Equipo 09

Open Weather API

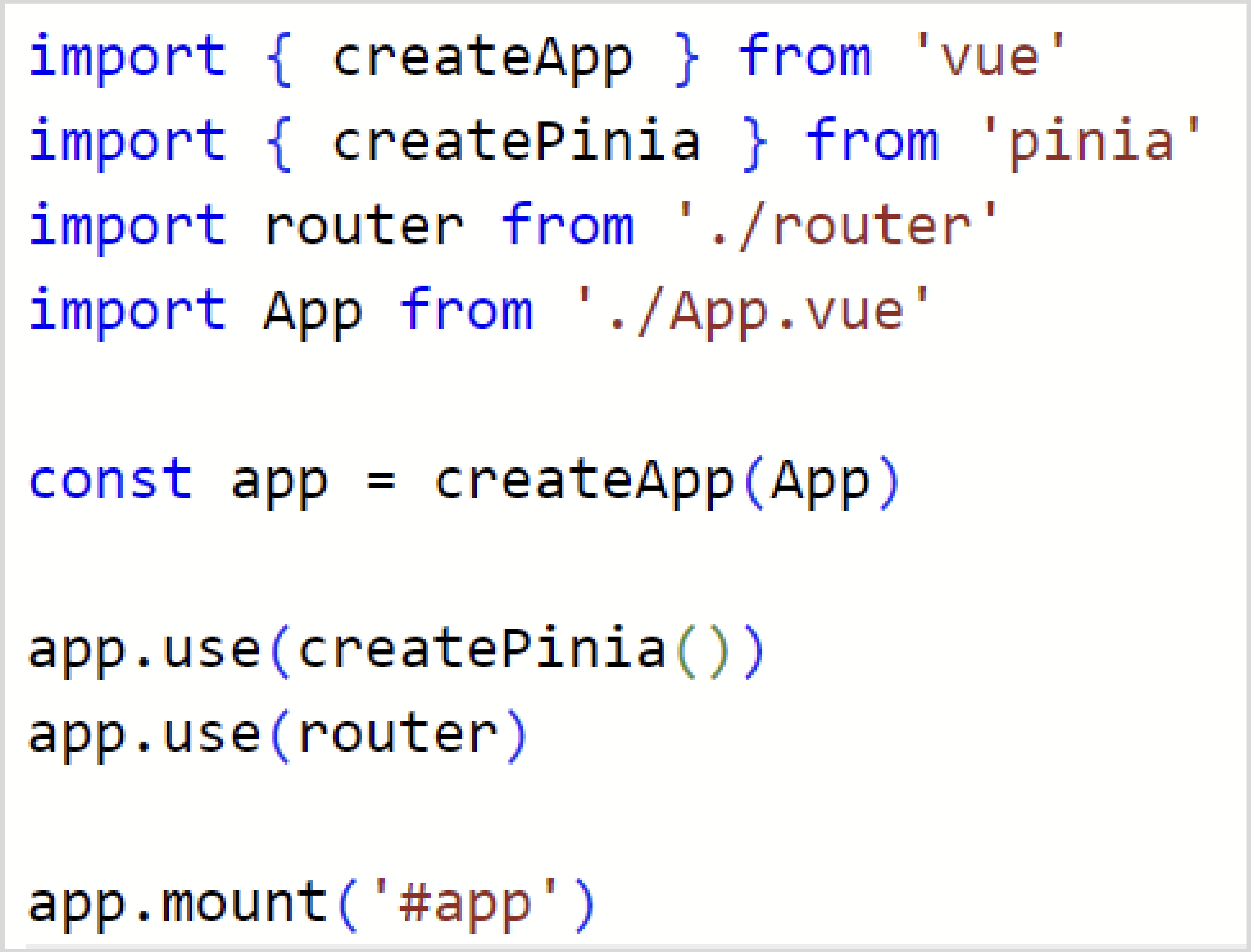
OpenWeather

Se utilizaron consultas de:

- Temperaturas
- Contaminación del aire

Demo

main.ts



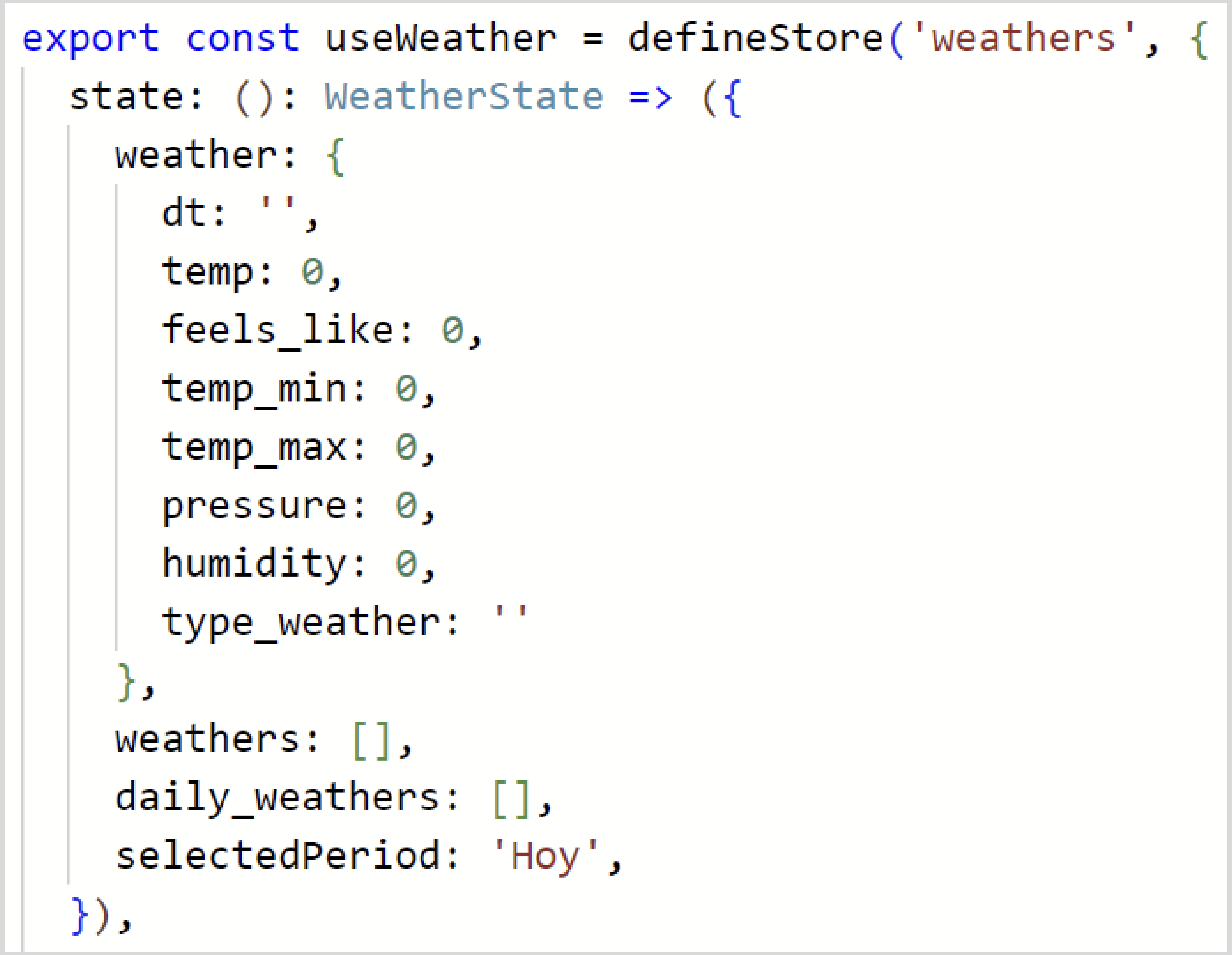
```
import { createApp } from 'vue'
import { createPinia } from 'pinia'
import router from './router'
import App from './App.vue'

const app = createApp(App)

app.use(createPinia())
app.use(router)

app.mount('#app')
```

Store: weather.ts

```
export interface Weather {
  dt: string
  temp: number
  feels_like: number
  temp_min: number
  temp_max: number
  pressure: number
  humidity: number
  type_weather: string
}
```

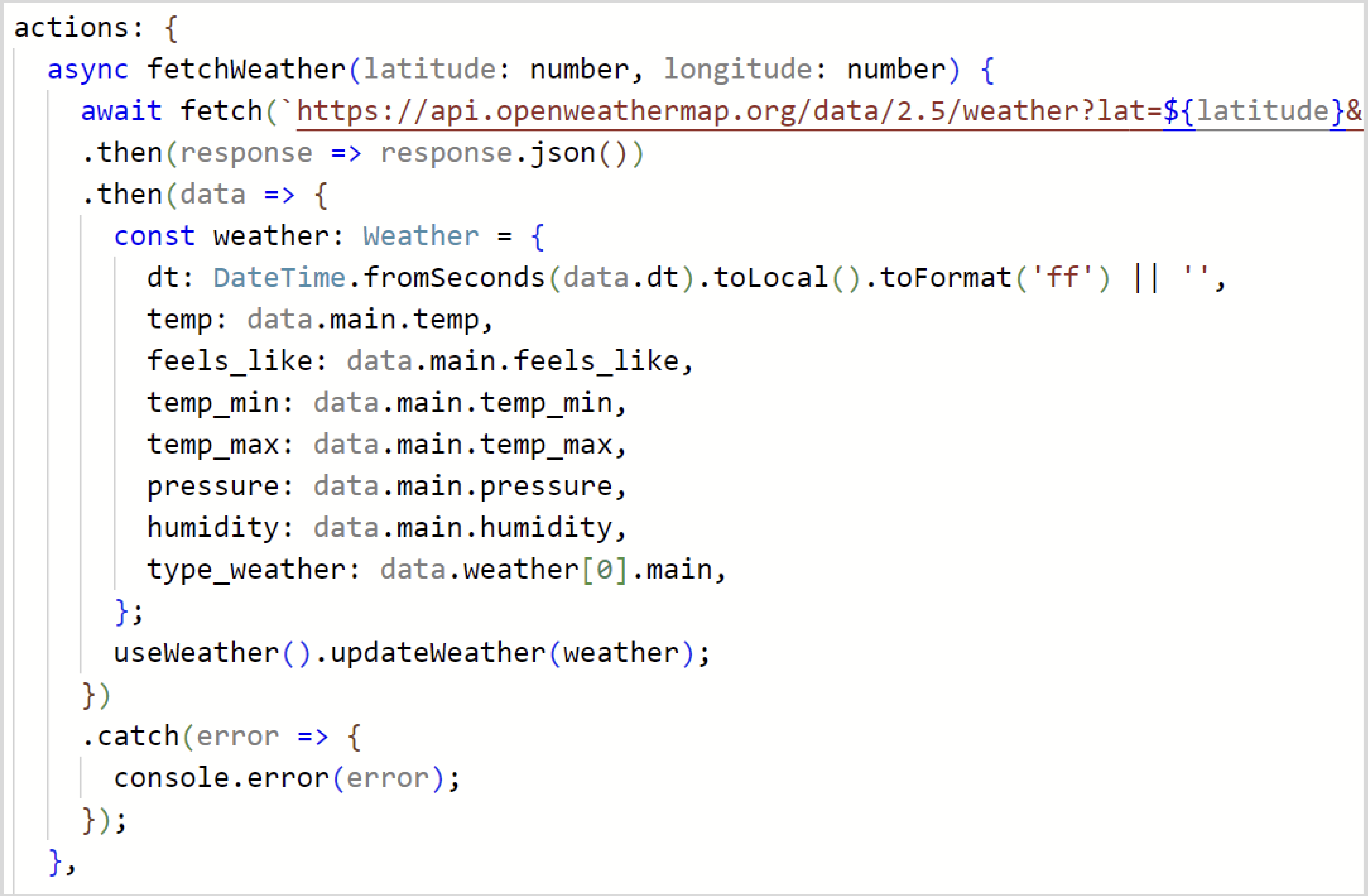
```
import { defineStore } from 'pinia'
import { DateTime } from 'luxon'
import { Weather } from '../scripts/weather'
import { PeriodTemp, periodsTemp } from '../scripts/constants'

interface WeatherState {
  weather: Weather
  weathers: Weather[],
  daily_weathers: Weather[][],
  selectedPeriod: PeriodTemp
}
```

