LABO 2: Vue, VueRouter, Vuex, VueResource

Doelstellingen

Dit labo bevat een allesomvattend opdracht van alle Vue en Vue gerelateerde aspecten die je over de afgelopen weken hebt leren kennen. Het hoofddoel van dit labo is zelfstandig leren op zoek gaan naar oplossingen, en deze succesvol integreren in je applicatie. Dit neemt niet weg dat de 10 minuten regel nog steeds geldt: als je meer dan 10 minuten vast zit, moet je hulp vragen. De docent zal je dan helpen met de juiste pointers om verder te kunnen.

Werkwijze, timing en upload

Dit labo loopt over een sessie en moet voor 7 April geüpload worden via Toledo. Upload één enkele zip met daarin je oplossingen.

Opgaves

1. Oefening Stock Manager

Meegegeven is een Vue project met daarin een component namelijk Header.vue. Het doel van de opdracht is om een Single Page Application te maken met meerdere pagina's om te fictief te handelen in aandelen. De opdracht moet een Home, Stocks en Portfolio pagina bevatten (zie fotos onderaan)

- De homepagina bevat wat uitleg over het gebruik van de applicatie.
- De stocks pagina bevat alle mogelijke aandelen waarin gehandeld kan worden.
- De portfolio pagina bevat jouw fictieve aankopen van aandelen.

A. Vue Router

Maak gebruik van VueRouter om navigatie te voorzien in de Header component. Start met het maken van het routes.js bestand en definieer componenten die voor elke route moeten ingeladen worden. Voeg een router-view element toe op de juiste plaats in de applicatie en voorzie router-link's in de Header component.

B. Globale Filter

Voorzie een globale filter die een getal omzet in de currency notatie hiervan. Ga voor de functionaliteit zelf eens op zoek via jouw favoriete zoekmachine. Gebruik deze filter in de Header en op de homepagina om de funds te weergeven.

C. Vuex

Maak gebruik van Vuex om globaal voor de applicatie data bij te houden. Begin met het aanmaken van het store.js bestand en koppel vuex aan jouw applicatie. Voorzie 3 data properties in de store, namelijk **stocks**, **trades** en **funds. stocks** is een array van aandeel objecten en deze objecten bevatten een id, naam en prijs. **trades** is een array van aankoop objecten en deze objecten bevatten een id, hoeveelheid, aankoopprijs en stock id. **funds** is een nummer dat het saldo van de gebruiker voorstelt. Voorzie wat dummy data om te tonen als de applicatie inlaadt.

D. Home Component

De home component bevat een basis uitleg van de applicatie en een weergave van het huidige saldo van de gebruiker. Haal dit saldo op uit de Vuex.Store d.m.v. een Getter en pas de globale currency filter toe.

E. Stocks Component

De stocks component bevat een overzicht van de huidige te verkrijgen aandelen. Deze worden weergegeven als een lijst van componenten m.a.w. elk aandeel is een aparte component. Voorzie in deze component info over het aandeel, een input veld voor de hoeveelheid en een koop-knop (zie screenshot). Stocks data dient uit de Vuex.Store gehaald te worden d.m.v. Getters. Het klikken op de koop-knop dient de state van de Vuex.Store aan te passen d.m.v. Actions en Mutations.

F. Portfolio Component

De portfolio component bevat een overzicht van de huidige aankopen van de gebruiker. Deze worden weergegeven al een lijst van componenten m.a.w. elk aandeel is een aparte component. Voorzie in deze component info over de aankoop, een input veld voor de hoeveelheid en een verkoop-knop (zie screenshot). Gebruik dynamic styling om aan te tonen wanneer een aandeel verlies heeft (rood), gelijk is gebleven (oranje) of winst heeft gemaakt (groen). Trade data dient uit de Vuex.Store gehaald te worden d.m.v. Getters. Het klikken op de verkoop-knop dient de state van de Vuex.Store aan te passen d.m.v. Actions en Mutations. Als alle aandelen verkocht zijn moet het verwijderd worden uit het overzicht. Verschillende aankopen mogen apart weergegeven worden om de verlies/winst berekening niet te complex te maken.

