```
<script setup>
import { ref, computed, watch, onMounted, onUnmounted } from 'vue';
import { useMapRouting } from '../composables/useMapRouting';

const props = defineProps({
  waypoints: {
    type: Array,
    default: () => []
  },
  locations: {
    type: Array,
    default: () => []
  },
  currentFloor: {
    type: String,
    required: true
  },
  userPosition: {
    type: Object,
    default: null
  },
  selectedLocation: {
    type: Object,
    default: null
  },
  mapScale: {
    type: Number,
    default: 1
  },
  loading: {
    type: Boolean,
    default: false
  }
});

const emit = defineEmits([
  'map-click',
  'marker-click',
  'marker-mouseenter',
  'marker-mouseleave',
  'update:mapDimensions'
]);

const mapRef = ref(null);
const mapImageRef = ref(null);
const mapDimensions = ref({ width: 0, height: 0 });

// Filtra locais do andar atual
const currentFloorLocations = computed(() => {
  return props.locations.filter(loc => loc.andar === props.currentFloor);
});

// Configuração do sistema de rotas
const waypointsRef = ref(props.waypoints); // Ref para os waypoints
const { routeSegments, debugWaypoints } = useMapRouting(
  ref(props.userPosition),
  ref(props.selectedLocation),
  ref(props.currentFloor),
  ref(props.mapScale),
  mapDimensions,
  waypointsRef
);

// Atualiza os waypoints quando mudam
watch(() => props.waypoints, (newWaypoints) => {
  waypointsRef.value = newWaypoints;
}, { deep: true });

// Estilo do mapa
const mapStyle = computed(() => ({
  transform: `scale(${props.mapScale})`
}));

```

```
// Handlers de eventos
const handleMapClick = (event) => {
  if (!mapRef.value) return;

  const rect = mapRef.value.getBoundingClientRect();
  const x = ((event.clientX - rect.left) / rect.width) * 100;
  const y = ((event.clientY - rect.top) / rect.height) * 100;

  emit('map-click', { x, y });
};

const handleLocationClick = (location) => {
  emit('marker-click', location);
};

const handleLocationHover = (location) => {
  emit('marker-mouseenter', location);
};

const handleLocationLeave = () => {
  emit('marker-mouseleave');
};

// Atualização das dimensões do mapa
const updateMapDimensions = () => {
  if (!mapRef.value) return;

  const rect = mapRef.value.getBoundingClientRect();
  mapDimensions.value = {
    width: rect.width,
    height: rect.height
  };
  emit('update:mapDimensions', mapDimensions.value);
};

// Lifecycle hooks
onMounted(() => {
  updateMapDimensions();
  window.addEventListener('resize', updateMapDimensions);

  // Configurar ResizeObserver para mapImageRef se necessário
  if (mapImageRef.value) {
    const resizeObserver = new ResizeObserver(() => {
      updateMapDimensions();
    });
    resizeObserver.observe(mapImageRef.value);
  }
});

onUnmounted(() => {
  window.removeEventListener('resize', updateMapDimensions);

  // Limpar ResizeObserver se foi configurado
  if (mapImageRef.value && resizeObserver) {
    resizeObserver.disconnect();
  }
});
</script>

<template>
  <div
    ref="mapRef"
    class="map-container"
    @click="handleMapClick"
  >
    <img
      ref="mapImageRef"
      src="../assets/campusRussas.jpg"
      alt="Mapa do Campus UFC Russas"
      class="map-image"
      :style="mapStyle"
    />

    <!-- Marcadores de locais -->
    <div
      v-for="location in currentFloorLocations"
      :key="location.id"
      class="location-marker"
      :class="{ 'selected': location === props.selectedLocation }"
      :style="{
        left: `${location.x}%`,
        top: `${location.y}%`
      }"
      @click.stop="handleLocationClick(location)"
      @mouseenter="handleLocationHover(location)"
      @mouseleave="handleLocationLeave"
    >
      <span class="marker-label">{{ location.nome }}</span>
    </div>

    <!-- Waypoints de debug (se necessário) -->
    <div
      v-for="waypoint in debugWaypoints"
      :key="waypoint.id"
      class="waypoint-marker"
      :style="{
        left: `${waypoint.x}%`,
        top: `${waypoint.y}%`
      }"
    />

    <!-- Segmentos da rota -->
    <div
      v-for="(segment, index) in routeSegments"
      :key="index"
      class="route-segment"
      :style="{
        left: `${segment.x}%`,
        top: `${segment.y}%`,
        width: `${segment.length}%`,
        transform: `rotate(${segment.angle}deg)`
      }"
    />

    <!-- Marcador de posição do usuário -->
    <div
      v-if="userPosition"
      class="user-marker"
      :style="{
        left: `${userPosition.x}%`,
        top: `${userPosition.y}%`
      }"
    />

    <!-- Indicador de carregamento -->
    <div v-if="loading" class="loading-overlay">
      <div class="loading-spinner"></div>
    </div>
  </div>
</template>

<style scoped>
.map-container {
  position: relative;
  width: 100%;
  height: 100%;
  overflow: hidden;
}

.map-image {
  width: 100%;
  height: 100%;
  object-fit: contain;
  transform-origin: center center;
}

.location-marker {
  position: absolute;
```

```css
  width: 20px;
  height: 20px;
  background-color: #2196F3;
  border-radius: 50%;
  transform: translate(-50%, -50%);
  cursor: pointer;
  z-index: 2;
}

.location-marker.selected {
  background-color: #4CAF50;
  box-shadow: 0 0 0 4px rgba(76, 175, 80, 0.3);
}

.marker-label {
  position: absolute;
  bottom: 100%;
  left: 50%;
  transform: translateX(-50%);
  white-space: nowrap;
  background-color: rgba(0, 0, 0, 0.7);
  color: white;
  padding: 2px 6px;
  border-radius: 4px;
  font-size: 12px;
}

.waypoint-marker {
  position: absolute;
  width: 8px;
  height: 8px;
  background-color: rgba(255, 0, 0, 0.5);
  border-radius: 50%;
  transform: translate(-50%, -50%);
  z-index: 1;
}

```

```css
.route-segment {
  position: absolute;
  height: 4px;
  background-color: #4CAF50;
  transform-origin: left center;
  z-index: 1;
}

.user-marker {
  position: absolute;
  width: 16px;
  height: 16px;
  background-color: #FF5722;
  border-radius: 50%;
  border: 2px solid white;
  transform: translate(-50%, -50%);
  z-index: 3;
  box-shadow: 0 0 4px rgba(0, 0, 0, 0.3);
}

.loading-overlay {
  position: absolute;
  top: 0;
  left: 0;
  right: 0;
  bottom: 0;
  background-color: rgba(255, 255, 255, 0.7);
  display: flex;
  justify-content: center;
  align-items: center;
  z-index: 4;
}

.loading-spinner {
  width: 40px;
  height: 40px;
  border: 4px solid #f3f3f3;
  border-top: 4px solid #3498db;
  border-radius: 50%;
  animation: spin 1s linear infinite;
}

@keyframes spin {
  0% { transform: rotate(0deg); }
  100% { transform: rotate(360deg); }
}
</style>
```