Store: weather.ts

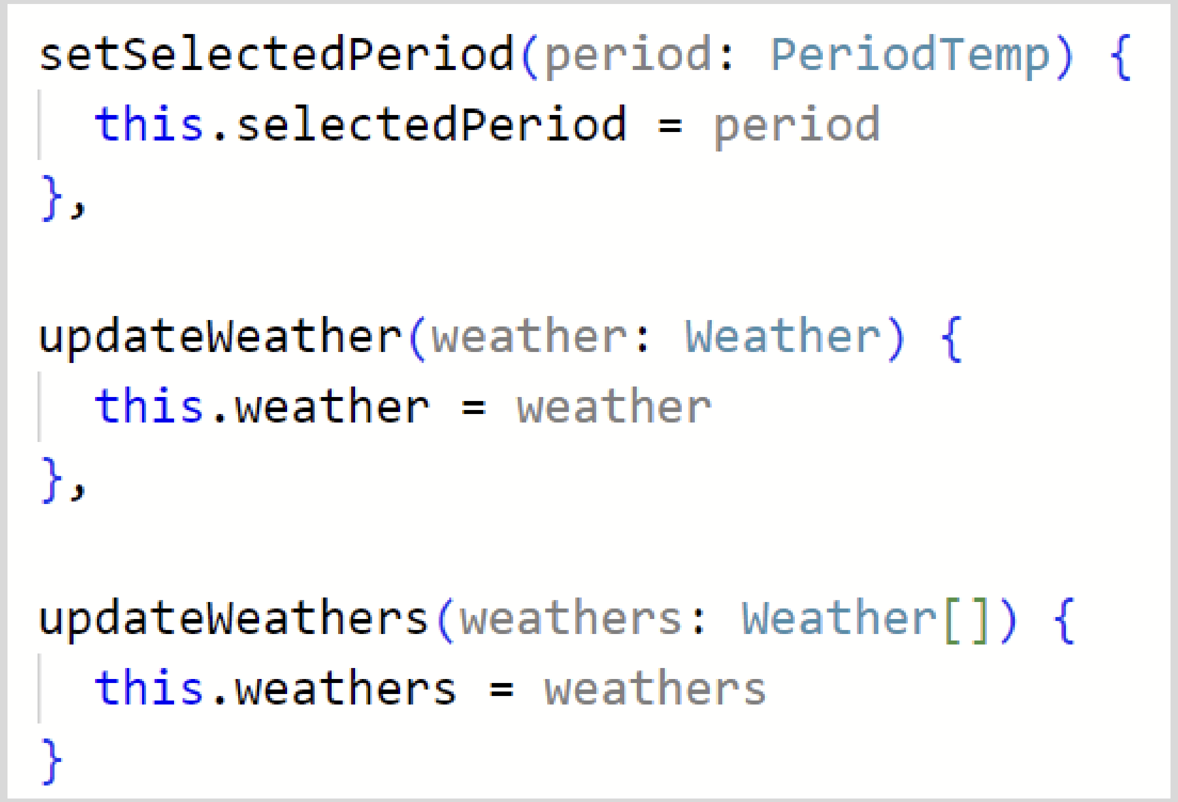


```
export const useWeather = defineStore('weathers', {
  state: (): WeatherState => ({
    weather: {
      dt: '',
      temp: 0,
      feels_like: 0,
      temp_min: 0,
      temp_max: 0,
      pressure: 0,
      humidity: 0,
      type_weather: ''
    },
    weathers: [],
    daily_weathers: [],
    selectedPeriod: 'Hoy',
  }),
```

Store: weather.ts

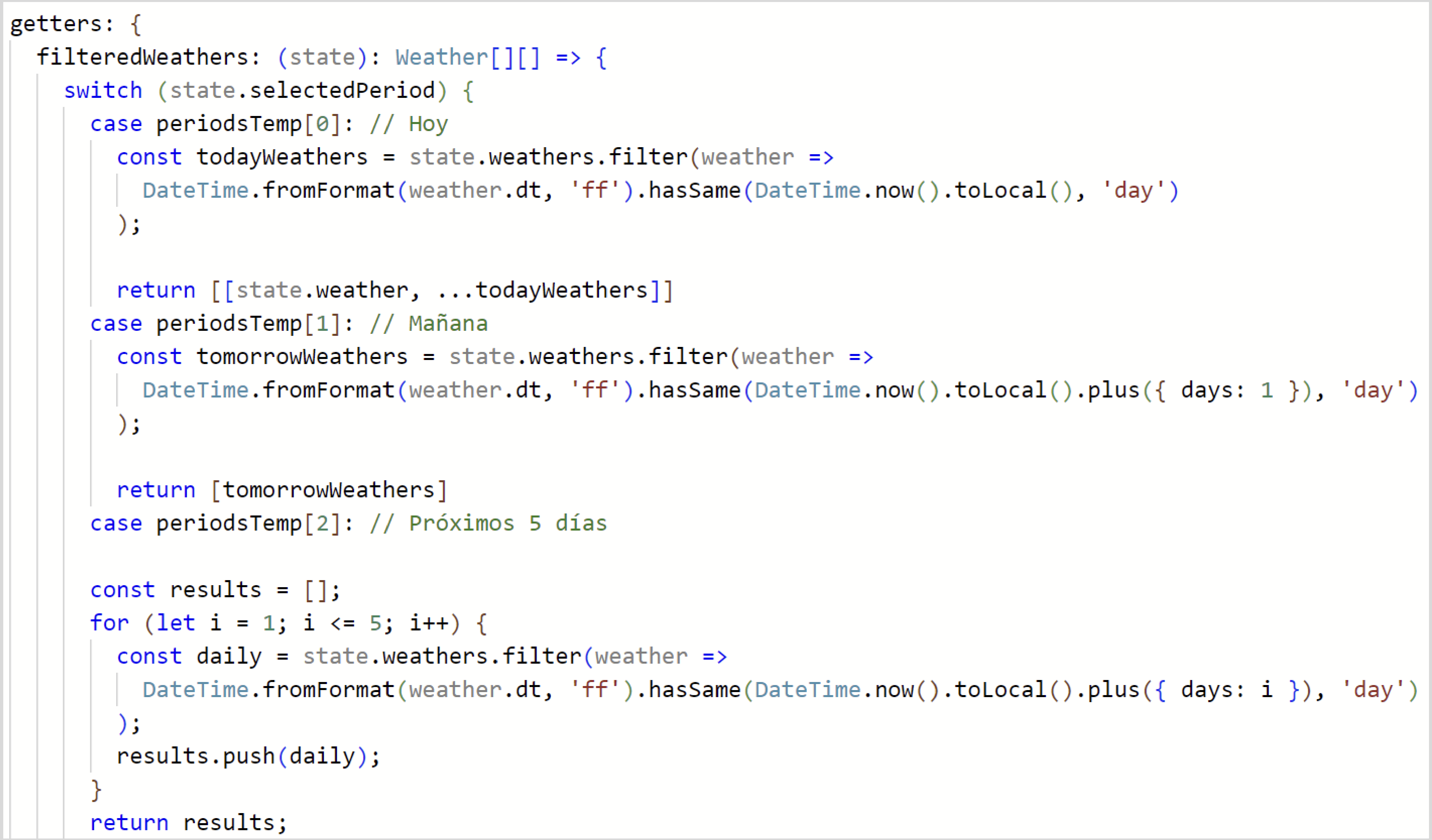


```
actions: {
  async fetchWeather(latitude: number, longitude: number) {
    await fetch(`https://api.openweathermap.org/data/2.5/weather?lat=${latitude}&
    .then(response => response.json())
    .then(data => {
      const weather: Weather = {
        dt: DateTime.fromSeconds(data.dt).toLocal().toFormat('ff') || '',
        temp: data.main.temp,
        feels_like: data.main.feels_like,
        temp_min: data.main.temp_min,
        temp_max: data.main.temp_max,
        pressure: data.main.pressure,
        humidity: data.main.humidity,
        type_weather: data.weather[0].main,
      };
      useWeather().updateWeather(weather);
    })
    .catch(error => {
      console.error(error);
    });
  },
```



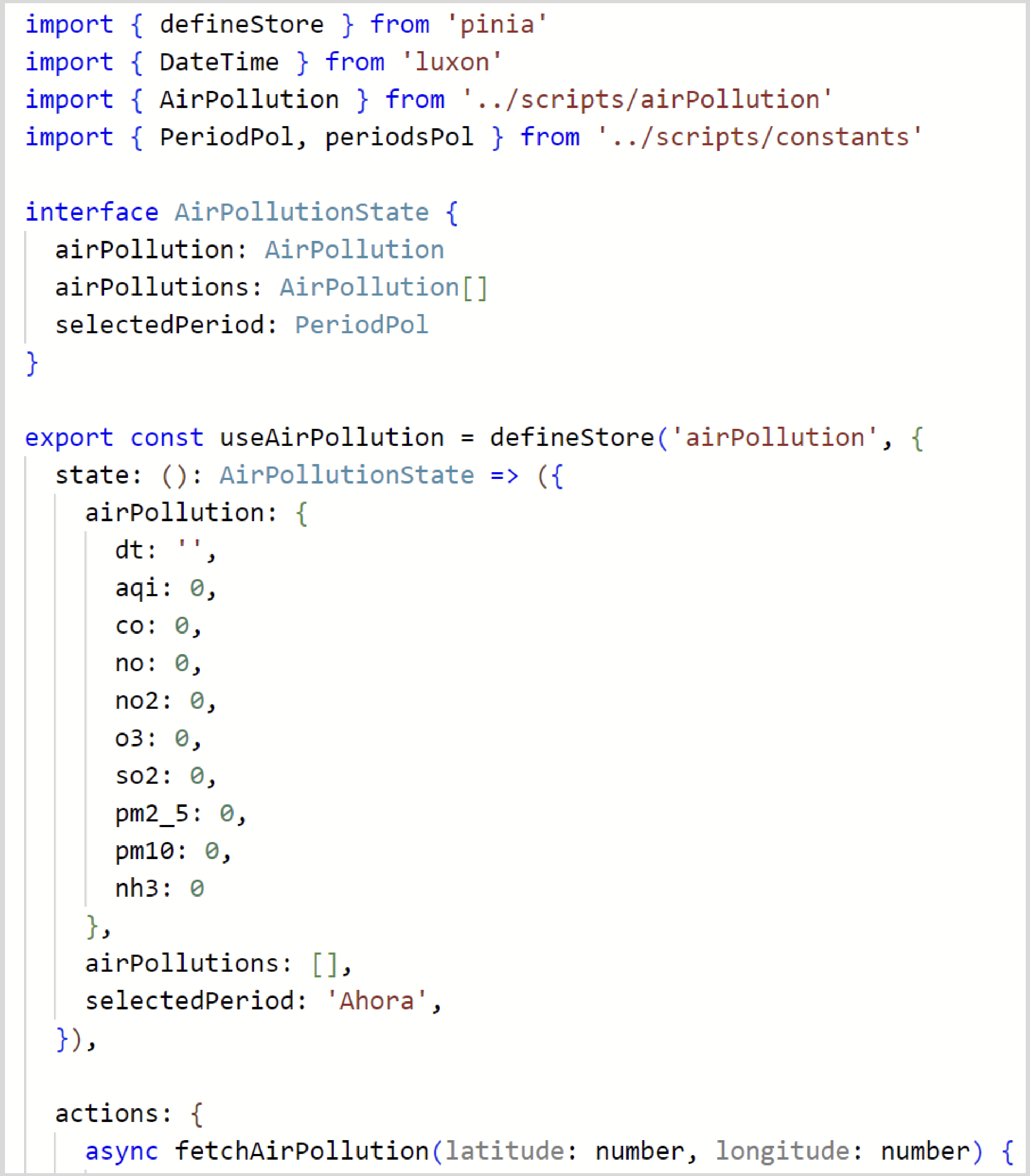
```
setSelectedPeriod(period: PeriodTemp) {
  this.selectedPeriod = period
},

updateWeather(weather: Weather) {
  this.weather = weather
},

updateWeathers(weathers: Weather[]) {
  this.weathers = weathers
}
```

Store: weather.ts

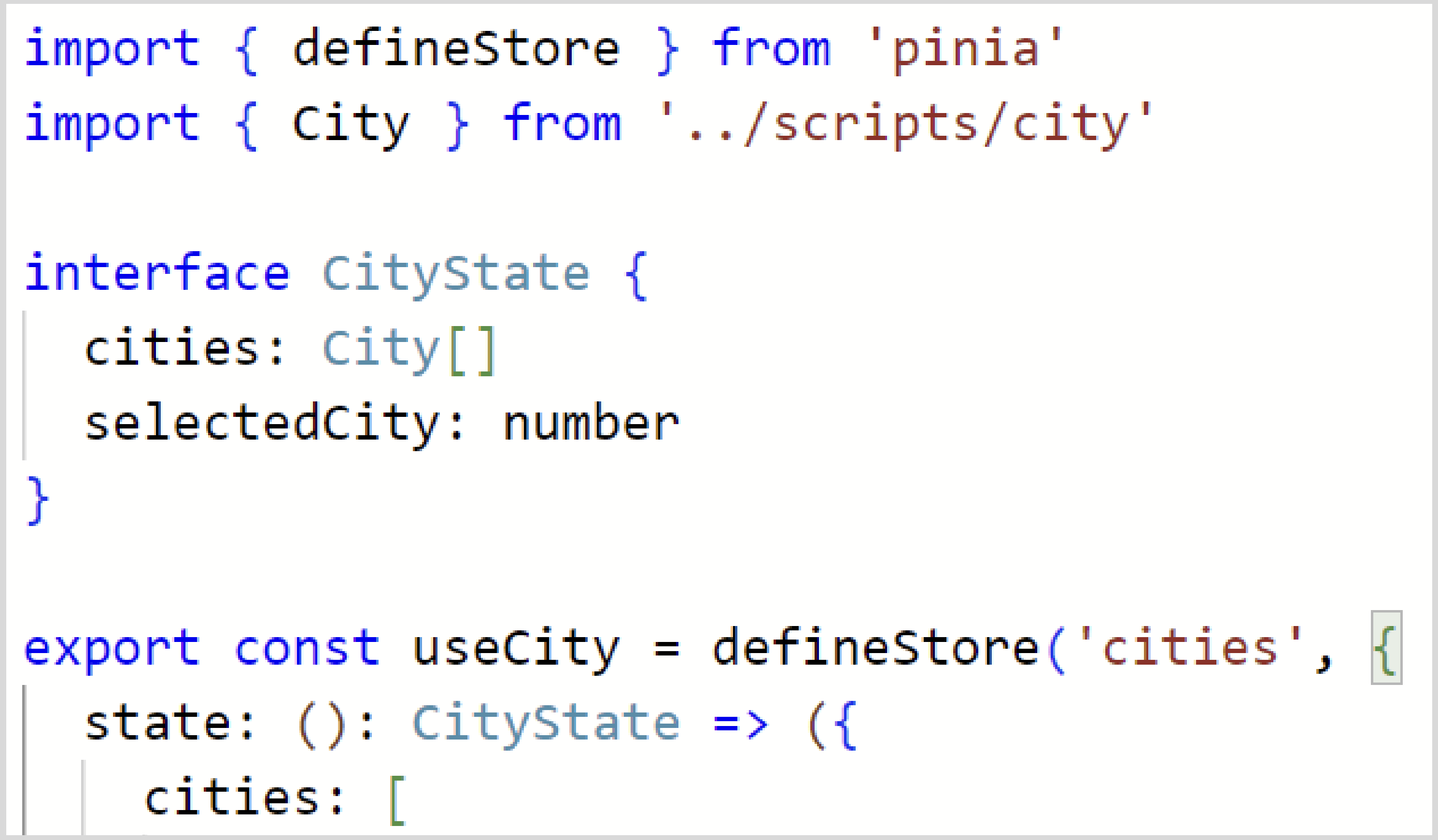


```
getters: {
  filteredWeathers: (state): Weather[][] => {
    switch (state.selectedPeriod) {
      case periodsTemp[0]: // Hoy
        const todayWeathers = state.weathers.filter(weather =>
          DateTime.fromFormat(weather.dt, 'ff').hasSame(DateTime.now().toLocal(), 'day')
        );

        return [[state.weather, ...todayWeathers]]
      case periodsTemp[1]: // Mañana
        const tomorrowWeathers = state.weathers.filter(weather =>
          DateTime.fromFormat(weather.dt, 'ff').hasSame(DateTime.now().toLocal().plus({ days: 1 }), 'day')
        );

        return [tomorrowWeathers]
      case periodsTemp[2]: // Próximos 5 días

      const results = [];
      for (let i = 1; i <= 5; i++) {
        const daily = state.weathers.filter(weather =>
          DateTime.fromFormat(weather.dt, 'ff').hasSame(DateTime.now().toLocal().plus({ days: i }), 'day')
        );
        results.push(daily);
      }
      return results;
```

