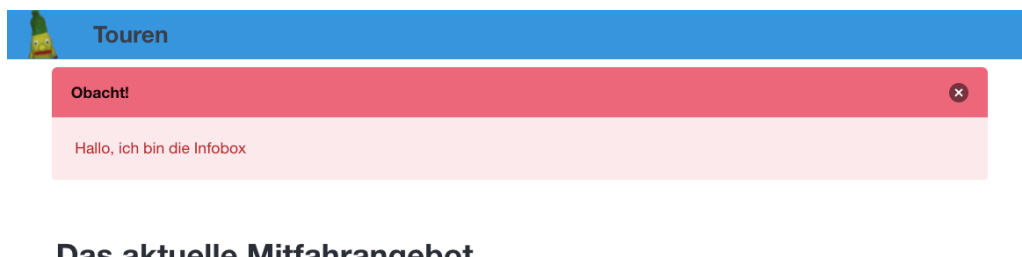


Webbasierte Anwendungen

Web-Ueb-09 (19.06.2024)

Aufgabe 1 (Composable: useInfo.ts)

Im bisherigen Projektstand haben wir in der Rahmenkomponente `App.vue` eine Fehlerbox installiert, welche sich zeigt, wenn in die Ref `info` ein nichtleerer String gesteckt wird.



Wir wollen nun von überall aus in unserem Frontend diese `info`-Nachricht setzen können, also auch von anderen Komponenten aus als `App.vue`. Die Idee ist daher, die Nachricht in ein überall einbindbares Composable, welches das reaktive `info` verwaltet, auszulagern.

Legen Sie bitte in `src` einen Ordner `composables` mit einem TypeScript-Modul `useInfo.ts`. Dieses Modul definiert auf Modulebene die bekannte String-Ref `info` und exportiert eine Composition Function `useInfo()`, welche ein Objekt mit drei Elementen zurückgibt:

- `info` als reaktives read-only des vom Composable verwalteten “State-Objekts” `info`
- eine Funktion `loescheInfo()`, welche `info` auf den Leerstring setzt und
- `setzeInfo(msg: string)`, welche `info` mit der übergebenen Nachricht belegt.

Entfernen Sie nun bitte die im dortigen `script`-Teil definierte alte `info`-Ref aus `App.vue` und verwenden Sie stattdessen Ihr Composable `useInfo()`, um an dessen `info` sowie die `loescheInfo()`-Funktion zu kommen, und zeigen Sie nun bitte dieses `info` in Ihrer Infobox an.

Verwenden Sie die `loescheInfo()`-Funktion in Ihrer Implementierung der Info-Box in `App.vue`, damit man diese über einen Button schließen kann, wenn man die Nachricht nicht mehr sehen möchte.

Hinweis: Das heißt *nicht*, dass Sie jedes Mal die Composition Function aufrufen, wenn Sie auf ein Element von deren Rückgabeobjekt zugreifen wollen (also *nicht* `useInfo().info` oder `useInfo().loescheInfo()`). Funktioniert zwar, ist aber garstig.

Rufen Sie `useInfo()` in `App.vue` *einmal* auf und nehmen Sie sich dabei per *Destructuring* die gewünschte(n) Teile, hier `info` und die `loescheInfo()`-Funktion.

Aufgabe 2 (Ein Pinia-Store für die Touren)

Analog möchten wir nun die aufzulistenden Touren nicht mehr in `TourenListView.vue` hinterlegen, sondern in einen Pinia-Store auslagern.

In einem ersten Schritt legen Sie im vorhandenen `stores`-Ordner bitte eine TypeScript-Datei `ITourDTD.ts` an, welche das bisher in `TourenListView.vue` definierte Interface `ITourDTD` exportiert.

Fügen Sie bitte im gleichen `stores`-Verzeichnis ein TypeScript-Modul `tourenstore.ts` an, das einen Pinia-Store namens `tourenstore` definiert und als `useTourenStore` nach außen bereitstellt (`export`). Der Store soll folgendes beinhalten und seinen begeisterten Nutzern über `useTourenStore()` anbieten:

- `tourdata` als ein reaktives Objekt mit einer bool'schen Property `ok` und der bekannten `tourliste` als `Array<ITourDTD>`. Dieses Objekt repräsentiert den vom Store verwalteten State.
- `updateTourListe()` als eine Funktion, welche die `tourliste`-Komponente unserer `tourdata` mit den Testdaten belegt, die bisher in `TourenListView.vue` hardcoded sind.

Bitte Sie aus `TourenListView.vue` nun die Interface-Definition und die Definition von `tourliste`, und bauen Sie diese stattdessen in Ihren Store ein, um dieselbe Funktionalität zu erhalten.

Beachten Sie bitte, dass die Tourliste im Store erst gefüllt wird, wenn jemand `updateTourListe()` aufruft. Seien Sie bitte dieser jemand und überlegen Sie sich, wie Sie dafür sorgen können, dass beim Laden der `TourenListView` anfangs einmal `updateTourListe()` aufgerufen wird. Sie sollten danach die im Store hinterlegte Touren-Liste in Ihrer Webansicht sehen (sollte wie zuvor aussehen).

