Vuetify – Frontend zu WDH REST Server

Robert Baumgartner

Du hast zur Wiederholung des Stoffes der vierten Klasse einen REST Server für ein Autohaus erstellt (WDH REST Server). Nun wollen wir ein Frontend in Vuetify zu diesem Server entwickeln. Dabei kannst du auch auf Teile deiner Lösung der Aufgabe WDH Vuetify zurückgreifen.

Neu bei dieser Aufgabe ist die Übergabe von Daten an eine Route. Es wird eine mögliche Variante vorgestellt. Weitere werden folgen.

Voraberklärung
Du kannst an den Routerview genauso wie an jede andere Komponente Props übergeben. Im Routerview wird ja eine Komponente (und zwar die via Router ausgewählte) angezeigt. Erinnere dich, dass Views auch nur Komponenten sind.

```
<router-view :cars="cars"></router-view>
```

Die Daten werden dann an die dargestellte View (Komponente) übergeben. Verwendet die View die Props nicht, dann ist das auch OK.

Weiters können Routen (wie am Server) auch dynamisch sein. Wir können also zum Beispiel eine Route mit Pfad

```
'/details/:id',
```

im Router definieren. Am Server hätten wir mit `req.params.id` darauf zugegriffen, doch wie geht das am Client? Dazu setzen wir das Property `props` auf `true`. Danach können wir uns in der View über ein zusätzliches Prop freuen. In diesem Fall `id`.

Beispiel bei **router/index.js:**

```
{
  path: '/details/:id',
  name: 'Details',
  props: true,
  component: () => import(/* webpackChunkName: "details" */ '../views/Details.vue'),
},
```

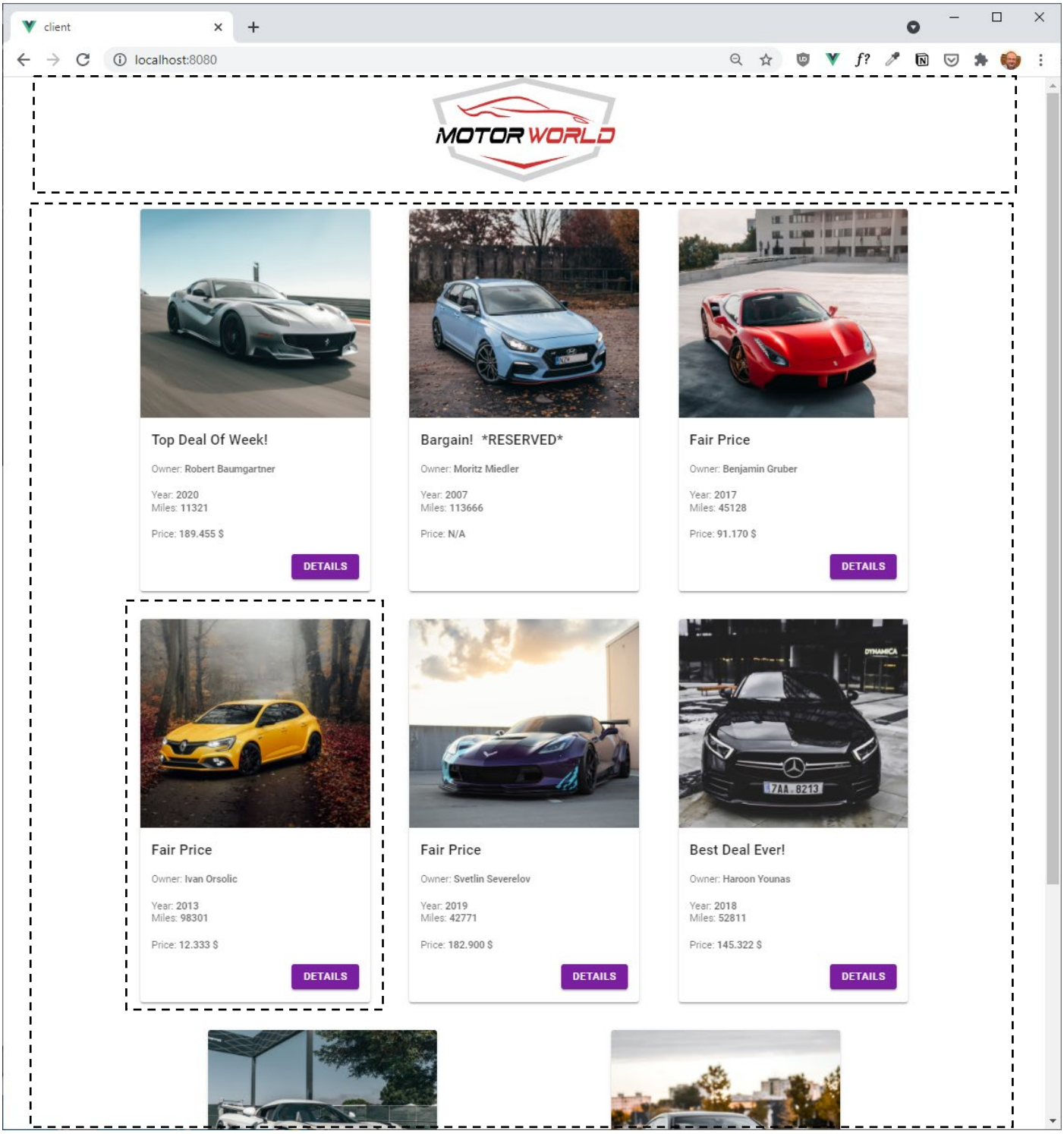
In **Details.vue:**

```
props: {
  id: {
    type: String,
  },
},
```