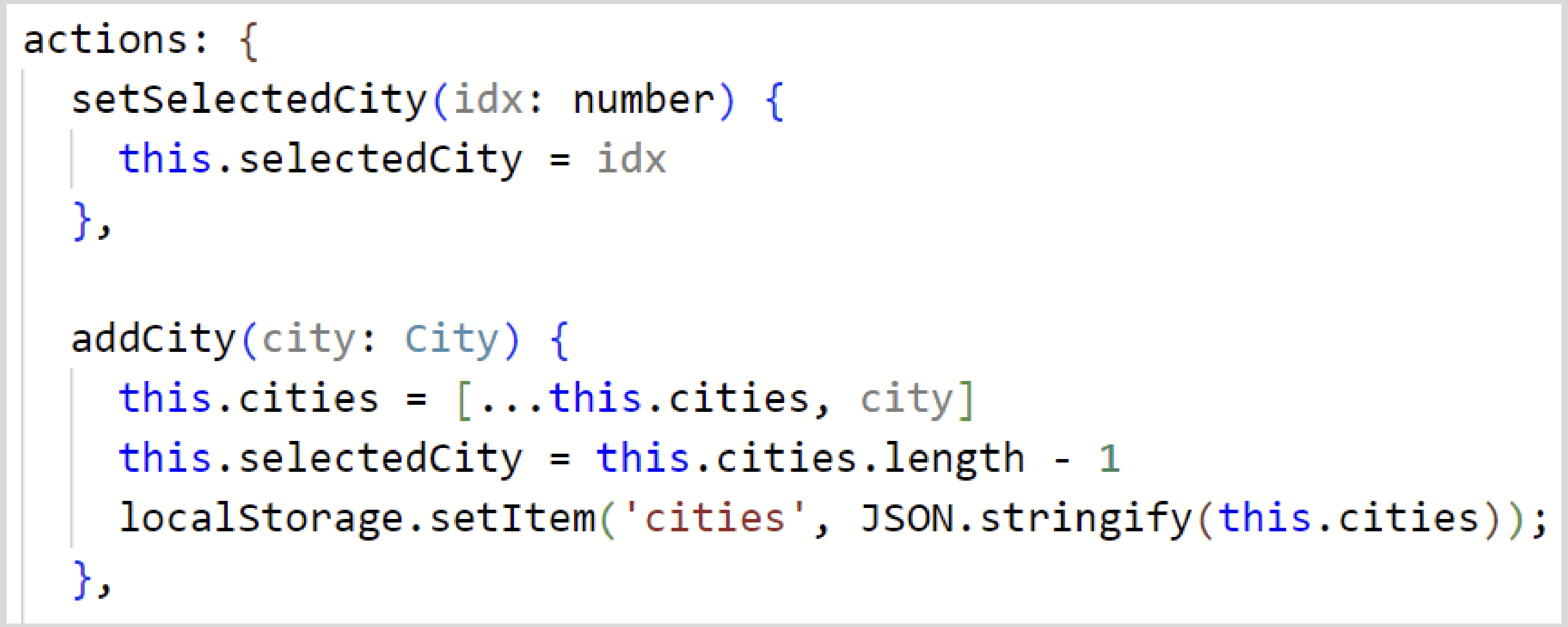
Store: airPollution.ts y city.ts



```
import { defineStore } from 'pinia'
import { DateTime } from 'luxon'
import { AirPollution } from '../scripts/airPollution'
import { PeriodPol, periodsPol } from '../scripts/constants'

interface AirPollutionState {
  airPollution: AirPollution
  airPollutions: AirPollution[]
  selectedPeriod: PeriodPol
}

export const useAirPollution = defineStore('airPollution', {
  state: (): AirPollutionState => ({
    airPollution: {
      dt: '',
      aqi: 0,
      co: 0,
      no: 0,
      no2: 0,
      o3: 0,
      so2: 0,
      pm2_5: 0,
      pm10: 0,
      nh3: 0
    },
    airPollutions: [],
    selectedPeriod: 'Ahora',
  }),

  actions: {
    async fetchAirPollution(latitude: number, longitude: number) {
```

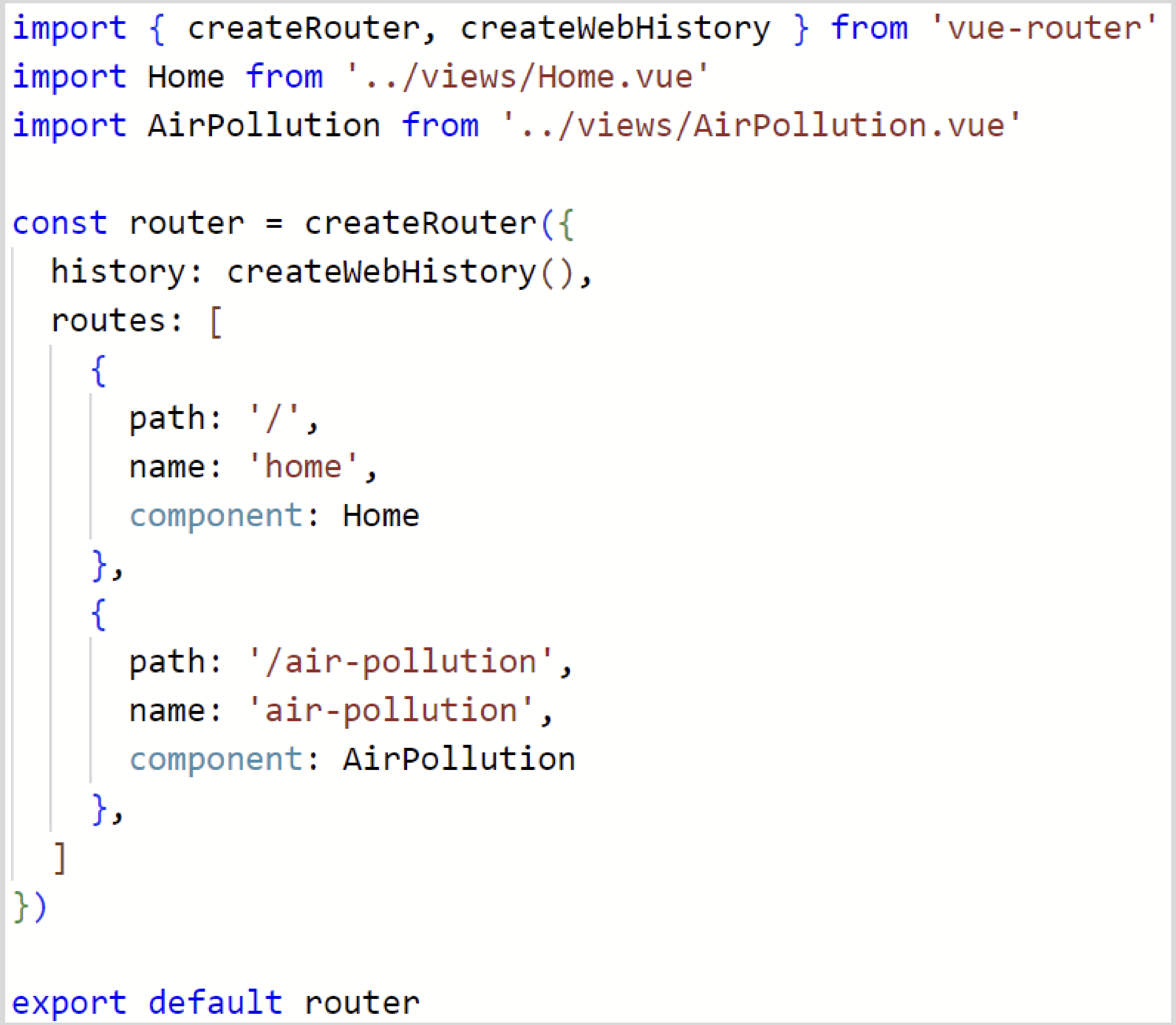


```
import { defineStore } from 'pinia'
import { City } from '../scripts/city'

interface CityState {
  cities: City[]
  selectedCity: number
}

export const useCity = defineStore('cities', {
  state: (): CityState => ({
    cities: [
```



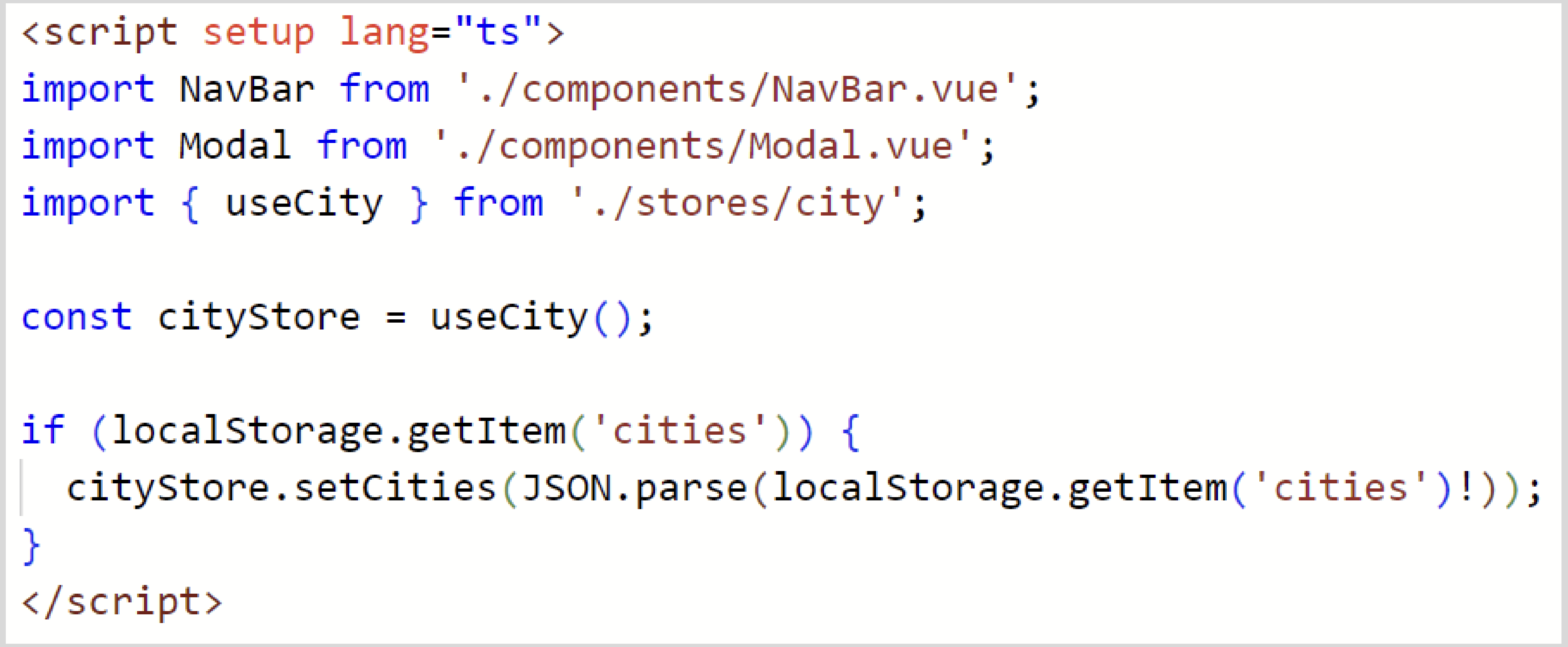
```
actions: {
  setSelectedCity(idx: number) {
    this.selectedCity = idx
  },

  addCity(city: City) {
    this.cities = [...this.cities, city]
    this.selectedCity = this.cities.length - 1
    localStorage.setItem('cities', JSON.stringify(this.cities));
  },
```