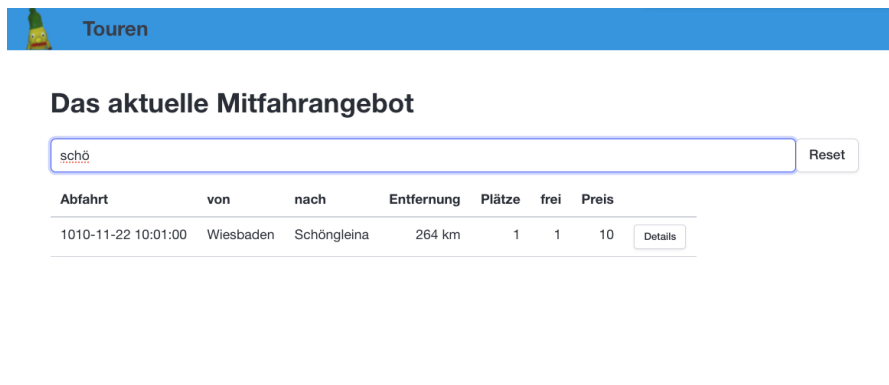
Nutzen Sie die Developer Tools in Ihrem Browser, um sich danach unter dem "Vue"-Tab den `tourenstore` anzusehen. Sie können den Inhalt live ändern (und sollten den Effekt gleich in Ihrer Webansicht sehen), könnten ihn als JSON exportieren oder importieren usw.

Aufgabe 3 (Inkrementelle Suche in Touren-Liste)

Bitte ergänzen Sie die `TourenListView` um ein `INPUT`-Feld, um eine inkrementelle Suche in der Tourenliste zu ermöglichen. Wenn das Suchfeld leer ist, soll wie bisher die ganze Liste der Touren-Angebote aus unserem `tourenstore` angezeigt werden.

Wird ein Suchstring eingetippt, so passt sich die ausgegebene Liste unverzüglich an, um nur noch Touren zu zeigen, in deren Start- oder Zielort-Namen der Suchstring vorkommt, wobei Groß-/Kleinschreibung keine Rolle spielt ("wie" würde also "**W**iesbaden" matchen).

Damit die oder der Usende bequem zur Vollansicht der List zurückkehren kann, fügen Sie bitte neben dem Suchfeld einen "Reset"-Button hinzu, der das Suchfeld auf den Leerstring zurücksetzt.



Abfahrt	von	nach	Entfernung	Plätze	frei	Preis	
1010-11-22 10:01:00	Wiesbaden	Schöngleina	264 km	1	1	10	<button>Details</button>

Denken Sie daran, dass das Filtern der anzuzeigenden Touren hier eine reine Frage der *Darstellung* ist (und daher z.B. in `TourenListView` plausibel verankert ist). Es soll dabei daher **nicht** die im `tourenstore` gehaltene Ursprungsliste verändert werden.

Aufgabe 4 (Vue-Router)

Nun möchten wir den “Details”-Button in unserer Tourenliste nutzen, um zur ausgewählten Tour eine Detailansicht mit mehr Informationen zu zeigen.

App.vue: RouterView nutzen

Momentan haben wir in der Rahmen-Komponente `App.vue` nur (hardcoded) unsere `TourenListView` eingebettet. Diese möchten wir nun auf Wunsch gegen eine (neu anzulegende) `TourView.vue` austauschen, und zwar mit dem Vue-Router.

Bitte ersetzen Sie in `App.vue` die `TourenListView` durch die `RouterView`-Komponente des Vue-Routers. Da wir immer noch zum Einstieg unsere `TourenListView` sehen möchten, konfigurieren Sie die Router-Konfigurationsdatei `router/index.ts` bitte so, dass unter dem Pfad `/touren` die `TourenListView` gezeigt wird. Zusätzlich tragen Sie bitte in die Router-Konfiguration ein, dass der Wurzelpfad `/` auf `/touren` umgeleitet werden soll (wenn man `http://localhost:5173` abrufen soll, soll man also automatisch auf `.../touren` kommen).

Bitte testen Sie browsend, ob Sie beim Einstieg über `http://localhost:5173` dorthin gelangen und Ihre vertraute Touren-Tabelle gezeigt bekommen.

TourView.vue anlegen

Legen Sie bitte unter `views` eine neue Ansicht `TourView.vue` an, welcher eine Property `tourid` übergeben werden kann. Die Ansicht sollte die Tour-Informationen übersichtlich anzeigen, einschließlich eines eventuell enthaltenen Info-Textes (im Beispiel direkt unter der Überschrift).

Um an die anzuzeigenden Daten zu kommen, greift die `TourView` auf die `tourdata` im eben angelegten `tourenstore` zu und fischt sich das Tour-Objekt zu der übergebenen `tourid` aus der `tourliste` heraus (Hinweis: die `tourid` ist ein `string`, also ggf. bei der weiteren Verarbeitung mit `parseInt()` in eine `number` konvertieren).



Tour 2: Wiesbaden - Clenze

Alle wollen nach Clenze - Du doch sicherlich auch! Oder nicht? Doch!