Aufgabenstellung
Der Flow sollte folgendermaßen sein: Zunächst wird die Hauptseite mit allen Autos angezeigt (Route: **/** bzw. **/home**). *Autos, die nicht mehr verfügbar sind, werden zwar angezeigt, aber nicht der Preis und der* **Details** *Button.*

Klickt der User auf den **Details** Button, wird zur Route `/details/:id` verzweigt und weitere Informationen zu dem Auto angezeigt. In dieser Ansicht kann der User das Auto kaufen (Status wird auf **reserviert** gesetzt, der Server wird mittels PATCH davon informiert). Das GUI wird geändert (siehe Screenshots weiter unten).

Route: **/** bzw. **/home**



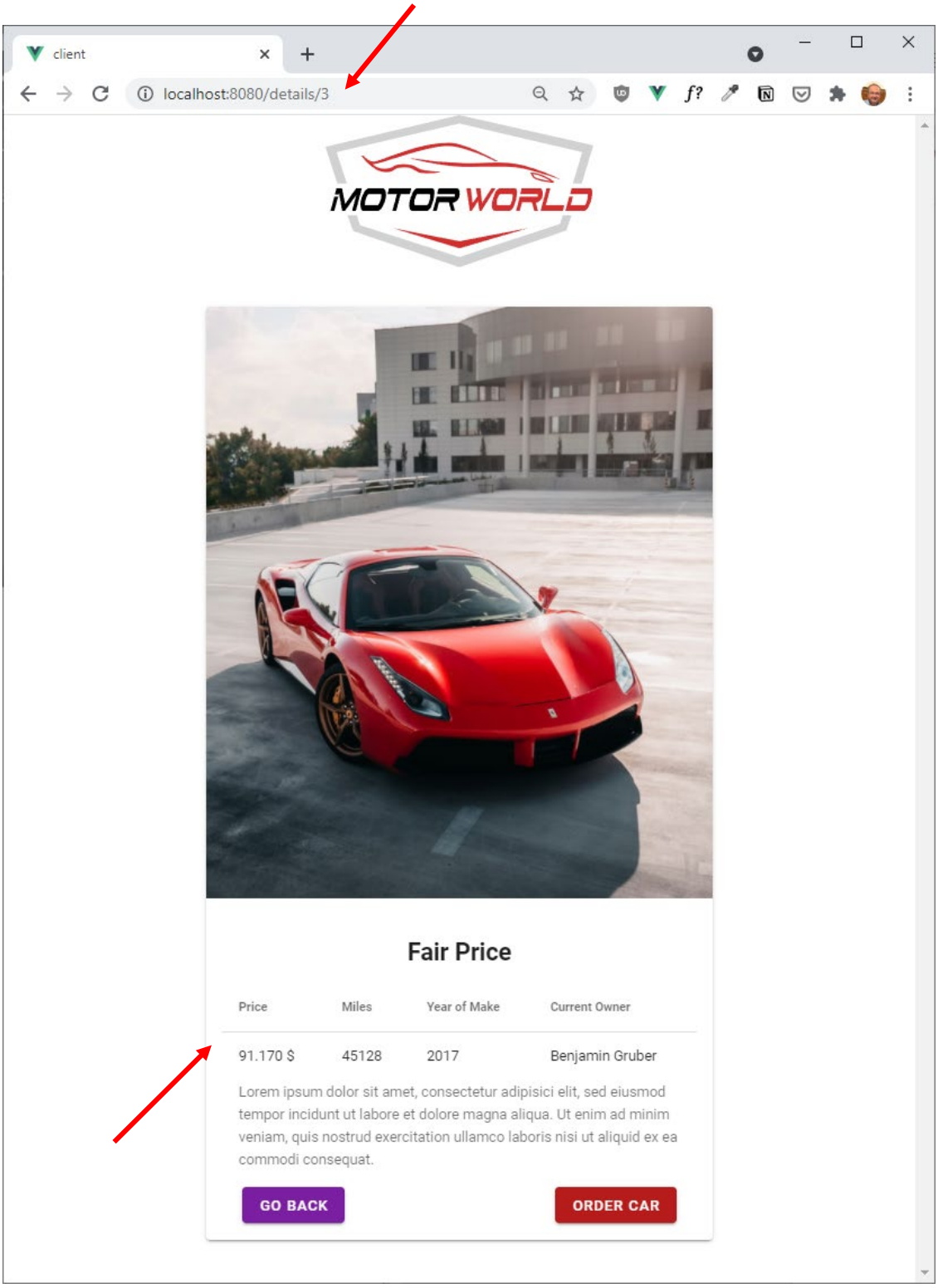
Das Auto in der Mitte der oberen Reihe wurde gekauft (Status: **reserviert**). Die Zusatzinformation wird angezeigt und der Button entfernt. Preis ist N/A

App.vue holt die Daten vom Server.

Komponenten: **LogoBar.vue (v-app-bar** mit Attribut **app)**, **CarCards.vue**, **CarCard.vue**.