Router

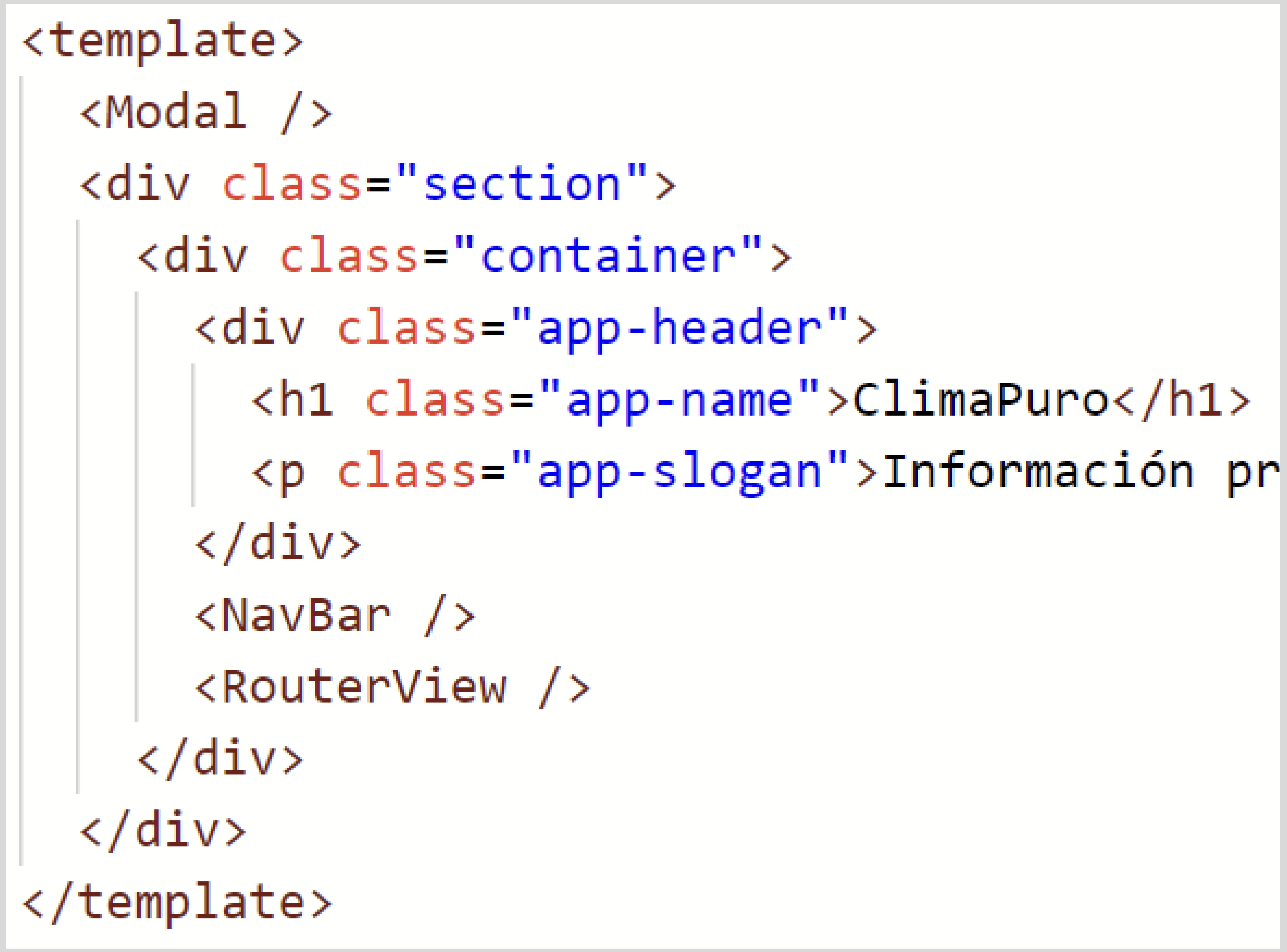


```
import { createRouter, createWebHistory } from 'vue-router'
import Home from '../views/Home.vue'
import AirPollution from '../views/AirPollution.vue'

const router = createRouter({
  history: createWebHistory(),
  routes: [
    {
      path: '/',
      name: 'home',
      component: Home
    },
    {
      path: '/air-pollution',
      name: 'air-pollution',
      component: AirPollution
    },
  ]
})

export default router
```

Componentes: App.vue

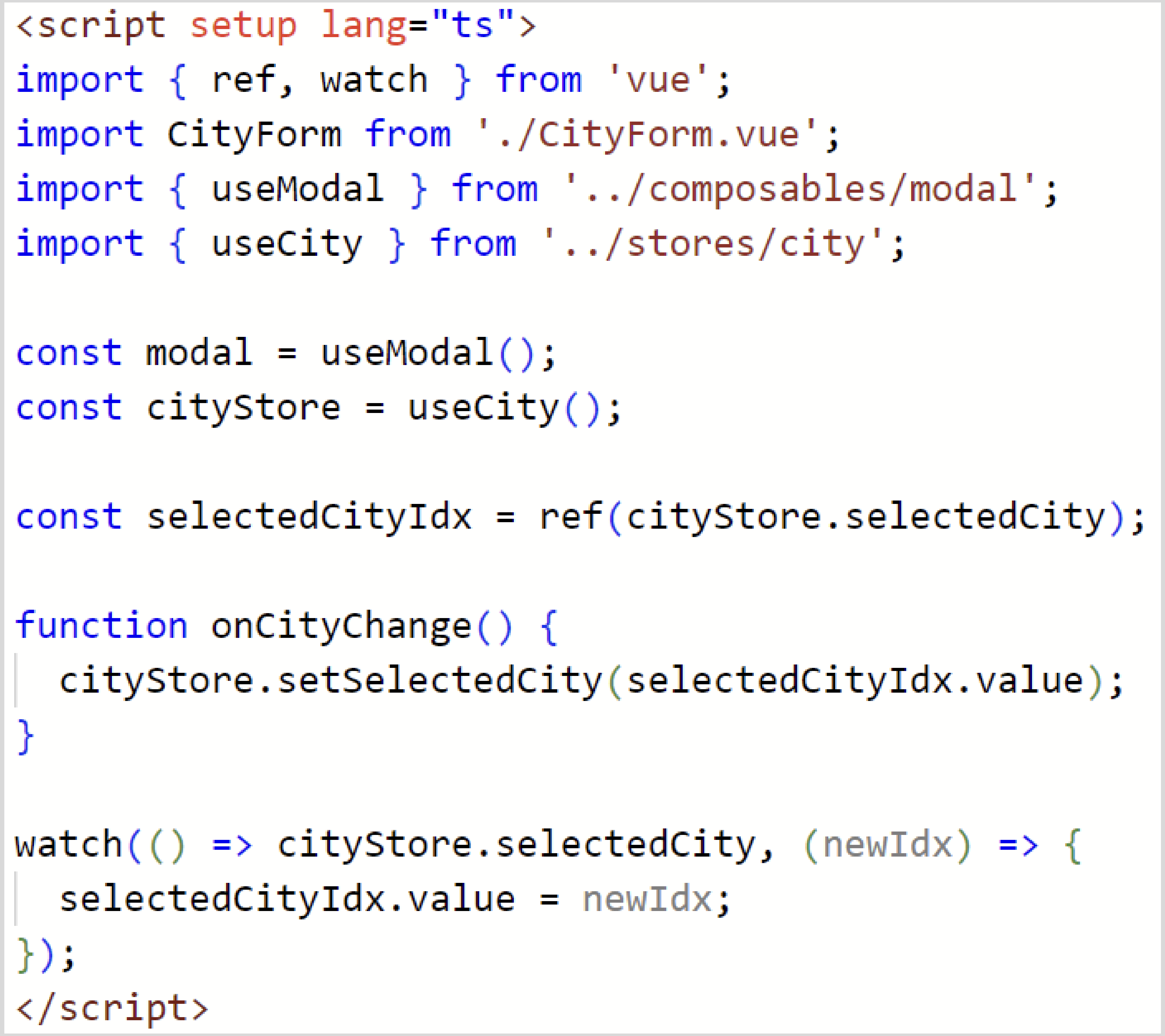


```
<script setup lang="ts">
import NavBar from './components/NavBar.vue';
import Modal from './components/Modal.vue';
import { useCity } from './stores/city';

const cityStore = useCity();

if (localStorage.getItem('cities')) {
  cityStore.setCities(JSON.parse(localStorage.getItem('cities')!));
}
</script>
```



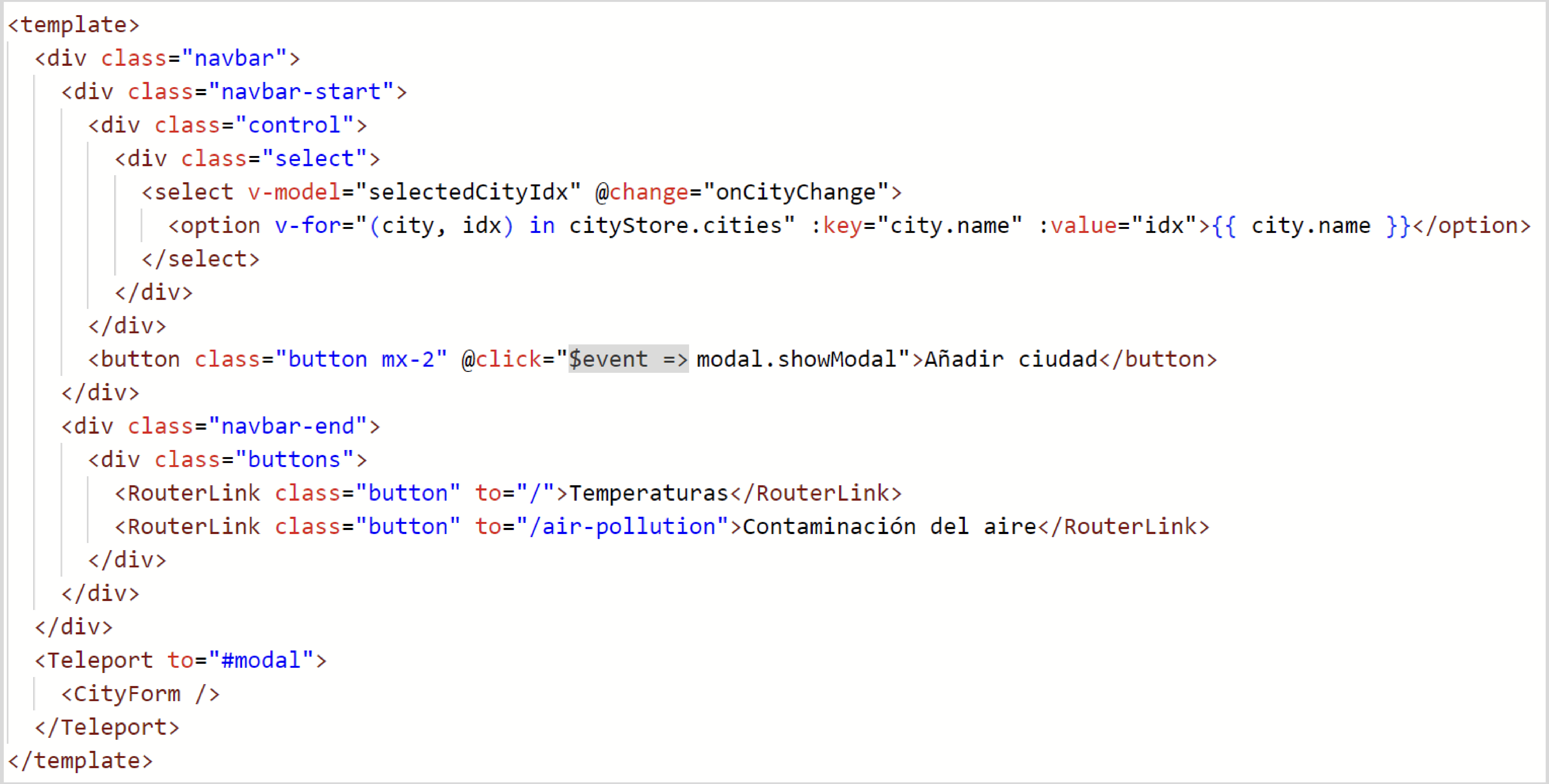
```
<template>
  <Modal />
  <div class="section">
    <div class="container">
      <div class="app-header">
        <h1 class="app-name">ClimaPuro</h1>
        <p class="app-slogan">Información pr
      </div>
      <NavBar />
      <RouterView />
    </div>
  </div>
</template>
```

Componentes: NavBar.vue



```
<script setup lang="ts">
import { ref, watch } from 'vue';
import CityForm from './CityForm.vue';
import { useModal } from '../composables/modal';
import { useCity } from '../stores/city';

const modal = useModal();
const cityStore = useCity();

const selectedCityIdx = ref(cityStore.selectedCity);

function onCityChange() {
  cityStore.setSelectedCity(selectedCityIdx.value);
}

watch(() => cityStore.selectedCity, (newIdx) => {
  selectedCityIdx.value = newIdx;
});
</script>
```

Componentes: NavBar.vue



```
<template>
  <div class="navbar">
    <div class="navbar-start">
      <div class="control">
        <div class="select">
          <select v-model="selectedCityIdx" @change="onCityChange">
            <option v-for="(city, idx) in cityStore.cities" :key="city.name" :value="idx">{{ city.name }}</option>
          </select>
        </div>
      </div>
      <button class="button mx-2" @click="$event => modal.showModal">Añadir ciudad</button>
    </div>
    <div class="navbar-end">
      <div class="buttons">
        <RouterLink class="button" to="/">Temperaturas</RouterLink>
        <RouterLink class="button" to="/air-pollution">Contaminación del aire</RouterLink>
      </div>
    </div>
  </div>
  <Teleport to="#modal">
    <CityForm />
  </Teleport>
</template>
```