Abfahrt am **2033-02-01 um 18:00:00 Uhr**,

Preis **17 EUR** für 370 km.

Anbieter ist **Joghurta Biffel**

es sind von 3 Plätzen bisher 0 gebucht (**3 freie Plätze**)

Sie können Ihre `TourView` leicht testen, indem Sie sie in `App.vue` vorübergehend direkt einbetten, also z.B. `<TourView tourid="1"></TourView>` um die Tour Nr. 1 anzuzeigen. Entfernen Sie die Komponente danach bitte wieder aus `App.vue`

Router-Konfiguration und Detail-Link in Touren-Liste

Nachdem wir uns überzeugt haben, dass unsere `TourView` funktioniert, möchten wir erreichen, dass sie aus der `TourenListView` direkt angesteuert werden kann.

- Ergänzen Sie die Vue-Router-Konfiguration bitte so, dass unter dem Pfad `/tour/:tourid` die Komponente `TourView` ausgewählt wird. Der Pfadparameter `tourid` soll vom Router als Prop an die Komponente durchgereicht werden (aus Sicht der Komponente wird sie also mit dem Tour-ID-Parameter “versorgt” wie oben von Hand).

- Ersetzen Sie bitte den “Details”-Link in Ihrer `TourenListeZeile.vue` durch einen `RouterLink`, so dass der `/tour/tourid`-Pfad mit der jeweiligen ID angesteuert wird. Danach sollten Sie per Klick auf den “Details”-Link von der Touren-Liste automatisch auf die ausgewählte Tour-View kommen.
- Ergänzen Sie bitte in der Kopfzeile Ihrer `App.vue` ebenfalls einen `RouterLink` namens “Touren”, der die Ansicht auf `/touren` (die Touren-Liste) setzt.

Sie sollten nun zwischen der Übersichtsliste und den jeweiligen Einzelansichten verzögerungslos hin- und herschalten können.

Touren

Das aktuelle Mitfahrangebot

Abfahrt	von	nach	Entfernung	Plätze	frei	Preis	
1010-11-22 10:01:00	Wiesbaden	Schöngleina	264 km	1	1	10	Details
2023-11-17 19:30:00	Wiesbaden	Taucha	333 km	3	3	5	Details
2024-05-10 16:27:00	Wiesbaden	Bochum	173 km	2	2	12	Details
2033-02-01 18:00:00	Wiesbaden	Clenze	370 km	3	3	17	Details

Touren

Tour 2: Wiesbaden - Clenze

Alle wollen nach Clenze - Du doch sicherlich auch! Oder nicht? Doch!

Abfahrt am **2033-02-01 um 18:00:00 Uhr**,
Preis **17 EUR** für 370 km.
Anbieter ist **Joghurta Biffel**
es sind von 3 Plätzen bisher 0 gebucht (**3 freie Plätze**)

Aufgabe 5 (useInfo in TourView einbauen)

Wir hatten zu Beginn das `useInfo`-Composable zur einfachen Nutzung unserer Info-Box in `App.vue` von “überall” gebaut.

Bitte integrieren Sie `useInfo` in `TourView`, um die Nutzer zu warnen, wenn die ausgewählte Tour mehr als 300 km lang ist. Das `ITourDTD` enthält bereits die `distanz`, siehe Interface-Definition. Die Nachricht enthält mindestens den Namen des Start- und des Zielorts dieser langen Tour.

Touren

Obacht!

Die Tour von Wiesbaden nach Clenze ist weiter als 300 km, bitte Kekse einpacken.

Tour 2: Wiesbaden - Clenze

Alle wollen nach Clenze - Du doch sicherlich auch! Oder nicht? Doch!

Abfahrt am **2033-02-01 um 18:00:00 Uhr**,
Preis **17 EUR** für 370 km.
Anbieter ist **Joghurta Biffel**
es sind von 3 Plätzen bisher 0 gebucht (**3 freie Plätze**)

Akzeptanzkriterien für diesen Projektstand

- Tourenansicht (Startseite) zeigt die im Pinia-Store `tourenstore` hinterlegten Touren an.
- Sobald im Suchfeld der Touren-Liste mindestens ein Zeichen eingegeben wird, wird die angezeigte Liste auf die Zeilen eingeschränkt, die den Suchstring im Start- oder Zielortnamen enthalten (Groß-/Kleinschreibung egal). Die Anpassung der Liste erfolgt unverzüglich bei jeder Veränderung des Suchfeldinhalts.

- Mit einem Reset-Button wird der Inhalt des Suchfelds wieder auf den Leerstring zurück gesetzt und man sieht wieder die ungefilterte Gesamtliste.
- Per “Details”-Link wird für jede Tour der Liste per Vue-Router auf eine Detailansicht-Seite gewechselt.
- Bei Touren über 300 km Länge erscheint bei Anzeige der Detailansicht die in `App.vue` integrierte Infobox mit einer Warnnachricht, die den Namen des Start- und des Zielorts enthält, umgesetzt über das `useInfo-Composable`.
- Die Infobox lässt sich über einen Button in der Box schließen.