Verwende in **CarCard.vue** : **v-card.**

Route: **/details/3**

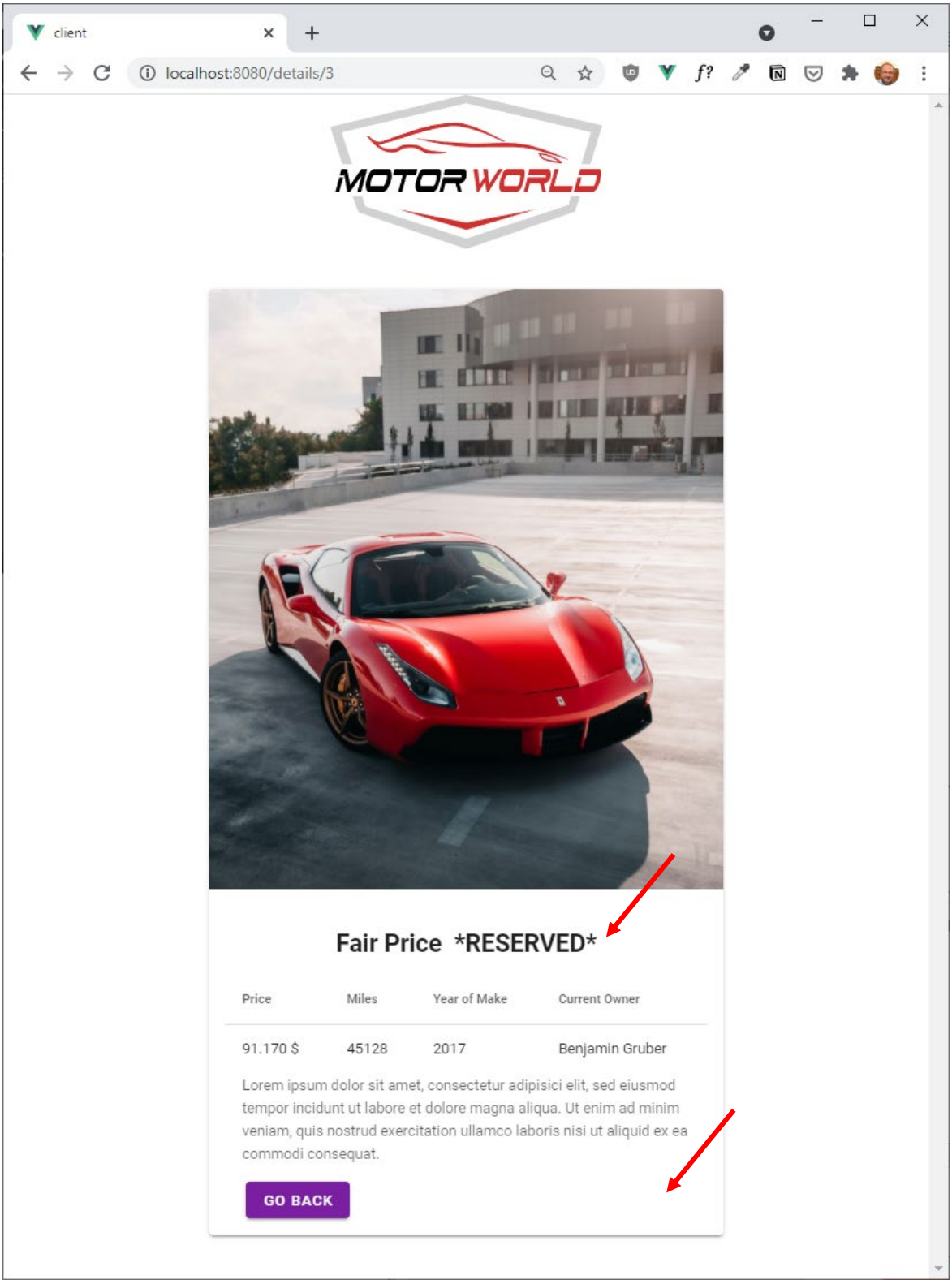


Der Einfachheit halber, verwende hier keine eigene Komponente.

