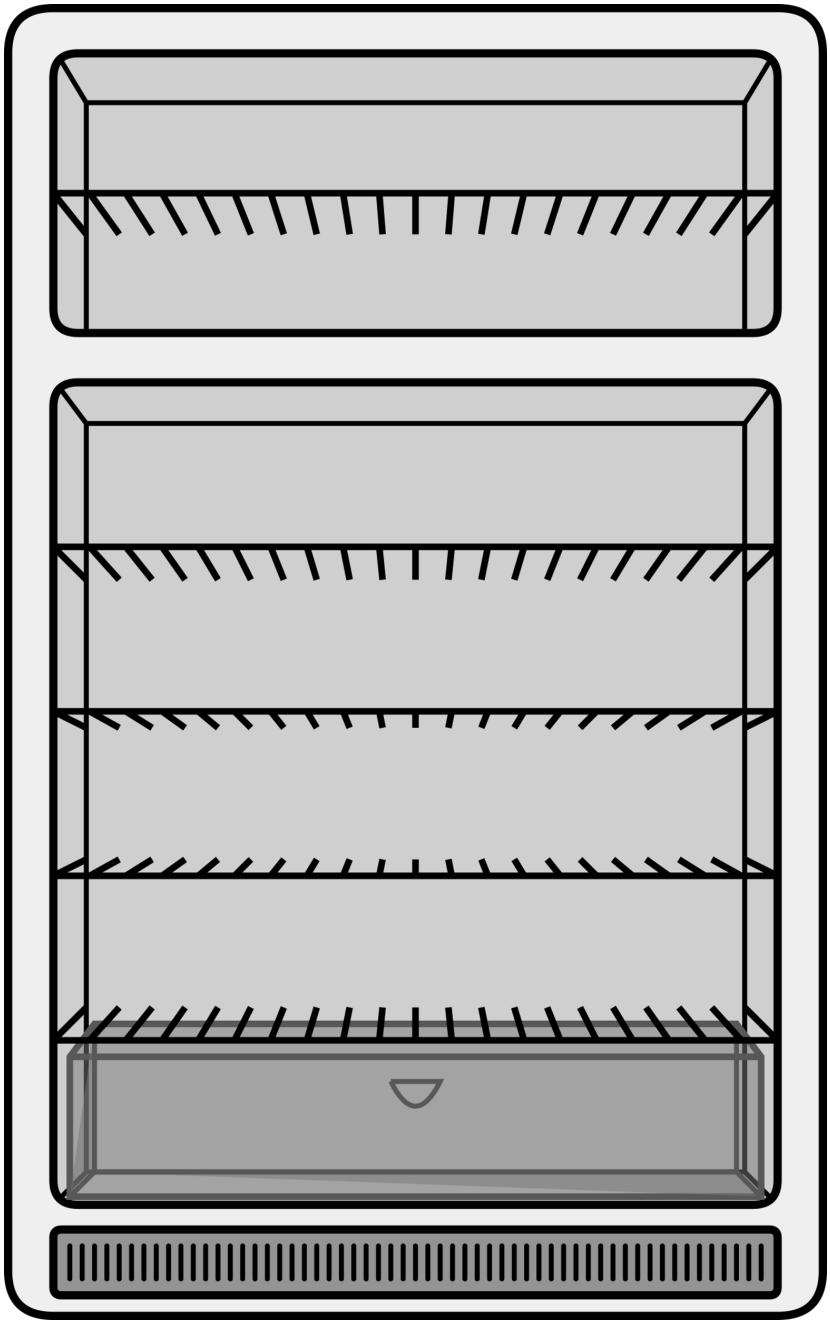
Kühlschrankverwaltung

Modularbeit für Modul 254 – Geschäftsprozesse im eigenen Berufsumfeld beschreiben



von

Joel Meccariello

Jonas Tochtermann

eingereicht am 09. November 2021

Inhaltsverzeichnis

Zielsetzung

Diese Applikation soll der einfachen Kontrolle eines Kühlschrankinhalts dienen: Man kann neu gekaufte Lebensmittel hinzufügen, aufgebrauchte entfernen, Änderungen vornehmen sowie den Bestand auslesen. Darüber hinaus wird an jedem Samstagmorgen ein Mail mit der aktuellen Bestandsliste verschickt, so dass man weiss, was man einkaufen muss. Ebenfalls wird einmal am Tag überprüft, ob ein Lebensmittel bald das Ablaufdatum erreicht; dann wird ebenfalls ein Mail verschickt.

