```
<template>
  <!-- ToDo:
  1. Berichtigung Vorschau / Fehlersuche -->
  <div style="margin-left: 30px; display: flex; align-items: center">
    <scale-icon-content-clock accessibility-title="clock" style="padding-
right: 10px" /><b
      >Termin / -serie</b
    >
  </div>
  {{ errorOption }}
  <div style="padding: 10px; padding-bottom: 30px; margin-left: 40px">
    <scale-radio-button-group class="radio-group">
      <scale-radio-button
        label="Sofortversand"
        name="standard"
        value="0"
        @scale-change="terminRadio"
      />
      <div class="flex-container" style="gap: 100px">
        <scale-radio-button label="Termin" value="1" name="standard" @scale-
change="terminRadio" />
        <div class="flex-container" style="gap: 10px">
          <scale-text-field
            id="dateField"
            label="Versandtermin"
            style="width: 50%"
            type="date"
            v-model="dateFieldValue"
            :disabled="selectedOption !== '1'"
          />
          <scale-text-field
            id="timeField"
            label="Sendezeit"
            style="width: 50%"
            type="time"
            v-model="timeFieldValue"
            :disabled="selectedOption !== '1'"
          />
        </div>
      </div>
      <div class="flex-container" style="gap: 70px; margin-bottom: 20px">
        <scale-radio-button
          label="Terminserie"
          name="standard"
          value="2"
          @scale-change="terminRadio"
        />
        <scale-button
          icon-only
          size="small"
```

```
          variant="secondary"
          @click="openSerienmuster"
          :disabled="selectedOption !== '2' || patterns.length >= 5"
        >
          <scale-icon-action-add accessibility-title="Serienmuster festlegen"
/>
        </scale-button>
      </div>

      <div v-if="patterns.length > 0" class="patterns-container">
        <div v-for="(pattern, index) in sortedPatterns" :key="index"
class="pattern-tag">
          <scale-tag color="teal" type="strong" dismissable
@scaleClose="removePattern(index)">
            {{ pattern }}
          </scale-tag>
        </div>
      </div>
    </scale-radio-button-group>
  </div>
  <!-- Serienmuster Modal -->
  <scale-modal ref="serienmuster" heading="Serienmuster festlegen"
size="default">
    <div style="margin-left: 50px">
      <div class="inline-container">
        <scale-radio-button
          label="Wöchentlich"
          name="schedule"
          input-id="choice-weekly"
          :checked="schedule.weekly"
          @scale-change="setSchedule('weekly')"
        />
        <scale-radio-button
          label="Monatlich"
          name="schedule"
          input-id="choice-monthly"
          :checked="schedule.monthly"
          @scale-change="setSchedule('monthly')"
        />
      </div>

      <!-- Wöchentlich Dropdowns -->
      <div v-if="schedule.weekly">
        <div class="inline-container">
          <span>Jede/Alle</span>
          <scale-dropdown-select
            v-model="selectedWeek"
            style="width: 70px; justify-content: center; align-self: center"
          >
```

```
            <scale-dropdown-select-item v-for="week in weeks" :key="week"
:value="week">
              {{ week }}
            </scale-dropdown-select-item>
          </scale-dropdown-select>
          <span>Woche(n) am</span>
        </div>
        <div class="inline-container">
          <scale-checkbox
            v-for="(checked, day) in days"
            :key="day"
            :label="day.charAt(0).toUpperCase() + day.slice(1)"
            :checked="checked"
            @scale-change="days[day] = $event.detail.checked"
          />
        </div>
      </div>

      <!-- Monatlich Dropdowns -->
      <div v-if="schedule.monthly">
        <div class="inline-container">
          <span>Am</span>
          <scale-dropdown-select
            v-model="selectedOrdinal"
            style="width: 150px; justify-content: center; align-self: center"
          >
            <scale-dropdown-select-item v-for="ordinal in ordinals"
:key="ordinal" :value="ordinal">
              {{ ordinal }}
            </scale-dropdown-select-item>
          </scale-dropdown-select>

          <scale-dropdown-select
            v-model="selectedDayOfWeek"
            style="width: 150px; justify-content: center; align-self: center"