Componentes: Modal.vue

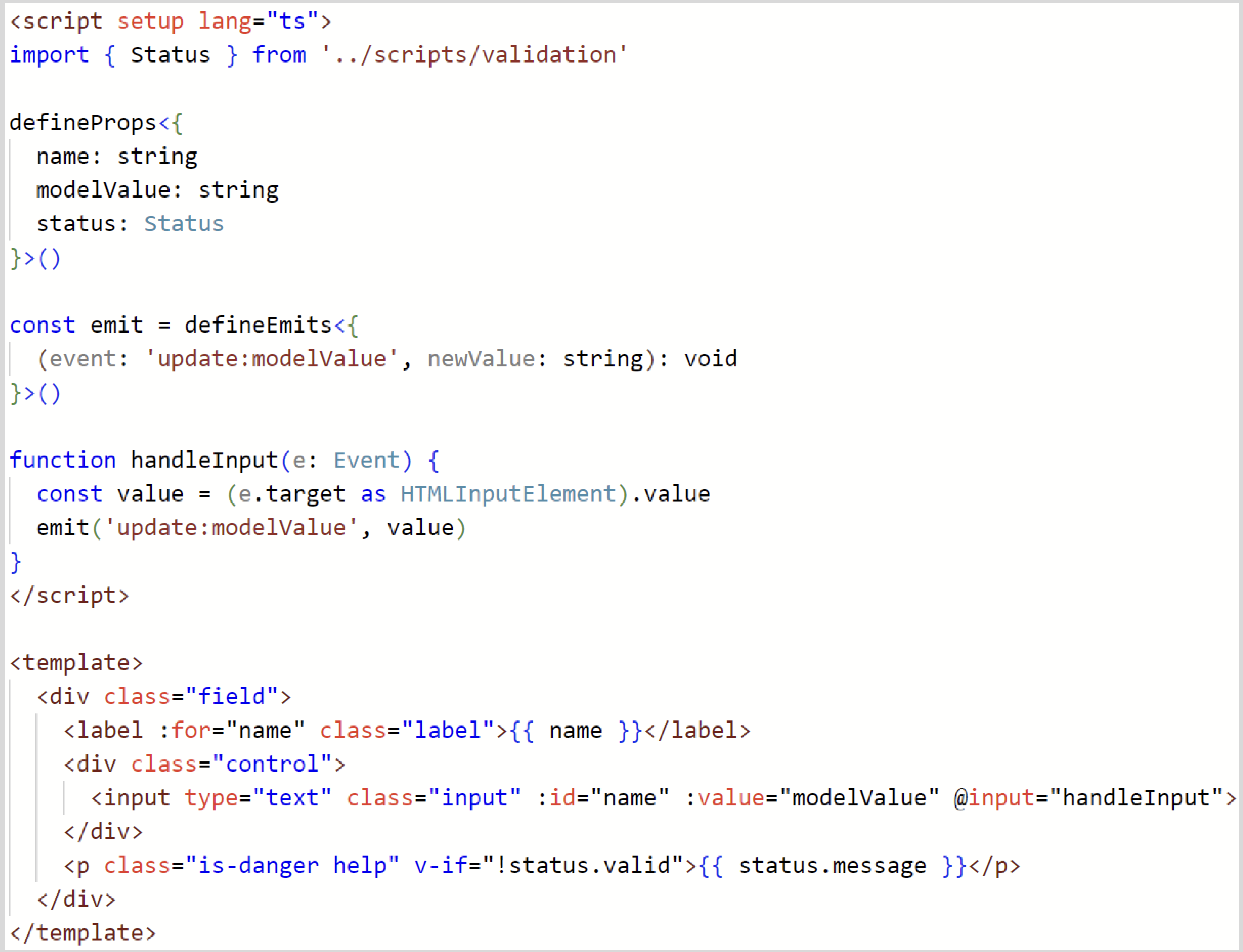
```ts
import { ref } from 'vue';

const show = ref(false);

export function useModal () {
  return {
    show,
    showModal: () => show.value = true,
    hideModal: () => show.value = false,
  }
}
```

```vue
<script setup lang="ts">
import { computed } from 'vue';
import { useModal } from '../composables/modal';

const modal = useModal();

const modalStyle = computed(() => {
  return {
    display: modal.show.value ? 'block' : 'none',
  };
});
</script>

<template>
  <div class="modal" style="color: white" :style="modalStyle">
    <div class="modal-background">
      <div class="modal-content">
        <div id="modal"></div>
      </div>
    </div>
    <button class="modal-close is-large" @click="$event => modal.hideModal"></button>
  </div>
</template>
```

Componentes: CityForm.vue

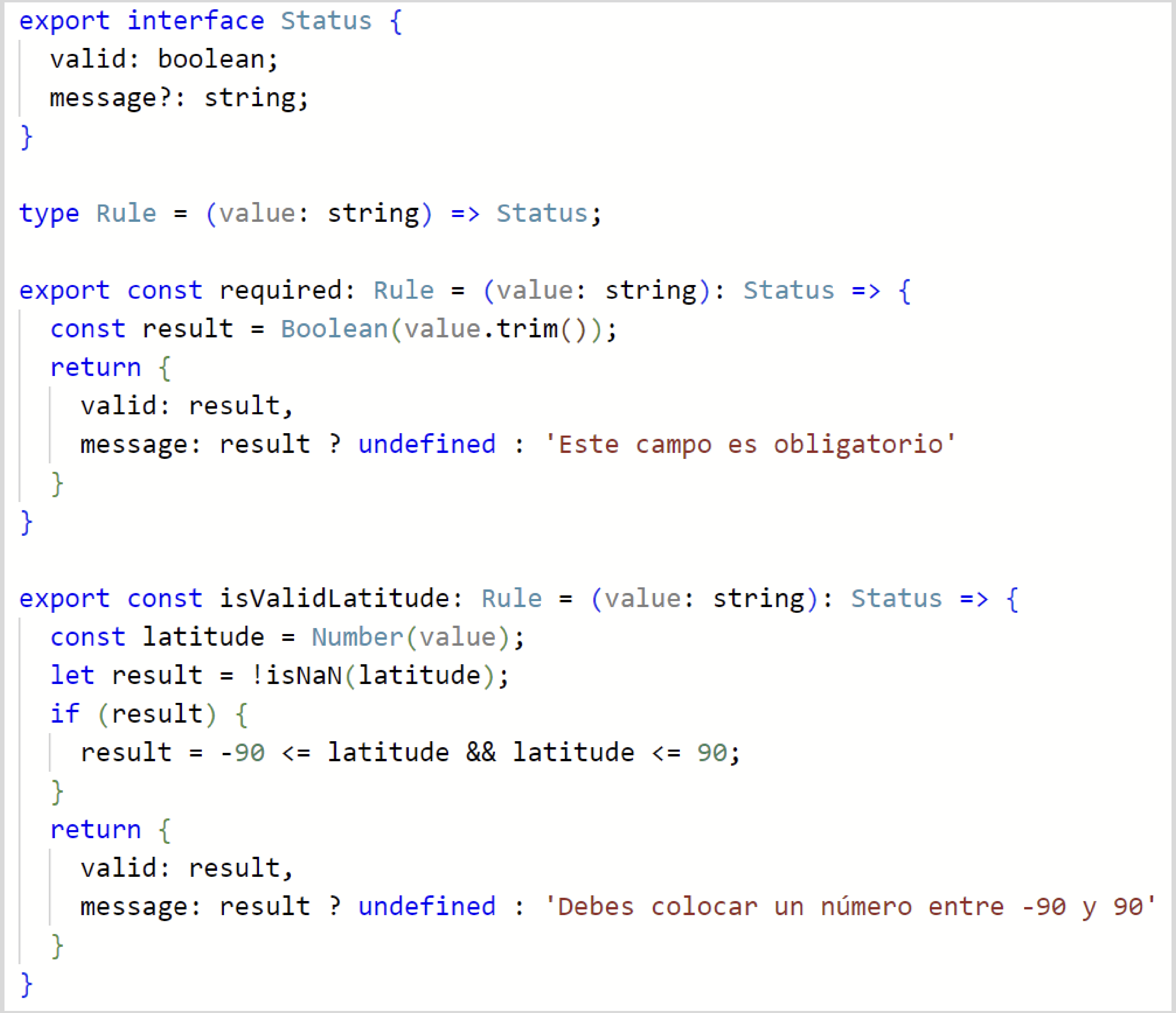
```
<template>
  <form class="form" @submit.prevent="handleSubmit">
    <FormInput name="Nombre de la ciudad" v-model="name" :status="nameStatus" />
    <!-- v-model is equivalent to:
        <CustomInput
          :modelValue="searchText"
          @update:modelValue="newValue => searchText = newValue"
        /> -->
    <FormInput name="País de la ciudad" v-model="country" :status="countryStatus" />
    <FormInput name="Latitud" v-model="latitude" :status="latitudeStatus" />
    <FormInput name="Longitud" v-model="longitude" :status="longitudeStatus" />
    <FormInput name="URL de la imagen" v-model="imageURL" :status="imageURLStatus" />
    <button class="button is-primary" :disabled="isInvalid">Añadir</button>
  </form>
</template>
```

Componentes: FormInput.vue

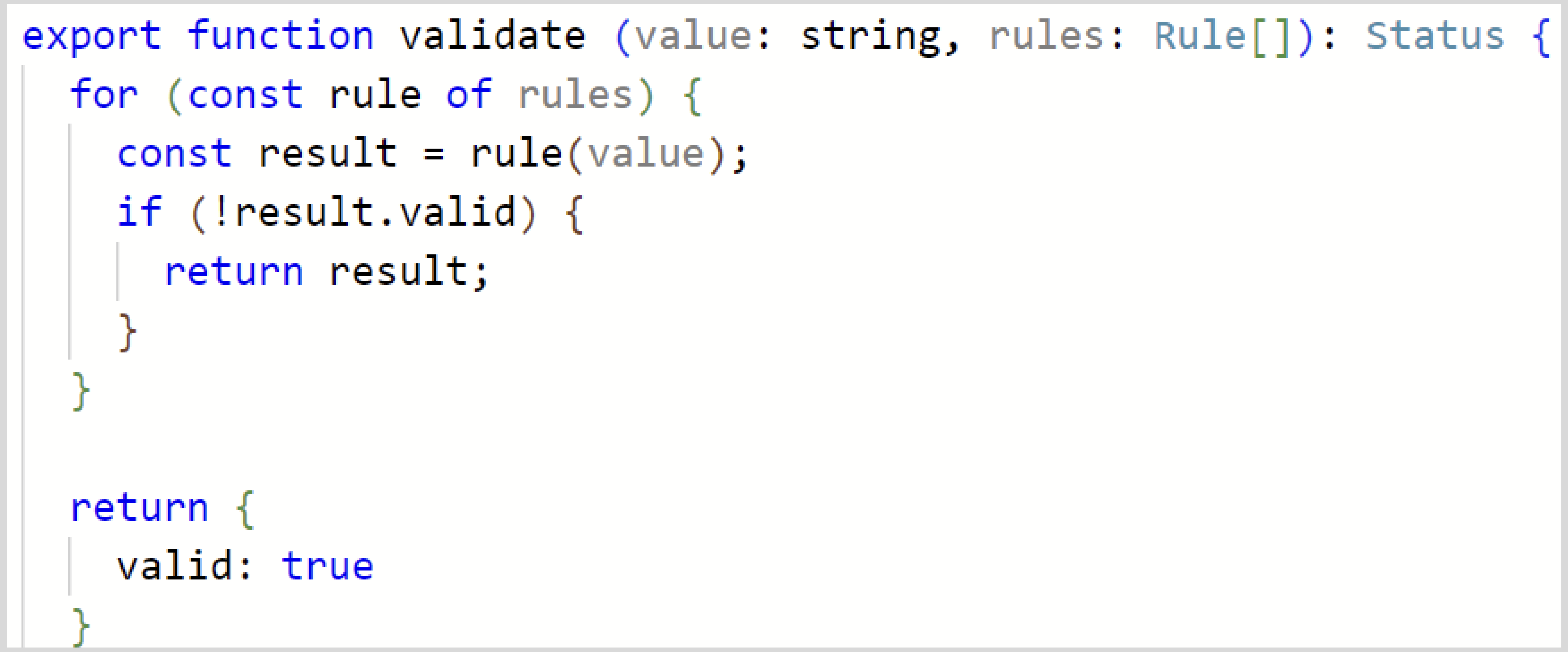


```vue
<script setup lang="ts">
import { Status } from '../scripts/validation'

defineProps<{
  name: string
  modelValue: string
  status: Status
}>()

const emit = defineEmits<{
  (event: 'update:modelValue', newValue: string): void
}>()

function handleInput(e: Event) {
  const value = (e.target as HTMLInputElement).value
  emit('update:modelValue', value)
}
</script>

<template>
  <div class="field">
    <label :for="name" class="label">{{ name }}</label>
    <div class="control">
      <input type="text" class="input" :id="name" :value="modelValue" @input="handleInput">
    </div>
    <p class="is-danger help" v-if="!status.valid">{{ status.message }}</p>
  </div>
</template>
```