Prozessbeschreibung

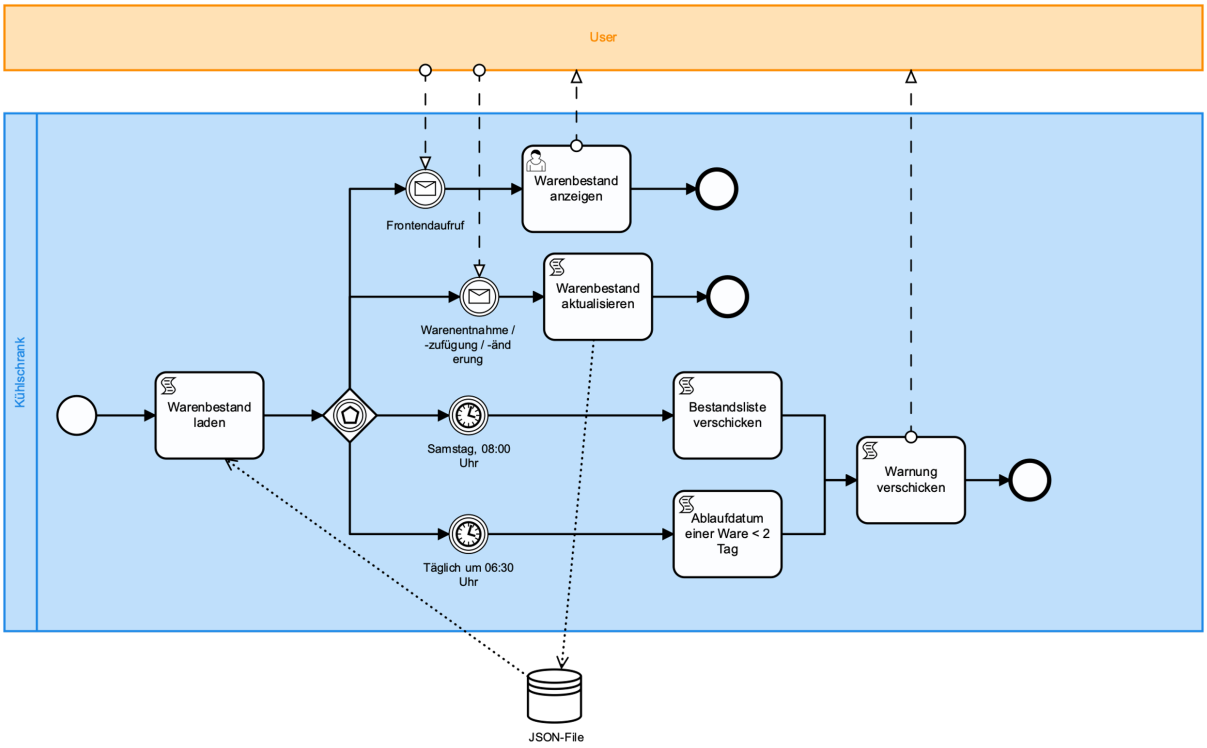
Nach dem Starten des Prozesses werden die Daten (Liste mit Lebensmitteln) aus einer Datei gelesen; anschliessend wird gewartet, bis entweder eine Nachricht eintrifft oder ein Timer aktiv wird.

Nachrichten gibt es zweierlei: Einmal eine einfache für das Abfragen aller Daten. Worauf ein UserTask für den Abschluss des Prozesses folgt. Dazwischen (im Modell nicht sichtbar) findet ein API-Call für die Abfrage der Prozessvariablen statt. Der UserTask folgt dann, damit der Prozess nicht vorzeitig beendet wird. Nach dem API-Call wird der UserTask ausgeführt und der Prozess beendet.

Die Andere Nachricht betrifft eine Änderung in den Daten; dabei wird innerhalb der Nachricht noch die Art (Hinzufügen, Löschen, Modifizieren) und der Inhalt der Änderung mitgegeben. Die Daten werden durch ein Skript bearbeitet, danach wird der Prozess beendet.

Auch Timer gibt es zweierlei: Jeden Samstag um 8:00 Uhr wird die Bestandsliste per Mail verschickt, und jeden Tag findet eine Prüfung der Ablaufdaten statt; hier wird die Liste derjenigen Lebensmittel verschickt, die in den nächsten zwei Tagen ablaufen.