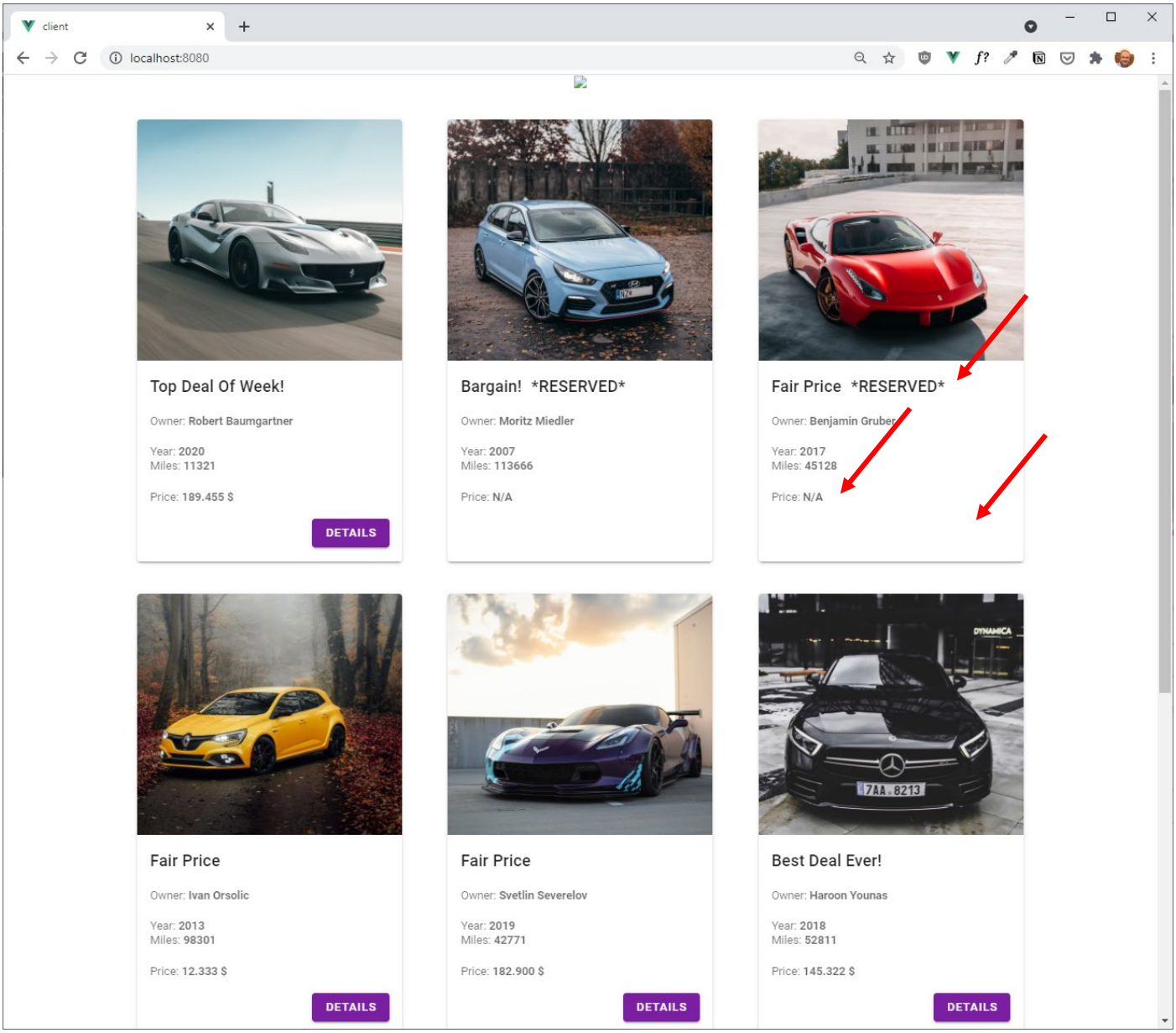
Details.vue macht den PATCH Call.

Verwende: `v-simple-table` für die Tabelle (eine Zeile).

Der Button **Go Back** führt zu **/home**, der Button **Order Car** führt einen Patch Call zum Server aus und ändert das GUI (siehe unten).



Drückt der User nun auf **Go Back**, so sieht er die geänderte Übersicht.


client
localhost:8080
Top Deal Of Week!
Owner: Robert Baumgartner
Year: 2020
Miles: 11321
Price: 189.455 $
DETAILS
Bargain! *RESERVED*
Owner: Moritz Miedler
Year: 2007
Miles: 113666
Price: N/A
Fair Price *RESERVED*
Owner: Benjamin Gruber
Year: 2017
Miles: 45128
Price: N/A
Fair Price
Owner: Ivan Orsolic
Year: 2013
Miles: 98301
Price: 12.333 $
DETAILS
Fair Price
Owner: Svetlin Severelov
Year: 2019
Miles: 42771
Price: 182.900 $
DETAILS
Best Deal Ever!
Owner: Haroon Younas
Year: 2018
Miles: 52811
Price: 145.322 $
DETAILS