validation.ts

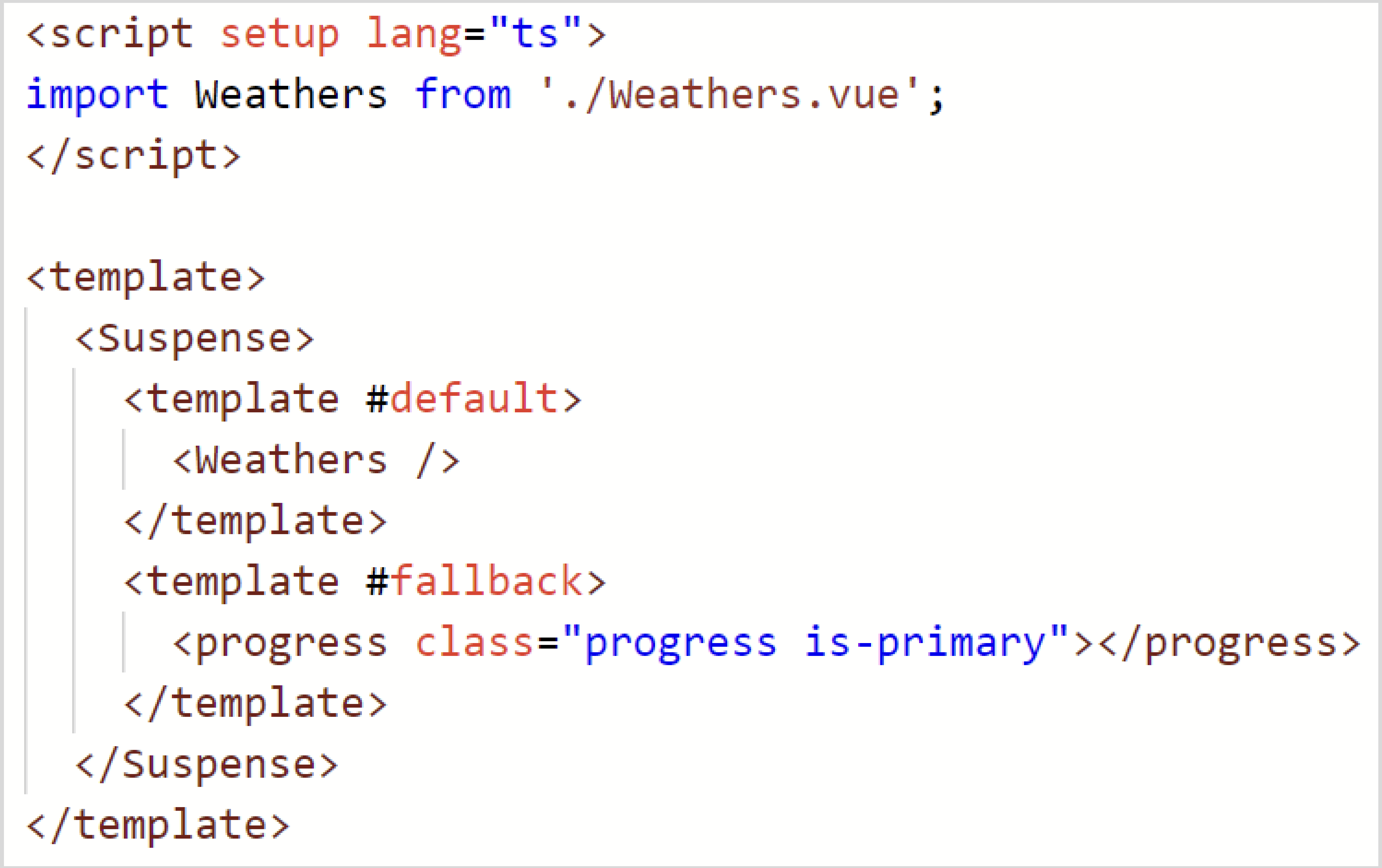


```typescript
export interface Status {
  valid: boolean;
  message?: string;
}

type Rule = (value: string) => Status;

export const required: Rule = (value: string): Status => {
  const result = Boolean(value.trim());
  return {
    valid: result,
    message: result ? undefined : 'Este campo es obligatorio'
  }
}

export const isValidLatitude: Rule = (value: string): Status => {
  const latitude = Number(value);
  let result = !isNaN(latitude);
  if (result) {
    result = -90 <= latitude && latitude <= 90;
  }
  return {
    valid: result,
    message: result ? undefined : 'Debes colocar un número entre -90 y 90'
  }
}
```



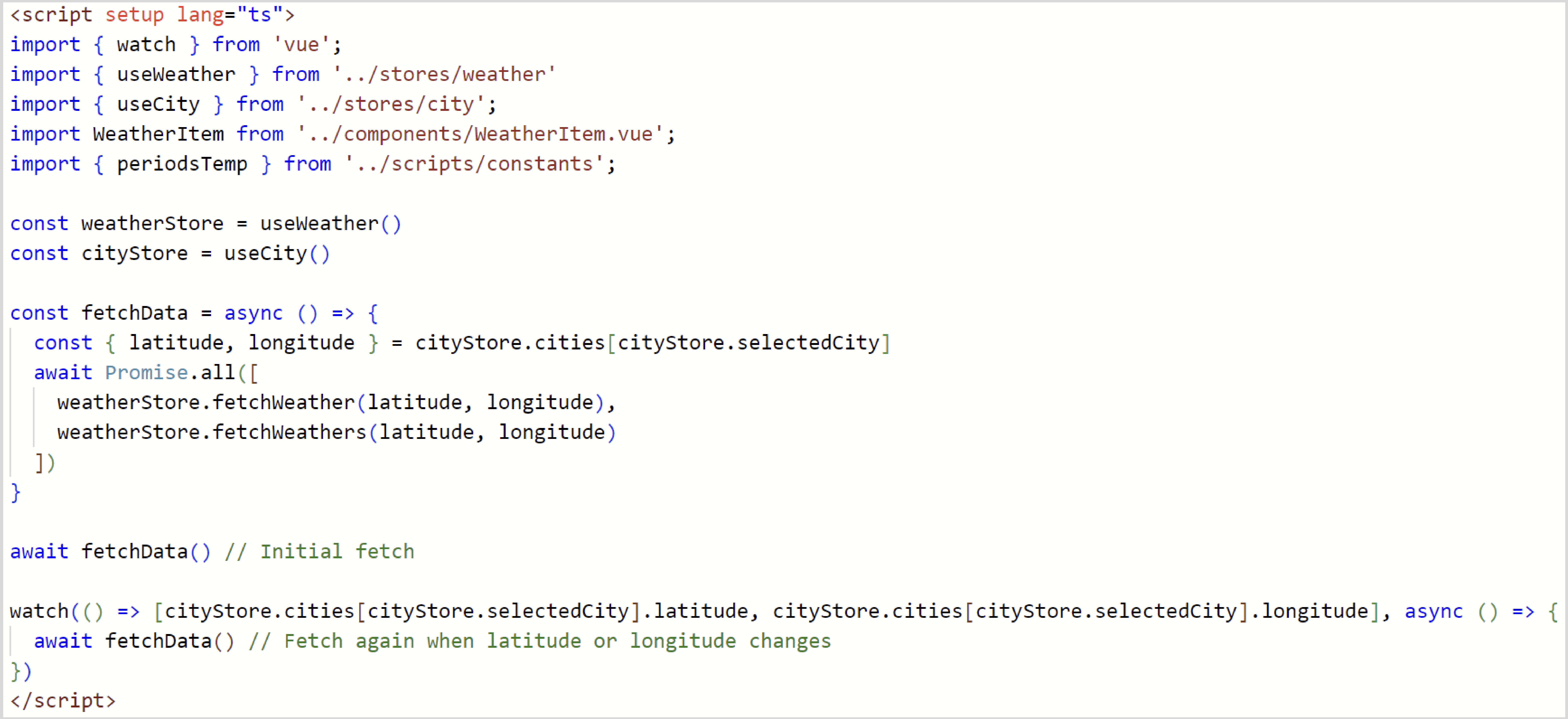
```typescript
export function validate (value: string, rules: Rule[]): Status {
  for (const rule of rules) {
    const result = rule(value);
    if (!result.valid) {
      return result;
    }
  }

  return {
    valid: true
  }
```

Componentes: Home.vue



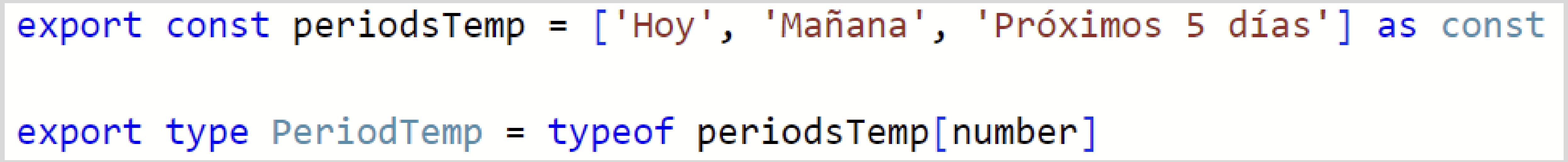
```
<script setup lang="ts">
import Weathers from './Weathers.vue';
</script>

<template>
  <Suspense>
    <template #default>
      <Weathers />
    </template>
    <template #fallback>
      <progress class="progress is-primary"></progress>
    </template>
  </Suspense>
</template>
```

Componentes: Weathers.vue

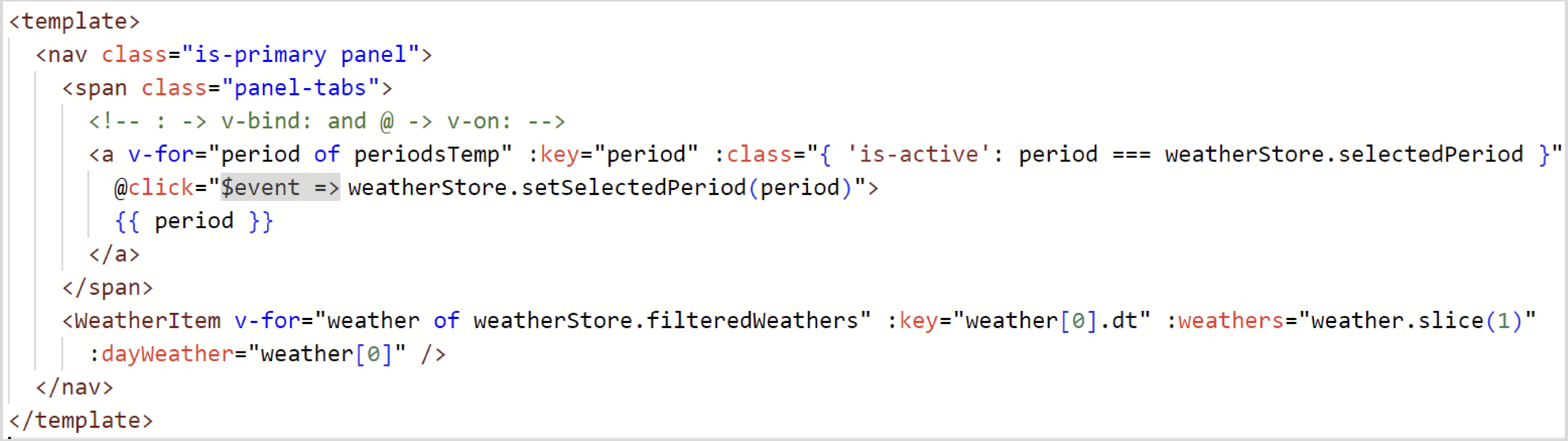


```
<script setup lang="ts">
import { watch } from 'vue';
import { useWeather } from '../stores/weather'
import { useCity } from '../stores/city';
import WeatherItem from '../components/WeatherItem.vue';
import { periodsTemp } from '../scripts/constants';

const weatherStore = useWeather()
const cityStore = useCity()

const fetchData = async () => {
  const { latitude, longitude } = cityStore.cities[cityStore.selectedCity]
  await Promise.all([
    weatherStore.fetchWeather(latitude, longitude),
    weatherStore.fetchWeathers(latitude, longitude)
  ])
}

await fetchData() // Initial fetch

watch(() => [cityStore.cities[cityStore.selectedCity].latitude, cityStore.cities[cityStore.selectedCity].longitude], async () => {
  await fetchData() // Fetch again when latitude or longitude changes
})
</script>
```

Componentes: Weathers.vue

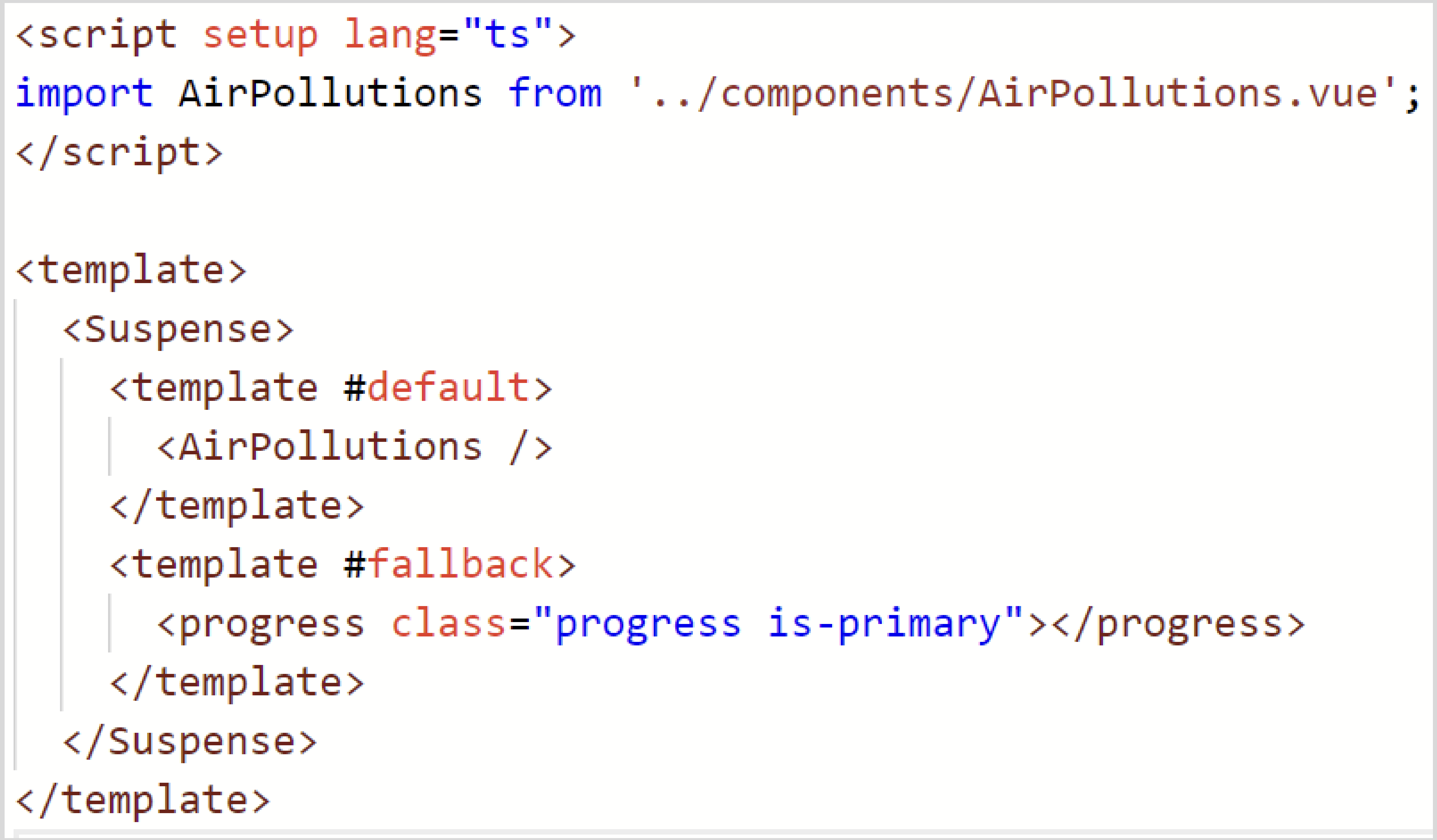


```
export const periodsTemp = ['Hoy', 'Mañana', 'Próximos 5 días'] as const

export type PeriodTemp = typeof periodsTemp[number]
```