G. Einde Dag Simuleren

Bij het klikken op de “End Day” link in de header worden alle aandelen (**stocks**) van prijs veranderd via een Action en Mutation. Bereken random een percentage tussen -10 en 10 voor elk aandeel en pas de prijs van het aandeel hiermee aan.

H. Data Opslaan

Bij het klikken op de “Save” link in de header stuur je alle state vanuit de Vuex.Store als een json string (JSON.Stringify) naar de “server” via een post request met VueResource. Gerbuik als server de meegegeven node/express api.

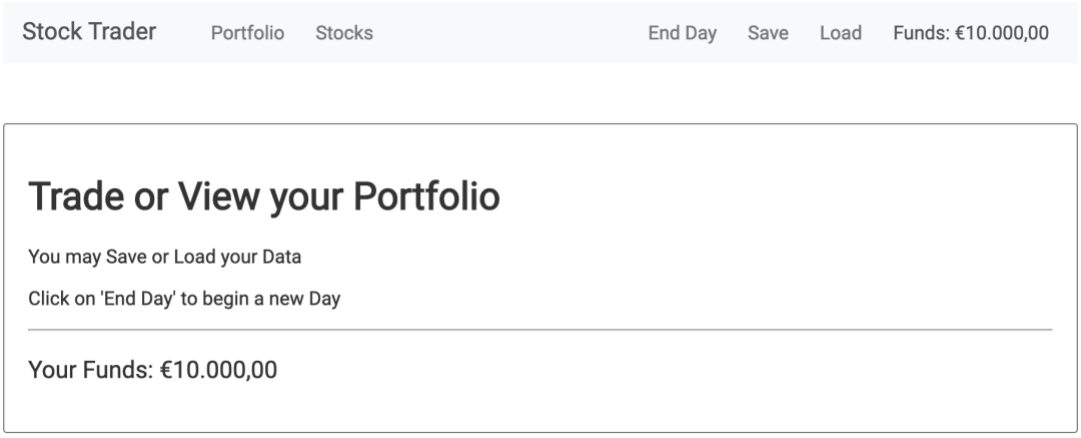
POST request: `http://localhost:3000/stocks`
Data: property stocks, value JSON string
Headers: `'Content-Type': 'application/json; charset=utf-8'`

I. Data Laden

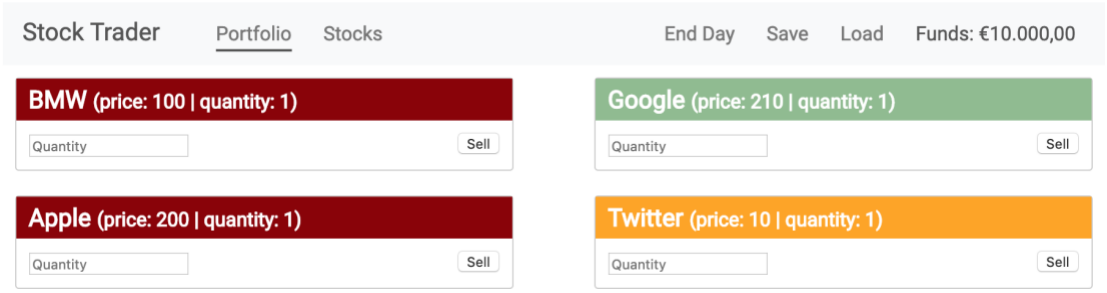
Bij het klikken op de “Load” link in de header haal je via een get request met VueResource alle state van de “server” via VueResource en pas je de huidige state aan naar de verkregen data.

GET request: `http://localhost:3000/stocks/last`

Homepagina



Portfolio



Stocks