          >
            <scale-dropdown-select-item v-for="day in weekDays" :key="day"
:value="day">
              {{ day }}
            </scale-dropdown-select-item>
          </scale-dropdown-select>
        </div>
        <div class="inline-container">
          <span>jeden/alle</span>
          <scale-dropdown-select
            v-model="selectedMonthFrequency"
            style="width: 70px; justify-content: center; align-self: center"
          >
            <scale-dropdown-select-item v-for="month in months" :key="month"
:value="month">
```

```
            {{ month }}
          </scale-dropdown-select-item>
        </scale-dropdown-select>
        <span>Monat(e)</span>
      </div>
    </div>

    <div class="inline-container">
      <p>Sendezeit</p>
      <scale-text-field
        ref="sendTime"
        type="time"
        v-model="sendTime"
        label="Uhrzeit"
        style="flex-grow: 1; justify-content: center; align-items: center"
        step="900"
        @scale-change="sendTime = $event.detail.value"
      />
      <p>Uhr</p>
    </div>

    <div class="inline-container" style="width: auto">
      <p>Vorschau</p>
      <scale-tag color="teal" type="strong">
        {{ previewText }}
      </scale-tag>
    </div>

    <div
      class="inline-container"
      style="width: auto; align-self: center; justify-content: space-
between"
    >
      <scale-button :disabled="!canSave"
@click="acceptSerienmuster">Speichern</scale-button>
      <scale-button variant="secondary" @click="closeSerienmuster">
Abbrechen </scale-button>
    </div>
  </div>
</scale-modal>

<!-- Modal zur Bestätigung der Änderungen -->
<scale-modal ref="confirmActionModal" heading="Termin / -serien löschen"
size="default">
  <p>Beim Wechsel der Option werden vorhandene Einträge gelöscht. Möchtest
du fortfahren?</p>
  <div class="inline-container" style="padding-top: 20px; justify-content:
space-between">
    <scale-button variant="secondary" @click="applyChanges">Ja</scale-
button>
```

```
      <scale-button variant="primary"
@click="closeConfirmModal">Abbrechen</scale-button>
    </div>
  </scale-modal>
</template>

<script>
export default {
  data() {
    return {
      ordinals: ["ersten", "zweiten", "dritten", "vierten", "letzten"],
      months: Array.from({ length: 12 }, (_, i) => i + 1),
      selectedWeek: null,
      selectedOrdinal: "",
      selectedDayOfWeek: "",
      selectedMonthFrequency: null,
      dateFieldValue: "",
      timeFieldValue: "",
      selectedOption: "",
      previousOption: "0",
      schedule: { weekly: true, monthly: false },
      weeks: Array.from({ length: 52 }, (_, i) => i + 1),
      days: {
        monday: false,
        tuesday: false,
        wednesday: false,
        thursday: false,
        friday: false
      },
      selectedWeek: null,
      weekDays: ["Montag", "Dienstag", "Mittwoch", "Donnerstag", "Freitag"],
      sendTime: "",
      patterns: [],
      patternToDeleteIndex: null,
      errorMessage: "" // Error message to display
    };
  },
  computed: {
    sortedPatterns() {
      const weekDayOrder = ["Montag", "Dienstag", "Mittwoch", "Donnerstag",
"Freitag"];
      return this.patterns.slice().sort((a, b) => {
        const firstDayA = this.getFirstWeekDay(a);
        const firstDayB = this.getFirstWeekDay(b);
        return weekDayOrder.indexOf(firstDayA) -
weekDayOrder.indexOf(firstDayB);
      });
    },
    canSave() {
      const requiredFieldsFilled =
```

```
        (this.schedule.weekly || this.schedule.monthly) &&
        this.sendTime &&
        (this.schedule.weekly
          ? this.selectedDaysOfWeek.length > 0
          : this.selectedDay && this.selectedMonth);
      return requiredFieldsFilled;
    },
    selectedDays() {
      if (this.schedule.weekly) {
        return Object.keys(this.days)
          .filter((day) => this.days[day])
          .map((day) => this.weekDays[Object.keys(this.days).indexOf(day)])
          .join(", ");
      } else if (this.schedule.monthly) {
        return `${this.selectedOrdinal} ${this.selectedDayOfWeek}`;
      }
      return "";
    },
    selectedSchedule() {
      return this.schedule.weekly ? "Wöchentlich" : "Monatlich";
    },
    previewText() {
      if (this.schedule.weekly && this.selectedWeek && this.sendTime) {
        const days = Object.keys(this.days)