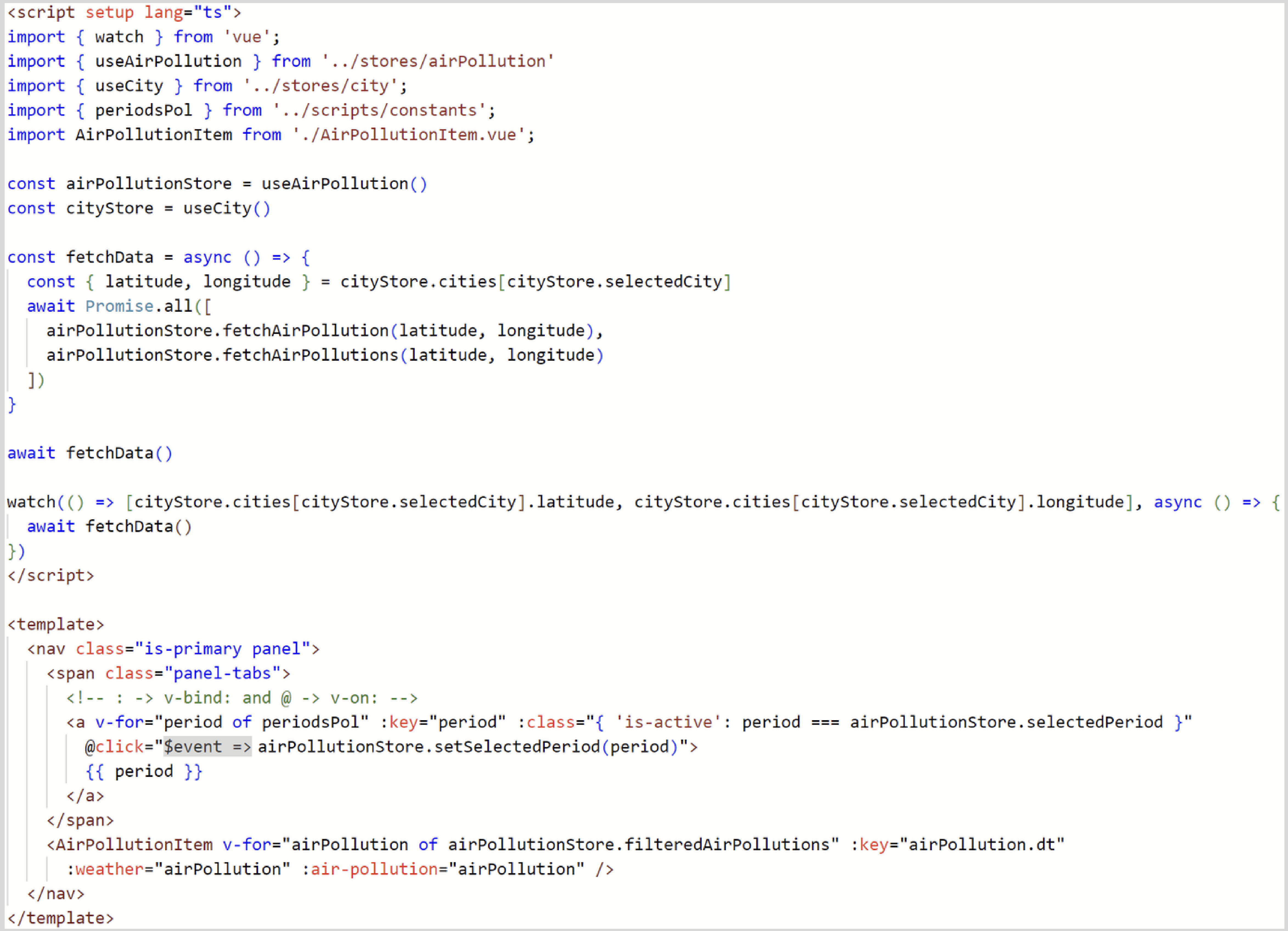
```
<template>
  <nav class="is-primary panel">
    <span class="panel-tabs">
      <!-- : -> v-bind: and @ -> v-on: -->
      <a v-for="period of periodsTemp" :key="period" :class="{ 'is-active': period === weatherStore.selectedPeriod }"
        @click="$event => weatherStore.setSelectedPeriod(period)">
        {{ period }}
      </a>
    </span>
    <WeatherItem v-for="weather of weatherStore.filteredWeathers" :key="weather[0].dt" :weathers="weather.slice(1)"
      :dayWeather="weather[0]" />
  </nav>
</template>
```

Componentes: AirPollution.vue



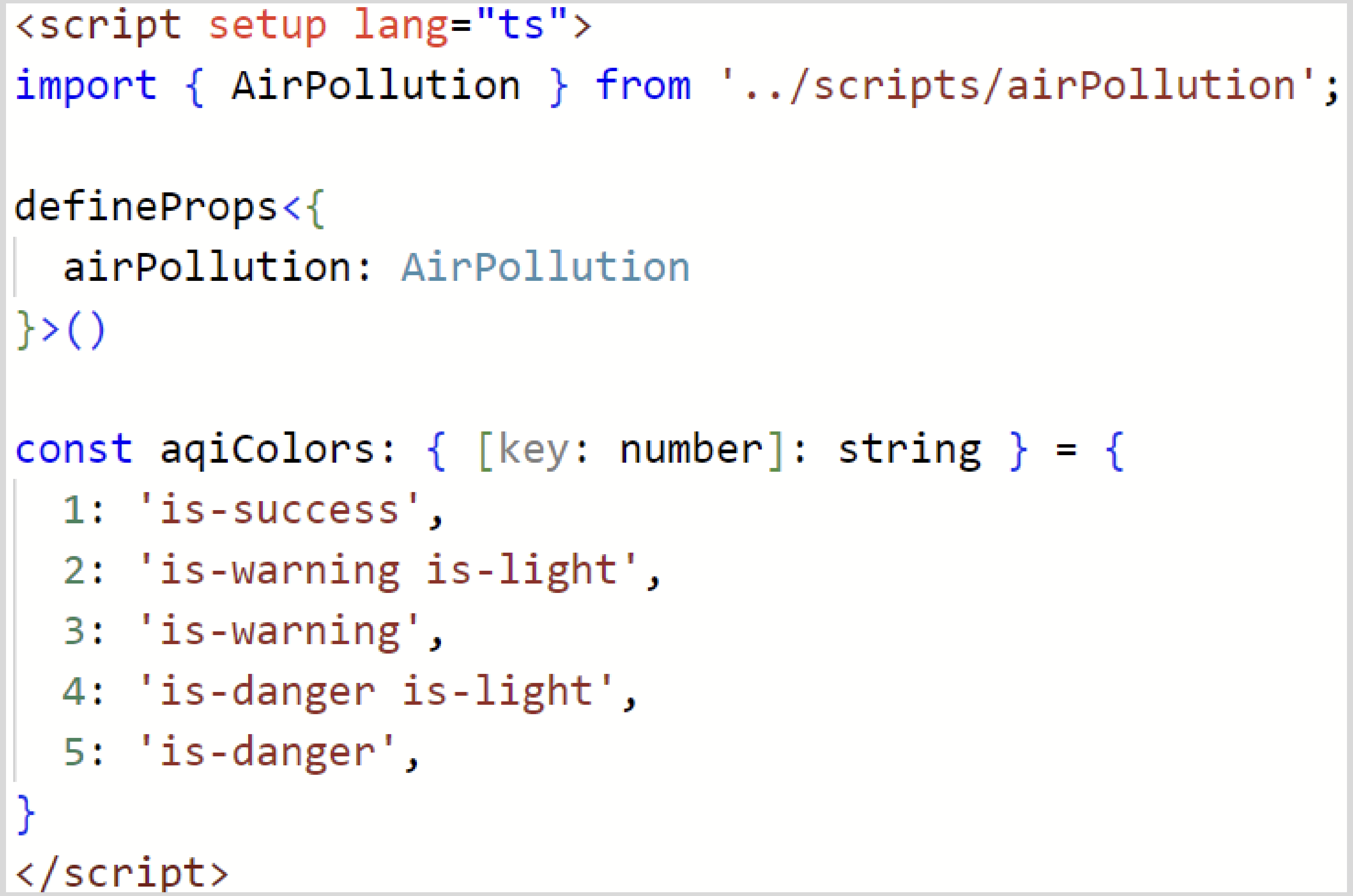
```
<script setup lang="ts">
import AirPollutions from '../components/AirPollutions.vue';
</script>

<template>
  <Suspense>
    <template #default>
      <AirPollutions />
    </template>
    <template #fallback>
      <progress class="progress is-primary"></progress>
    </template>
  </Suspense>
</template>
```

Componentes: AirPollutions.vue

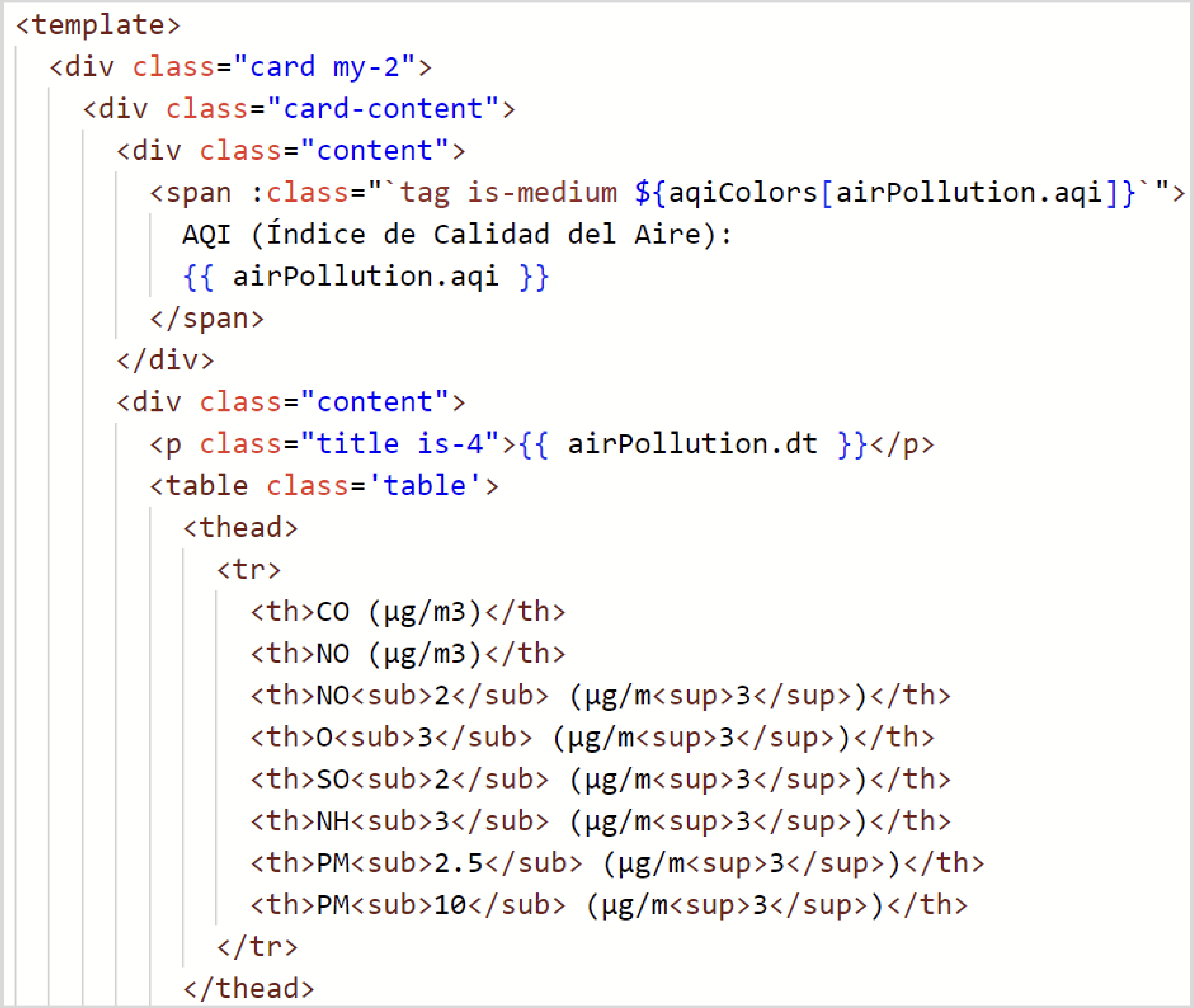


```
<script setup lang="ts">
import { watch } from 'vue';
import { useAirPollution } from '../stores/airPollution'
import { useCity } from '../stores/city';
import { periodsPol } from '../scripts/constants';
import AirPollutionItem from './AirPollutionItem.vue';

const airPollutionStore = useAirPollution()
const cityStore = useCity()

const fetchData = async () => {
  const { latitude, longitude } = cityStore.cities[cityStore.selectedCity]
  await Promise.all([
    airPollutionStore.fetchAirPollution(latitude, longitude),
    airPollutionStore.fetchAirPollutions(latitude, longitude)
  ])
}

await fetchData()

watch(() => [cityStore.cities[cityStore.selectedCity].latitude, cityStore.cities[cityStore.selectedCity].longitude], async () => {
  await fetchData()
})
</script>

<template>
  <nav class="is-primary panel">
    <span class="panel-tabs">
      <!-- : -> v-bind: and @ -> v-on: -->
      <a v-for="period of periodsPol" :key="period" :class="{ 'is-active': period === airPollutionStore.selectedPeriod }"
        @click="$event => airPollutionStore.setSelectedPeriod(period)">
        {{ period }}
      </a>
    </span>
    <AirPollutionItem v-for="airPollution of airPollutionStore.filteredAirPollutions" :key="airPollution.dt"
      :weather="airPollution" :air-pollution="airPollution" />
  </nav>
</template>
```

