta Feliz!");
192         } else {
193             this.humor = "Neutro";
194             this.setMessage("O Tamagotchi esta Neutro!");
195         }
196         this.$emit('humorUpdate', this.humor);
197     },

```

```

198     sleep() {
199         if (this.energy > 60) {
200             window.alert("TAMAGOTCHI NAO PRECISA DORMIR!");
201         } else if (this.energy <= 60) {
202             const resposta = confirm("TAMOGOTCHI IRÁ DORMIR...");
203             if (resposta == true) {
204                 if (this.intervalId) {
205                     clearInterval(this.intervalId);
206                     this.intervalId = null;
207                 } else {
208                     this.humor = "Sleeping";
209                     let tamagotchiRecarregado = false;
210                     let count = 0;
211                     this.$emit('humorUpdate', this.humor);
212                     this.setMessage("Tamagotchi esta dormindo!");
213                     this.getClassColor();
214                     this.intervalId = setInterval(() => {
215                         this.energy += 10;
216                         count++;
217                         if (count == 2) {
218                             if (this.hunger < 100) {
219                                 this.hunger += 10;
220                             } else if (this.hunger == 100) {
221                                 this.hunger = 100;
222                             }
223                             count = 0;
224                         }

```

```

225             if (this.energy >= 100) {
226                 this.energy = 100;
227                 clearInterval(this.intervalId);
228                 this.intervalId = null;
229                 tamagotchiRecarregado = true;
230                 // Usar setTimeout para exibir o alerta após a última iteração
231                 setTimeout(() => {
232                     if (tamagotchiRecarregado) {
233                         if (this.hunger > 70) {
234                             window.alert("TAMAGOTCHI ACORDOU FAMINTO!");
235                         } else if (this.hunger <= 70 && this.hunger >= 60) {
236                             window.alert("TAMAGOTCHI ACORDOU COM FOME!");
237                         }
238                         if (this.happiness > 0) {
239                             this.happiness -= 30;
240                         } else if (this.happiness <= 0) {
241                             this.happiness = 0;
242                         }
243                         if (this.happiness <= 0) {
244                             window.alert("TAMAGOTCHI ACORDOU TRISTE!");
245                         }
246                         window.alert("TAMAGOTCHI RECARREGADO 100%!");
247                         tamagotchiRecarregado = false; // Resetar a variável para
248                             futuras recargas
249                         this.checkStatusSleep();
250                     }, 0); // 0ms para que o alerta seja colocado na fila de tarefas
251                 }
252             }, 0); // 0ms para que o alerta seja colocado na fila de tarefas
253         }, 1000); // 1000ms = 1 segundo
254     }
255     } else {
256         window.alert("TAMAGOTCHI ACORDADO!");
257     }
258 },
259 },

```

```

260         setMessage(newMessage) {
261             this.messageText = newMessage;
262         },
263     }
264 };
265 </script>
266

```

```

267 <style>
268 |
269 > .message { ...
282 | }
283
284 > .message p { ...
293 | }
294
295 > .message h2 { ...
302 | }
303

```

```

304 > .status { ...
312 | }
313
314 > .status p { ...
321 | }
322
323 > .status h2 { ...
327 | }

```

```

329 > .actions { ...
334 | }
335
336 > .actions button { ...
339 | }
340

```

```

341 /* Cores de indentificacao para a energia do tamagotchi */
342 > .energia-baixa { ...
344 | }
345
346 > .energia-media { ...
348 | }
349
350 > .energia-alta { ...
352 | }
353

```

```

354 /* Cores de identificacao para a felicidade do tamagotchi */
355 > .hapinnes-baixo { ...
357 | }
358
359 > .hapinnes-medio { ...
361 | }
362
363 > .hapinnes-ok { ...
365 | }
366
367 > .hapinnes-alto { ...
369 | }
370

```

```
371  /* Cores de identificacao para a fome do tamagotchi */
372 > .hunger-baixo {...
374   }
375
376 > .hunger-medio {...
378   }
379
380 > .hunger-alto {...
382   }
383
384 > .starving {...
386   }
```

```
388  /* Cores de identificacao para o humor do tamagotchi */
389 > .angry-humor {...
391   }
392
393 > .normal-humor {...
395   }
396
397 > .happy-humor {...
399   }
400
401 > .hungry-humor {...
403   }
404
405 > .tired-humor {...
407   }
408
409 > .sad-humor {...
411   }
412
413 > .sleeping-humor {...
415   }
```

```
417  /* Cores da borda da caixa de acoes */
418 > .border-happy {...
420   }
421
422 > .border-angry {...
424   }
425
426 > .border-sad {...
428   }
429
430 > .border-hungry {...
432   }
433
434 > .border-tired {...
436   }
437
438 > .border-sleeping {...
440   }
441
442 > .border-normal {...
444   }
445 </style>
446
```

Main.css: este arquivo define o layout padrao de toda a aplicacao importando o arquivo base.css como padrao na criacao do projeto que define alguns esquemas de cores default. Aqui são feitas as edicoes sobre o header e a disposicao dos objetos na pagina.