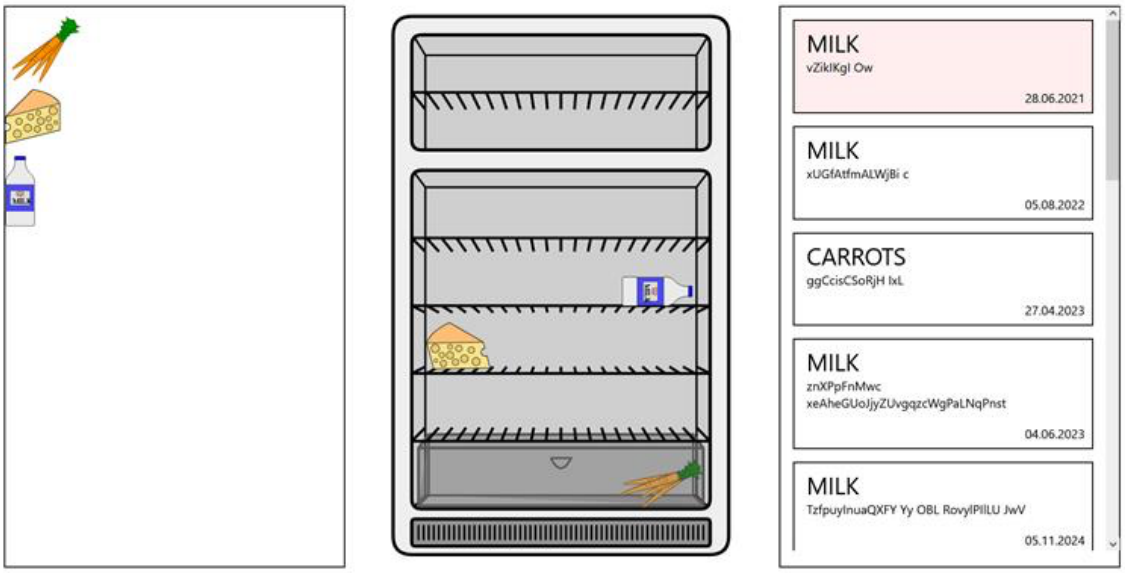
Graphische Darstellung



Architektur

Unsere Applikation besteht aus 3 Teilen: Frontend (React), Gateway (Dart) und Backend (Camunda, wie oben dargestellt).

Frontend



- Links ist der Store zu sehen mit den verfügbaren, neu hinzuzufügenden Lebensmitteln.
- In der Mitte ist der Kühlschrank mit den vorhandenen Lebensmitteln.
- Rechts ist eine detaillierte Auflistung der im Kühlschrank vorhandenen Lebensmitteln zu sehen.

Das Frontend ist als React-App geschrieben und unterstützt Drag-&-Drop

GateWay-Implementierung

Das Gateway nimmt allgemein gehaltene REST-API-Calls vom Frontend entgegen und gibt sie (verarbeitet) an die Camunda-Platform weiter. Das heisst, sie startet den Prozess und füttert ihn mit Nachrichten bzw. lässt die Timer arbeiten.

Geschrieben ist das GateWay als Http-Server in Dart.

Beschreibung Vorgehen

Am ersten Tag der Modularbeit überlegten wir uns, was unser Kühlschrankprogramm alles können sollte. Wir entschieden uns neben grundlegenden Aktionen wie Bestand auslesen, Hinzufügen, Wegnehmen und Ändern auch für zwei zusätzliche Möglichkeiten: Eine wöchentliche Einkaufsliste und eine tägliche Kontrolle auf bald ablaufende Lebensmittel.

Für die Umsetzung entschieden wir uns (neben der obligatorischen Schicht BPMN) für ein React-Frontend und ein Dart-Gateway; das Frontend würde Joel schreiben, das Gateway Jonas. Die Erarbeitung des BPMN-Prozesses sollte von beiden zusammen erfolgen.

Wir erstellten zusammen die erste Version des Prozesses und definierten die API-Endpoints für das Gateway

In der Folgezeit hatten wir einige Kommunikationsschwierigkeiten, an welchen Tagen wir in der Schule und wann im Distance-Learning sind. Einerseits verloren wir dadurch Zeit, andererseits bemerkten wir zu spät, dass einige Überlegungen nicht für Camunda brauchbar waren: Da wir nicht wussten, wie Camunda-Platform mit einer Datenbank verbinden werden kann, wurde das GateWay zunächst um den Prozess herumgebaut (anstelle einer Verbindung nur zwischen Frontend und Camunda auch zwischen Camunda und Datenbank). Nach einem Gespräch mit dem Lehrer wurde die Idee einer Datenbank ersetzt durch die Speicherung in einer Datei, was mit Camunda etwas einfacher durchzuführen sei.