Componentes: AirPollutionItem.vue



```vue
<script setup lang="ts">
import { AirPollution } from '../scripts/airPollution';

defineProps<{
  airPollution: AirPollution
}>()

const aqiColors: { [key: number]: string } = {
  1: 'is-success',
  2: 'is-warning is-light',
  3: 'is-warning',
  4: 'is-danger is-light',
  5: 'is-danger',
}
</script>
```

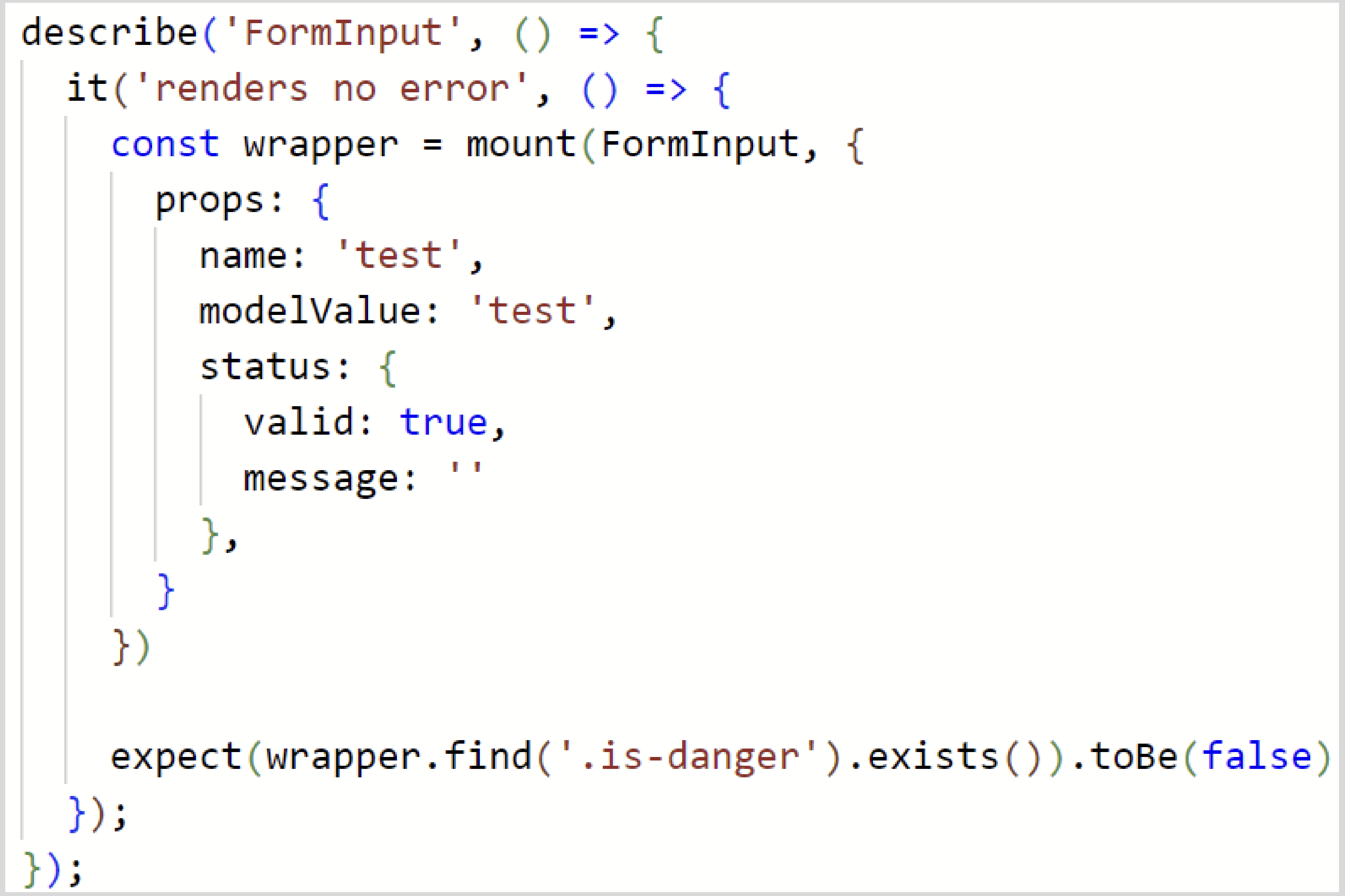


```vue
<template>
  <div class="card my-2">
    <div class="card-content">
      <div class="content">
        <span :class="`tag is-medium ${aqiColors[airPollution.aqi]}`">
          AQI (Índice de Calidad del Aire):
          {{ airPollution.aqi }}
        </span>
      </div>
      <div class="content">
        <p class="title is-4">{{ airPollution.dt }}</p>
        <table class='table'>
          <thead>
            <tr>
              <th>CO (µg/m3)</th>
              <th>NO (µg/m3)</th>
              <th>NO<sub>2</sub> (µg/m<sup>3</sup>)</th>
              <th>O<sub>3</sub> (µg/m<sup>3</sup>)</th>
              <th>SO<sub>2</sub> (µg/m<sup>3</sup>)</th>
              <th>NH<sub>3</sub> (µg/m<sup>3</sup>)</th>
              <th>PM<sub>2.5</sub> (µg/m<sup>3</sup>)</th>
              <th>PM<sub>10</sub> (µg/m<sup>3</sup>)</th>
            </tr>
          </thead>
```

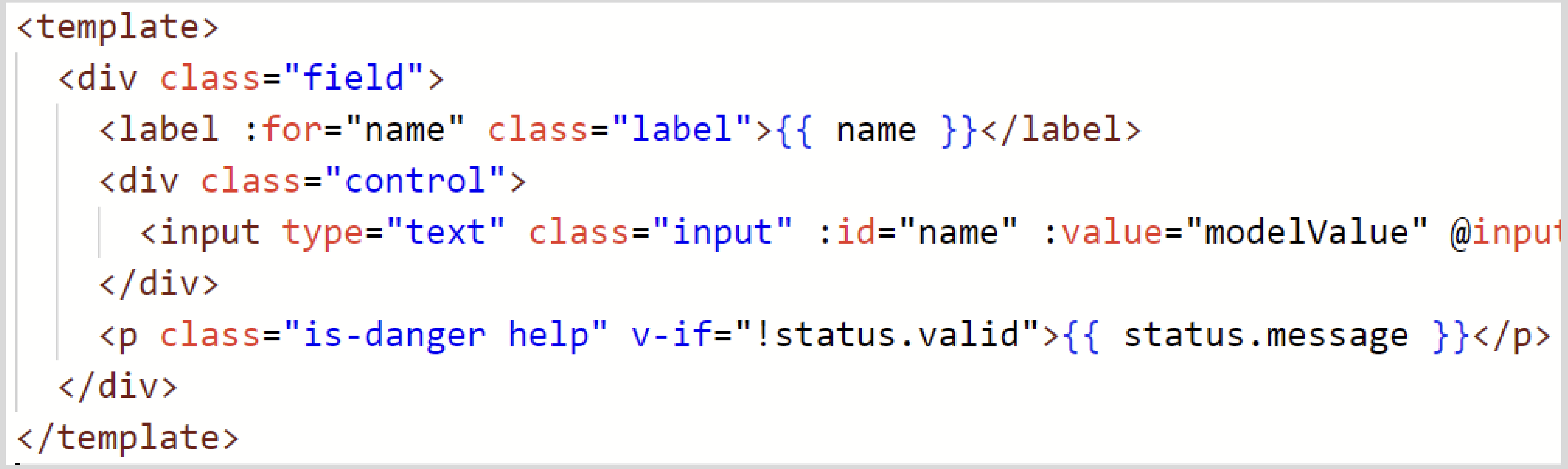
Testing: Vitest

Vitest

A blazing fast unit-test framework powered by Vite

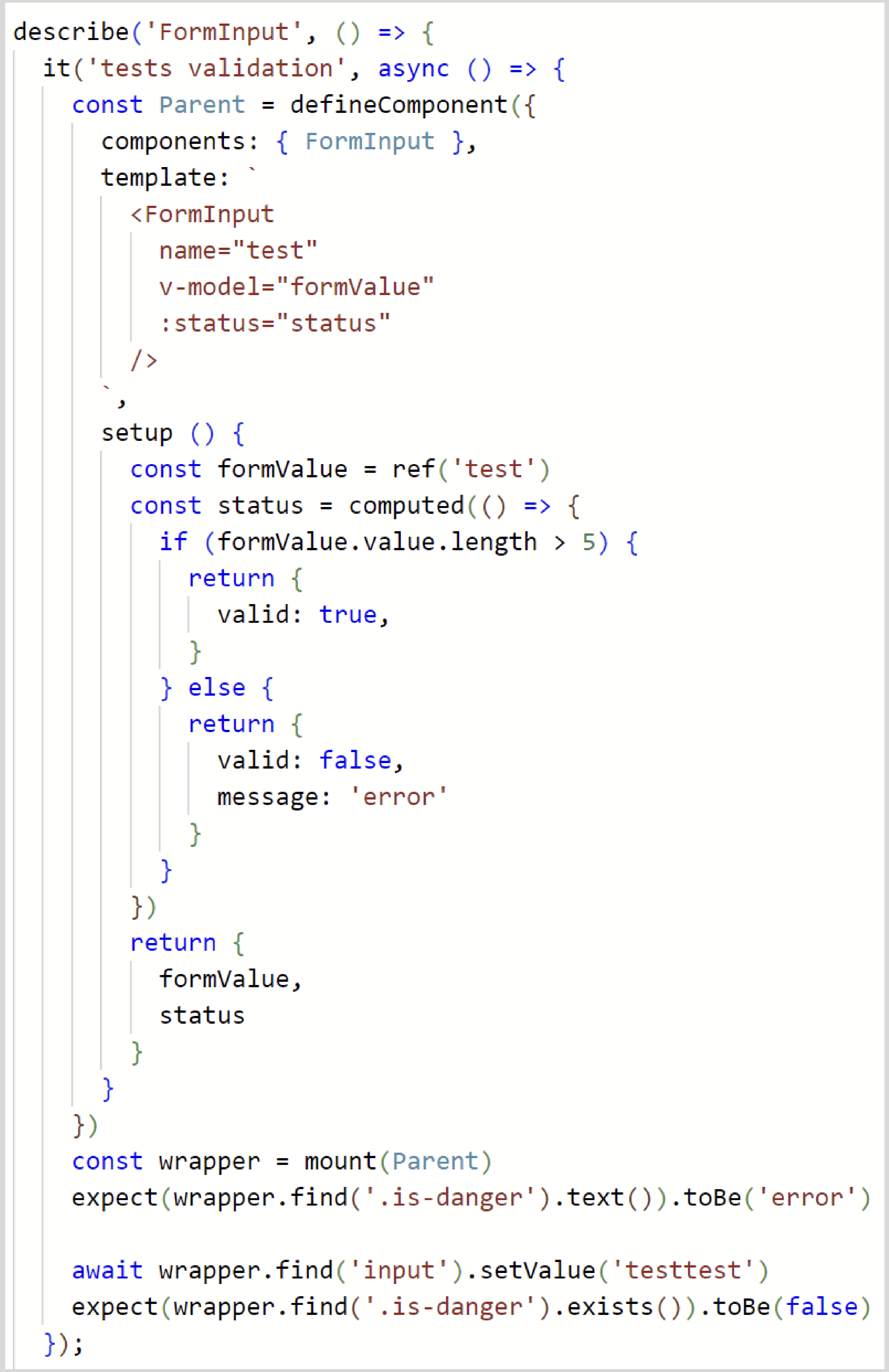


```
describe('FormInput', () => {
  it('renders no error', () => {
    const wrapper = mount(FormInput, {
      props: {
        name: 'test',
        modelValue: 'test',
        status: {
          valid: true,
          message: ''
        },
      }
    })

    expect(wrapper.find('.is-danger').exists()).toBe(false)
  });
});
```



```
<template>
  <div class="field">
    <label :for="name" class="label">{{ name }}</label>
    <div class="control">
      <input type="text" class="input" :id="name" :value="modelValue" @inpu
    </div>
    <p class="is-danger help" v-if="!status.valid">{{ status.message }}</p>
  </div>
</template>
```

Testing: Vitest



```
describe('FormInput', () => {
  it('tests validation', async () => {
    const Parent = defineComponent({
      components: { FormInput },
      template: `
        <FormInput
          name="test"
          v-model="formValue"
          :status="status"
        />
      `,
      setup () {
        const formValue = ref('test')
        const status = computed(() => {
          if (formValue.value.length > 5) {
            return {
              valid: true,
            }
          } else {
            return {
              valid: false,
              message: 'error'
            }
          }
        })
        return {
          formValue,
          status
        }
      }
    })
    const wrapper = mount(Parent)
    expect(wrapper.find('.is-danger').text()).toBe('error')

    await wrapper.find('input').setValue('testtest')
    expect(wrapper.find('.is-danger').exists()).toBe(false)
  });
```

CONCLUSIONES

OpenWeather

TS

Equipo 09