

src/components/AlertMessage.vue

```
1 <template>
2   <transition name="fade">
3     <div
4       v-if="show"
5       :class="['alert', `alert-${type}`, isClosable ? 'closable' : '']"
6       role="alert"
7     >
8       <div class="alert-content">
9         <strong v-if="title" class="alert-title">{{ title }}</strong>
10        <p class="alert-message">{{ message }}</p>
11      </div>
12      <button
13        v-if="isClosable"
14        class="alert-close"
15        @click="close"
16        aria-label="Fechar"
17      >
18        &times;
19      </button>
20    </div>
21  </transition>
22 </template>
23
24 <script setup>
25 import { defineProps, defineEmits, ref, watch, onMounted } from 'vue';
26
27 const props = defineProps({
28   message: {
29     type: String,
30     required: true
31   },
32   title: {
33     type: String,
34     default: ''
```

```
35   },
36   type: {
37     type: String,
38     default: 'info',
39     validator: (value) => ['info', 'success', 'warning', 'error'].includes(value)
40   },
41   isClosable: {
42     type: Boolean,
43     default: true
44   },
45   autoClose: {
46     type: Boolean,
47     default: false
48   },
49   duration: {
50     type: Number,
51     default: 5000
52   }
53 });
54
55 const emit = defineEmits(['close']);
56 const show = ref(true);
57
58 const close = () => {
59   show.value = false;
60   emit('close');
61 };
62
63 // Fecha automaticamente após a duração especificada
64 let timeoutId = null;
65
66 const startAutoCloseTimer = () => {
67   if (props.autoClose && props.duration > 0) {
68     timeoutId = setTimeout(() => {
69       close();
70     }, props.duration);
71   }
```

```
72 | };
73 |
74 | // Limpa o temporizador quando o componente é desmontado
75 | onMounted(() => {
76 |   startAutoCloseTimer();
77 | });
78 |
79 | // Reseta o temporizador quando as props mudam
80 | watch(() => [props.message, props.type], () => {
81 |   if (timeoutId) {
82 |     clearTimeout(timeoutId);
83 |   }
84 |   show.value = true;
85 |   startAutoCloseTimer();
86 | });
87 | </script>
88 |
89 | <style scoped>
90 | .alert {
91 |   display: flex;
92 |   align-items: flex-start;
93 |   justify-content: space-between;
94 |   padding: 12px 16px;
95 |   margin-bottom: 16px;
96 |   border-radius: 4px;
97 |   font-size: 0.95rem;
98 |   box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);
99 | }
100 |
101 | .alert-content {
102 |   flex: 1;
103 | }
104 |
105 | .alert-title {
106 |   display: block;
107 |   margin-bottom: 4px;
108 |   font-weight: 600;
```

```
109 | }
110 |
111 | .alert-message {
112 |     margin: 0;
113 |     line-height: 1.4;
114 | }
115 |
116 | .alert-close {
117 |     background: transparent;
118 |     border: none;
119 |     font-size: 1.5rem;
120 |     padding: 0 8px;
121 |     margin-left: 8px;
122 |     cursor: pointer;
123 |     opacity: 0.7;
124 |     transition: opacity 0.2s;
125 | }
126 |
127 | .alert-close:hover {
128 |     opacity: 1;
129 | }
130 |
131 | /* Tipos de alerta */
132 | .alert-info {
133 |     background-color: #e3f2fd;
134 |     border-left: 4px solid #2196f3;
135 |     color: #0d47a1;
136 | }
137 |
138 | .alert-success {
139 |     background-color: #e8f5e9;
140 |     border-left: 4px solid #4caf50;
141 |     color: #1b5e20;
142 | }
143 |
144 | .alert-warning {
145 |     background-color: #fff3e0;
```

```
146     border-left: 4px solid #ff9800;
147     color: #e65100;
148 }
149
150 .alert-error {
151     background-color: #ffebee;
152     border-left: 4px solid #f44336;
153     color: #b71c1c;
154 }
155
156 /* Animações */
157 .fade-enter-active,
158 .fade-leave-active {
159     transition: opacity 0.3s ease;
160 }
161
162 .fade-enter-from,
163 .fade-leave-to {
164     opacity: 0;
165 }
166
167 /* Media Query para telas menores */
168 @media (max-width: 768px) {
169     .alert {
170         padding: 10px 12px;
171         font-size: 0.9rem;
172     }
173 }
174 </style>
```