RpmCalendar

Hieronder een stap-voor-stap workflow voor de belangrijkste gebruikers- en systeeminteracties in de kalenderapplicatie:

1. **Applicatie laden**
   * Bij mount van RpmCalendar
     + Laad eerst categorieën (fetchCategories).
     + Laad vervolgens RPM-blokken (fetchRpmBlocks).
   * Stel defaultweergave in op “week” en huidige datum als startpunt.
2. **Kalenderdata ophalen**
   * Telkens wanneer currentDate of viewMode verandert:
     1. Bepaal startDate en endDate op basis van “day”/“week”/“month”.
     2. Roep /api/calendar-events?startDate=…&endDate=… aan.
     3. Converteer response naar interne CalendarEventDay[] en sla op in state.
3. **Categorieën filteren**
   * Toon in CategoryBar alle categorieën als ronde iconen met tooltip.
   * Klik op een categorie → zet activeCategory → herlaad of filter calendarEvents op die categorie.
4. **View-navigatie**
   * **Tabs**: gebruiker kiest “dag”/“week”/“maand” → setViewMode.
   * **Datumpijlen**: vooruit/terug één dag/week/maand → setCurrentDate(add…/sub…).
   * **Mini-Calendar** (alleen in maand- of weekweergave):
     + Toon mini-maand, klik op een dag → onDateSelect → setCurrentDate.
5. **Weergave-rendering**
   * **Maandview**: grid van 7×N cellen, elke cel is <CalendarDay>.
   * **Weekview**: per dag één kolom, met bovenaan dagheader + 24 <HourSlot>-cellen.
   * **Dagview**: lijst van 24 <HourSlot>-cellen voor de geselecteerde dag.
6. **MassiveAction-slepen**
   * **Bron**: EventItem in RPM-blokkenlijst is drag-source.
   * **Doel**: <CalendarDay> of <HourSlot> is drop-target (behalve voor verleden tijd of al “planned”).
   * Bij drop:
     1. Vul ontbrekende velden (uur, duur, status, etc.) aan.
     2. Roep onDrop(item, dateKey) of onDrop(itemWithHour) aan.
     3. In RpmCalendar werk je calendarEvents bij (POST/PUT naar API).
7. **Event-details bekijken/bewerken**
   * Klikken op een geplande actie in een <CalendarDay> of <HourSlot> → open CalendarPopup met die actie.
   * **In de popup**:
     + Pas titel, datum of datumbereik, uur, herhaling en status aan.
     + Bewerk rich-text beschrijving via de <Tiptap>-editor.
     + Voeg notities toe/bewerk/verwijder.
     + Toggle “Planned” of “Recurring” aan/uit.
     + Klik “Opslaan” → onUpdate(updatedAction, dateKey) → sync met API en update state.
     + Klik “Verwijderen” → onActionRemove(actionId, dateKey) → verwijder uit state en API.
8. **Event verwijderen**
   * In <CalendarDay> zit een kruis-icoon; klik → bevestigings-Dialog.
   * Bevestig → onActionRemove → verwijder actie uit calendarEvents en uit de backend.
9. **Visuele feedback**
   * **Huidige dag/uur**: ring of highlight.
   * **Verleden dag/uur**: gedimd (opacity-50) en drop uitgeschakeld.
   * **Drag-over**: achtergrondkleur verandert tijdelijk.
   * **Tooltips** op badges voor extra metadata (duur, status, gemist datum).
10. **Herhaling & datumbereik**
    * Bij recurring: selecteer weekdagen.
    * In fetchCalendarEvents en rendering wordt bij elk event met isDateRange en recurrDays gekeken of het in het huidige interval valt, en dan op de juiste dagen/uren getoond.

Met deze workflow zie je zowel de gebruikers­stappen (navigeren, slepen, bewerken) als de onderliggende datastromen en component­interacties.

**Functioneren van meerdaagse acties (multi-day events)**

1. **Inplannen in de pop-up**
   * In de CalendarPopup zet je de switch **“Meerdere dagen”** (isDateRange) aan.
   * Er verschijnen twee datum‐velden: **Begindatum** en **Einddatum**.
   * Kies de datum waarop de actie start en de datum waarop ze eindigt (bijv. 28 april t/m 30 april).
2. **Opslaan van de meerdaagse actie**
   * Bij klikken op **“Opslaan”** bundelt de component alle velden in een CalendarEvent-object, inclusief:

{

startDate: "2025-04-28",

endDate: "2025-04-30",

isDateRange: true,

… overige velden …

}

* + Dit object wordt via een **PUT**-request naar de API gestuurd (/api/calendar-events/:id).