          .filter((day) => this.days[day])
          .map((day) => this.weekDays[Object.keys(this.days).indexOf(day)])
          .join(", ");
        return `Jede ${this.selectedWeek} Woche(n) am ${days}
(${this.sendTime} Uhr)`;
      } else if (
        this.schedule.monthly &&
        this.selectedOrdinal &&
        this.selectedDayOfWeek &&
        this.selectedMonthFrequency &&
        this.sendTime
      ) {
        return `Am ${this.selectedOrdinal} ${this.selectedDayOfWeek}, jeden
${this.selectedMonthFrequency}. Monat (${this.sendTime} Uhr)`;
      }
      return "Keine Vorschau verfügbar";
    },
    // Error message based on user selection and input
    errorOption() {
      if (this.selectedOption === "") {
        return "Bitte wähle eine der nachfolgenden Optionen.";
      }
      if (this.selectedOption == "1" && (!this.dateFieldValue ||
!this.timeFieldValue)) {
        return "Bitte lege einen Versandtermin in der Zukunft fest.";
      }
```

```
    if (this.selectedOption == "2" && this.patterns.length === 0) {
      return "Bitte lege ein Serienmuster fest.";
    }
    return ""; // No error
  }
},
methods: {
  updatePreview() {
    this.previewText;
  },
  updateValue(event) {
    this.terminValue = event.detail.value;
  },
  setSchedule(type) {
    this.schedule.weekly = type === "weekly";
    this.schedule.monthly = type === "monthly";
  },
  openSerienmuster() {
    this.$refs.serienmuster.opened = true;
  },
  closeSerienmuster() {
    this.$refs.serienmuster.opened = false;
    this.resetModal();
  },
  resetModal() {
    this.schedule.weekly = true;
    this.schedule.monthly = false;
    this.days = {
      monday: false,
      tuesday: false,
      wednesday: false,
      thursday: false,
      friday: false
    };
    this.selectedWeek = null;
    this.sendTime = "";
    this.selectedOrdinal = "";
    this.selectedDayOfWeek = "";
    this.selectedMonthFrequency = null;
  },
  acceptSerienmuster() {
    if (this.canSave) {
      const newPattern = `${this.previewText}`;
      this.patterns.push(newPattern);
      this.$refs.serienmuster.opened = false;
      this.resetModal();
    }
  },
  terminRadio(event) {
    this.previousOption = this.selectedOption;
```

```
      this.selectedOption = event.detail.value;
      this.confirmAction();
    },
    confirmAction() {
      const hasDataInFields = this.dateFieldValue && this.timeFieldValue;
      const hasPatterns = this.patterns.length > 0;
      if (
        (this.previousOption === "1" &&
          (this.selectedOption === "0" || this.selectedOption === "2") &&
          hasDataInFields) ||
        (this.previousOption === "2" &&
          (this.selectedOption === "0" || this.selectedOption === "1") &&
          hasPatterns)
      ) {
        this.openConfirmModal();
      } else {
        this.applyChanges();
      }
    },
    openConfirmModal() {
      this.$refs.confirmActionModal.opened = true;
    },
    closeConfirmModal() {
      this.$refs.confirmActionModal.opened = false;
    },
    applyChanges() {
      if (
        this.previousOption === "1" &&
        (this.selectedOption === "0" || this.selectedOption === "2")
      ) {
        this.dateFieldValue = "";
        this.timeFieldValue = "";
      }
      if (
        this.previousOption === "2" &&
        (this.selectedOption === "0" || this.selectedOption === "1")
      ) {
        this.patterns = [];
      }

      this.closeConfirmModal();
    },
    removePattern(index) {
      this.patterns.splice(index, 1);
    },
    getFirstWeekDay(pattern) {
      const weekDayOrder = ["Montag", "Dienstag", "Mittwoch", "Donnerstag",
"Freitag"];
      for (const day of weekDayOrder) {
        if (pattern.includes(day)) {
```

```
          return day;
        }
      }
      return null;
    }
  }
};
</script>

<style scoped>
.radio-group {
  display: flex;
  flex-direction: column;
  gap: 10px;
}

.flex-container {
  display: flex;
  align-items: center;
}

#text-field {
  width: 400px;
}

.inline-container {
  display: flex;
  align-items: center;
  gap: 10px;
  margin-bottom: 10px;
}

.scale-notification-banner {
  margin-top: 10px;
}

.patterns-container {
  margin-top: 10px;
  display: flex;
  flex-direction: column;
  gap: 10px;
}

.pattern-tag {
  display: flex;
  align-items: center;
  gap: 10px;
}
</style>
```