src/components/AlertMessage.vue

```
<template>
  <transition name="fade">
    <div
      v-if="show"
      :class="['alert', `alert-${type}`, isClosable ? 'closable' : '']"
      role="alert"
    >
      <div class="alert-content">
        <strong v-if="title" class="alert-title">{{ title }}</strong>
        <p class="alert-message">{{ message }}</p>
      </div>
      <button
        v-if="isClosable"
        class="alert-close"
        @click="close"
        aria-label="Fechar"
      >
        &times;
      </button>
    </div>
  </transition>
</template>

<script setup>
import { defineProps, defineEmits, ref, watch, onMounted } from 'vue';

const props = defineProps({
  message: {
    type: String,
    required: true
  },
  title: {
    type: String,
    default: ''
  },
  type: {
    type: String,
    default: 'info',
    validator: (value) => ['info', 'success', 'warning', 'error'].includes(value)
  },
  isClosable: {
    type: Boolean,
    default: true
  },
  autoClose: {
    type: Boolean,
    default: false
  },
  duration: {
    type: Number,
    default: 5000
  }
});

const emit = defineEmits(['close']);
const show = ref(true);

const close = () => {
  show.value = false;
  emit('close');
};

// Fecha automaticamente após a duração especificada
let timeoutId = null;

const startAutoCloseTimer = () => {
  if (props.autoClose && props.duration > 0) {
    timeoutId = setTimeout(() => {
      close();
    }, props.duration);
  }
```

```
};

// Limpa o temporizador quando o componente é desmontado
onMounted(() => {
  startAutoCloseTimer();
});

// Reseta o temporizador quando as props mudam
watch(() => [props.message, props.type], () => {
  if (timeoutId) {
    clearTimeout(timeoutId);
  }
  show.value = true;
  startAutoCloseTimer();
});
</script>

<style scoped>
.alert {
  display: flex;
  align-items: flex-start;
  justify-content: space-between;
  padding: 12px 16px;
  margin-bottom: 16px;
  border-radius: 4px;
  font-size: 0.95rem;
  box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);
}

.alert-content {
  flex: 1;
}

.alert-title {
  display: block;
  margin-bottom: 4px;
  font-weight: 600;
```

```css
}

.alert-message {
  margin: 0;
  line-height: 1.4;
}

.alert-close {
  background: transparent;
  border: none;
  font-size: 1.5rem;
  padding: 0 8px;
  margin-left: 8px;
  cursor: pointer;
  opacity: 0.7;
  transition: opacity 0.2s;
}

.alert-close:hover {
  opacity: 1;
}

/* Tipos de alerta */
.alert-info {
  background-color: #e3f2fd;
  border-left: 4px solid #2196f3;
  color: #0d47a1;
}

.alert-success {
  background-color: #e8f5e9;
  border-left: 4px solid #4caf50;
  color: #1b5e20;
}

.alert-warning {
  background-color: #fff3e0;
  border-left: 4px solid #ff9800;
  color: #e65100;
}

.alert-error {
  background-color: #ffebee;
  border-left: 4px solid #f44336;
  color: #b71c1c;
}

/* Animações */
.fade-enter-active,
.fade-leave-active {
  transition: opacity 0.3s ease;
}

.fade-enter-from,
.fade-leave-to {
  opacity: 0;
}

/* Media Query para telas menores */
@media (max-width: 768px) {
  .alert {
    padding: 10px 12px;
    font-size: 0.9rem;
  }
}
</style>
```