1. **Server-side uitrollen per dag**
   * De backend ontvangt de range en genereert voor elke datum in het interval een dag-occurrence.
     + Bijvoorbeeld voor de periode 28–30 april maakt de server 3 records: één voor 2025-04-28, één voor 2025-04-29 en één voor 2025-04-30, allemaal gelinkt aan dezelfde actie-ID en met isDateRange = true.
2. **Data ophalen in de client**
   * Na opslaan (en bij elke view-verandering) roept RpmCalendar.fetchCalendarEvents de API aan met startDate/endDate van de huidige “dag”, “week” of “maand”.
   * De response is een array van CalendarEventDay-objecten:

[

{ date: "2025-04-28", events: [ … alle acties voor 28 april … ] },

{ date: "2025-04-29", events: [ … alle acties voor 29 april … ] },

{ date: "2025-04-30", events: [ … alle acties voor 30 april … ] },

… enzovoort …

]

1. **Omzetten naar interne state**
   * De client doet:

const convertedEventsByDay = data.map(dayObj => ({

date: dayObj.date, // bv. "2025-04-29"

events: dayObj.events, // lijst met alle acties voor die dag

// elk evt heeft nu ook evt.isDateRange === true

}));

setCalendarEvents(convertedEventsByDay);

1. **Maand- en weekweergave**
   * Bij elke cel <CalendarDay dateKey="YYYY-MM-DD"> wordt events gefilterd op event.date === dateKey.
   * Omdat de server voor elke dag in de range een entry terugstuurt, verschijnt de actie automatisch op elke dag van je gekozen interval.
2. **Dagweergave**
   * In renderDayCalendar wordt eerst gecheckt welke events voor de geselecteerde dag beschikbaar zijn:

eventsForDay = calendarEvents

.filter(dayObj => dayObj.date === dateKey

|| (action.isDateRange

&& dateKey tussen action.startDate én action.endDate))

* + Zo zie je in de dagweergave alle urenblokken waarin je meerdaagse actie loopt.

1. **Uurweergave**
   * Elke <HourSlot dateKey="…" hour={H}> krijgt de events voor die dag aangeleverd en toont alleen de acties waarvan action.hour === H.
   * Een meerdaagse actie met een vast uur verschijnt in dat uurblok op elke dag binnen de range.

**Kort samengevat**

* **UI:** via de switch “Meerdere dagen” kies je start- en einddatum.
* **API:** zet range om in losse dag‐occurrences met isDateRange = true.
* **Client:** toont de actie op iedere dag én in de juiste urenblokken zolang de datum binnen [startDate, endDate] valt.

Hier een overzicht van de belangrijkste test-scenario’s die je script doorloopt:

1. **Updaten van één specifieke dag binnen een meerdaags event (Test 9)**
   * Er wordt eerst een “Three Day Conference” aangemaakt met isDateRange=true van 1 mei t/m 3 mei.
   * Je controleert dat er drie afzonderlijke occurrences bestaan (één per dag).
   * Vervolgens roep je de endpoint PUT /api/calendar-events/occurrences/:occurrenceId aan voor de tweede dag (2 mei) en wijzig je alleen de tijd, duur en locatie van dát ene dag-object.
   * Na de update verifieer je:
     + Dag 1 en 3 zijn **ongewijzigd** (nog steeds 9:00, 480 min, “High Value”).
     + Dag 2 is **geüpdatet** (14:00, 120 min, “Very High Priority”, “Main Conference Hall”).
2. **Verplaatsen van een single-day event (Test B)**
   * Je maakt een één-daags event aan op 1 juni 2025 om 9:00 (isDateRange=false).
   * Je controleert dat er precies één occurrence op 2025-06-01 bestaat.
   * Via PUT /api/calendar-events/:actionId wijzig je startDate naar 15 juni 2025 en de tijd naar 14:00, plus andere eigenschappen.
   * Daarna verifieer je:
     + De **oude** occurrence op 1 juni is **verwijderd**.
     + Er is exact **één nieuwe** occurrence op 15 juni, met alle aangepaste velden (tijd, duur, etc.).
3. **Luisteren en verwijderen binnen een date-range event**
   * **Aanmaken** van een multi-day event (bv. 1 – 3 mei).
   * **Updaten** van één dag (Test 9, hierboven).
   * **Verwijderen** van één dag uit de range (Test 10): via DELETE /api/calendar-events/:occurrenceId/:date verwijder je bv. de occurrence voor 1 mei.
     + Je controleert dat de andere dagen (2 en 3 mei) **gewoon blijven bestaan**.
     + En dat de hoofd-event zelf **niet wordt gereset** (startDate / endDate blijven ongewijzigd zolang er nog dagen over zijn).
   * Tenslotte kun je via DELETE /api/calendar-events/:actionId de hele date-range event inklusief alle resterende occurrences opruimen.

Met deze scenario’s dek je alle cruciale use-cases af:

* Updaten van **één enkele occurrence** binnen een reeks.
* **Verplaatsen** van een niet-terugkerend event.
* **Creëren**, **aanpassen** en **(gedeeltelijk) verwijderen** binnen een meerdaags event.

**Verplaatsen van een enkel-dags RPM Massive Action (Test B)**

* Je maakt een RPM Massive Action aan met één occurrence op **1 juni 2025** om **09:00** (isDateRange=false).
* Je controleert dat er precies **één** record in rpm\_massive\_action\_occurrence bestaat voor die actionId op **2025-06-01**.
* Je verstuurt een PUT /api/calendar-events/:actionId waarin je alleen de startDate, hour (en eventuele andere velden) bijwerkt naar **15 juni 2025 14:00**.
* Vervolgens verifieer je dat:
  1. De **oude** occurrence (voor 1 juni 2025) echt **is verwijderd** uit rpm\_massive\_action\_occurrence.
  2. Er nu **exact één** nieuwe occurrence bestaat op **15 juni 2025** met de bijgewerkte tijd en eigenschappen.