```
1  @import './base.css';
2
3  > html { ...
5  }
6
7  > header { ...
17 }
18
19 > header h1 { ...
22 }
23
24 > header a { ...
36 }
37
38 > header a:hover { ...
44 }
45
46 > .header-voltar { ...
49 }
50
51 > .header-voltar h1 { ...
53 }
54
55 > .header-voltar a { ...
60 }
61
62 > .header-voltar a:hover { ...
69 }
```

```

71 > .header-sobre { ...
74 }
75
76 > #app { ...
84 }
85
86 > #page2 { ...
96 }
97
98 > #page2 p { ...
107 }
108
109 > #page2 div { ...
115 }
116
117 > #page2 h1 { ...
119 }
120 |
121 > @media (min-width: 1024px) { ...
132 }

```

Main.js: instancia a aplicação app.vue bem como a importação do bootstrap global para todo o projeto.

```

1 import './assets/main.css'
2 import { createApp } from 'vue'
3 import App from './App.vue'
4 import router from './router'
5 import 'bootstrap';
6
7
8 const app = createApp(App)
9
10 app.use(router)
11
12 app.mount('#app')
13

```

Rodando o projeto:

```

PS C:\Users\HP\OneDrive - UNIOESTE\Área de Trabalho\RedPanda\redpandatamagotchi> npm run dev
> redpandatamagotchi@0.0.0 dev
> vite

VITE v4.4.4 ready in 1136 ms

  → Local:   http://127.0.0.1:5173/
  → Network: use --host to expose
  → press h to show help

```

TAMAGOTCHI

[Sobre](#)[Create PET](#)

TAMAGOTCHI

[Voltar](#)

Breve Historia Sobre Tamagotchi

O Tamagotchi é um brinquedo eletrônico virtual criado pela empresa japonesa Bandai. Foi lançado pela primeira vez em 1996 no Japão e rapidamente se tornou um fenômeno mundial, conquistando uma enorme popularidade entre crianças e adultos de diversas faixas etárias. A ideia por trás do Tamagotchi era proporcionar uma experiência de cuidar de um animal de estimação virtual. Cada dispositivo possuía uma tela de LCD em preto e branco, onde um ovo digital aparecia inicialmente. O objetivo era cuidar e alimentar o Tamagotchi, assim como se faz com um animal de verdade, através de uma série de botões para selecionar ações como alimentação, limpeza e brincadeiras.

O Tamagotchi tinha suas próprias necessidades e emoções, e o modo como era cuidado determinava o seu crescimento e desenvolvimento. Se fosse bem alimentado e cuidado, ele se transformava em uma criatura saudável e feliz. No entanto, se fosse negligenciado, poderia ficar doente ou até mesmo morrer. Essa interação e responsabilidade criavam um senso de conexão emocional entre os usuários e seus Tamagotchis virtuais.

A popularidade do Tamagotchi atingiu seu auge na década de 1990, com milhões de unidades vendidas em todo o mundo. O brinquedo se tornou uma verdadeira febre, gerando diversos produtos licenciados, como desenhos animados, filmes, jogos de vídeo game e até mesmo uma linha de roupas e acessórios.

Com o tempo, o interesse no Tamagotchi diminuiu, mas a nostalgia dos anos 90 trouxe de volta a popularidade do brinquedo em várias ocasiões. A Bandai lançou várias versões atualizadas e reedições do Tamagotchi, e ele continua sendo um brinquedo amado por muitos colecionadores e entusiastas.

O Tamagotchi deixou um legado duradouro na cultura pop e é considerado um dos primeiros brinquedos eletrônicos virtuais de grande sucesso. Sua história é uma lembrança vívida da evolução da tecnologia e do poder que um simples brinquedo pode ter para cativar a imaginação das pessoas em todo o mundo.



Link do repositório codeberg que contém a implementação do projeto:
<https://codeberg.org/Sinclair3131/Tamagotchi-VueJS.git>