Das nächste Problem stellte die Einbindung von Skripts in Camunda dar: Da in den meisten Tutorials groovy als Skriptsprache vorherrschend war, haben auch wir uns an diese Sprache gewagt. Es dauerte danach eine sehr lange Zeit, bis wir herausfanden, dass dafür eine zusätzliche Jar-Datei (die Engine) nötig ist. Und nochmals eine Zeit dauerte es, bis wir die richtige fanden. In einigen Distributionen von Camunda ist diese Datei dabei, in der unsrigen leider nicht. Mit viel Google kamen wir schliesslich zum Ziel.

In der Zwischenzeit wurde auch sichtbar, dass die Erstellung des Frontends viele kleine Details erforderte, was auch die zur Verfügung stehende Zeit verkürzt hat.

Nachdem groovy einmal gebraucht werden konnte, musste diese Sprache erst einmal etwas angelernt werden. Mit Hilfe einiger kleiner BPMN-Prozesse wurde die Funktionalität getestet und im Projekt eingefügt.

Die letzten Schwierigkeiten, die leider auch bis Abgabe nicht vollständig gelöst werden konnten, zeigten sich beim Zusammenfügen der Einzelteile: Vor Allem zwischen Gateway und Backend haperte noch einiges; durch die Fehlermeldungen wurde uns auch bewusst, dass wir infolge mangelhafter Kenntnis von Camunda und BPMN einen zwar theoretisch funktionierenden, aber unschönen und schwierig handhabbaren Prozess definiert haben. So sollten keine repetitiven Timer auf ein eventbasiertes Gateway folgen, und anstelle von neuen Starts bei jedem Aufruf wäre ein sich wiederholender Prozess sinnvoller gewesen.

Testfälle

Nr.	Bezeichnung	Eingabe	Erwartete Ausgabe	Tatsächliche Ausgabe
1	Bestand anzeigen	Start des Backends, danach Start des Frontends	Graphische Darstellung gemäss Inhalt der Datei (eine Milch)	Eine Milch
2	Bestand hinzufügen	Drag&Drop eines neuen Lebensmittels in den Kühlschrank	Änderung der Datei	Fehler im Backend
3	Bestand ändern	Bearbeiten eines Elements im Kühlschrank	Änderung der Datei	Fehler im Backend
4	Bestand löschen	Löschen eines Elements in der Bestandsliste	Änderung der Datei	Die Datei wurde wie erwartet geändert
5	Ablaufdatum überprüfen	Backend vor 06:25 einschalten	Mail wurde verschickt	Noch nicht getestet
6	Bestandsliste versenden	Backend vor Samstag, 07:55 einschalten	Mail wurde verschickt	Noch nicht getestet

Voraussetzungen / Installation

Voraussetzungen

Folgendes muss installiert sein:

- Camunda Platform (Community Edition)
- Camunda Modeler (für das Deployment)
- npm

Backend (Camunda)

Im Ordner configuration/userlib müssen die beiden Dateien send_mail.groovy und groovy-all-2.4.13.jar hinzugefügt werden. In der Datei send-mail.groovy müssen Email-Adresse, Password sowie Servereinstellungen angepasst werden. Die Platform wird durch das Ausführen des Skripts start.sh (für Mac/Linux) bzw. start.bat (für Windows) gestartet, danach durch Ctrl-c beendet.

Der Prozess (fridge-process.bpmn) muss noch deployed werden.

Gateway

Im Ordner m254-gateway/build sind die ausführbaren Dateien m254-gateway-wd (für Windows), m254-gateway-mc (für macOS) bzw. m254-gateway-lx (für Linux); beendet werden sie durch Eingeben von "exit" und Betätigen der Enter-Taste.

Frontend

Um das Frontend zu starten, braucht es lediglich drei Befehle:

- in den Ordner gehen: -> cd m254-lb02-frontend
- Abhängigkeiten installieren. -> npm i
- -> npm start

Alternativ kann man auch einen Build erstellen:

- -> npm i serve
- -> npm build
- -> serve -s build