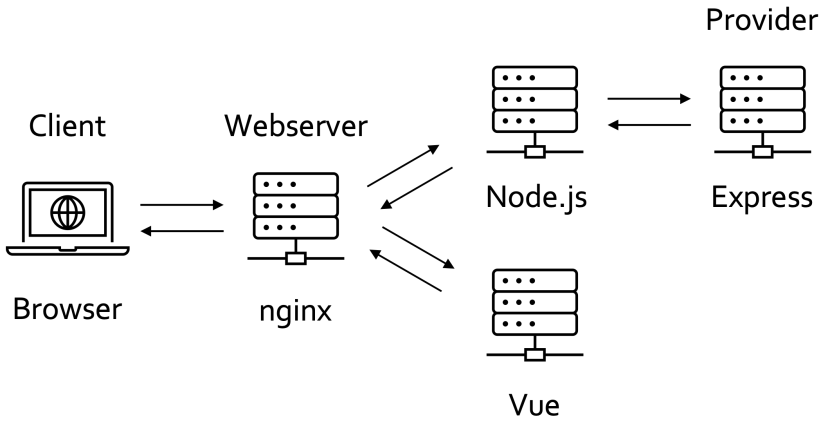
Übungsblatt 12: n-Tier Applikationen

Vorstellung in den Tutorien am 31.01. - 02.02.2022

Letzte Woche haben wir kennengelernt, wie man einen simplen Node.js Anwendungsserver implementiert und nginx als Reverse Proxy konfiguriert. Heute nutzen wir diese Technologien, um eine „Flugsuchmaschine“ zu entwickeln, die Nutzern eine Auswahl an Flügen verschiedener Fluganbieter (Provider) entsprechend ihrer Suchparameter zusammenstellt.

Die Architektur dieser Anwendung wird im nachstehenden Diagramm abgebildet:



Die Anwendung funktioniert wie folgt:

1. Um die Anwendung zu nutzen, verbinden sich Clients mit dem nginx Webserver, welcher auf dem HTTP-Standardport 80 läuft und mittels `proxy_pass` auf die Startseite des Live-Servers mit der Vue-Anwendung weiterleitet.
2. Durch Klick auf den Button „Get Flights“ wird eine GET Anfrage mit den gewählten Suchparametern an den nginx Webserver geschickt. Diese Anfrage wird mittels `proxy_pass` an den Node.js Anwendungsserver weitergeleitet.
3. Der Anwendungsserver extrahiert die Suchparameter aus der Anfrage und schickt entsprechende Anfragen an die in den Parametern spezifizierten Provider.
4. Die Antworten der Provider werden vom Anwendungsserver aggregiert und über den nginx Reverse Proxy an den Client zurückgegeben, woraufhin die Ergebnisse im Browser angezeigt werden.

12.1 Konfiguration

Ihnen sind in der Vorgabe bereits große Teile der beschriebenen Anwendung gegeben. Sie müssen allerdings einige Schritte befolgen, um die Anwendung richtig zu konfigurieren.

1. Um den nginx Webserver zu konfigurieren, ergänzen Sie die Konfiguration von letzter Woche um folgende Einträge und starten Sie anschließend nginx:

```
server {
  listen 80;

  location / {
   # Vue-based Frontend
    proxy_pass http://localhost:8080;
  }

  location /flights {
   # Node-based Backend
    proxy_pass http://localhost:3000;
  }
}
```

2. Auch die Provider sind bereits gegeben. Diese nutzen das Node-Modul express, das in der kommenden Vorlesung vorgestellt wird. Installieren Sie dieses Modul mit dem Befehl `npm install express`. Anschließend können Sie mit dem Befehl `node providers.js` die Provider ausführen. Der so gestartete Server ist unter der URL `http://localhost:3001` erreichbar.

   Es gibt drei Fluganbieter, die jeweils unter einem eigenen Pfad erreichbar sind:

   - SqueezyJet - `http://localhost:3001/squeezyjet/flights`
   - BrianAir - `http://localhost:3001/brianair/flights`
   - AxelAirways - `http://localhost:3001/axelairways/flights`

12.2 Node-Server

Implementieren Sie den Node.js Anwendungsserver der oben beschriebenen Flugsuchmaschine! Ergänzen Sie dazu die Datei *server.js*.

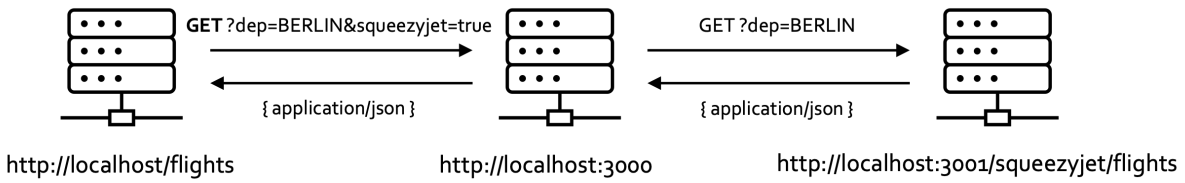
Der Anwendungsserver soll GET Anfragen vom Client verarbeiten können. Diese haben folgende Parameter:

Parameter	Typ	Beschreibung
`dep`	string	Startflughafen
`squeezyjet*`	boolean	Interesse an SqueezyJet-Flügen
`brianair*`	boolean	Interesse an BrianAir-Flügen
`axelairways*`	boolean	Interesse an AxelAirways-Flügen

*Besteht kein Interesse an einem Fluganbieter, kann der jeweilige Parameter weggelassen werden (äquivalent zu false)

1. Für jeden angegebenen Fluganbieter soll eine GET Anfrage an den entsprechenden Provider geschickt werden. Als Parameter wird der angegebene Startflughafen mitgereicht.
2. Die Antworten aller Provider sollen zu einem Array aggregiert werden. Der Einfachheit halber bietet jeder Fluganbieter nur einen Flug pro Startflughafen an.
3. Nachdem alle Antworten eingetroffen sind, soll das aggregierte Array als Response an den Client geschickt werden.

Das folgende Diagramm veranschaulicht den Nachrichtenfluss für eine GET-Anfrage:



Hinweis: Die von den Providern zurückgegeben Fluginformationen befinden sich natürlich im Response Body, müssen also aus Chunks zusammengefügt werden!

12.3 Vue-Client

Nun fehlt nur noch das Frontend. Ihnen ist im Ordner */vue* der Vorgabe eine Vue-Anwendung gegeben, welche Sie mit dem Befehl `npm run serve` ausführen können. Bisher passiert allerdings noch nichts, wenn man auf den *Get Flights*-Button klickt.

Ergänzen Sie in der Komponente `SearchForm` die Methode `getFlights()`, indem Sie mittels der *fetch*-API eine GET-Anfrage an den zuvor implementieren Node-Server senden. Reichen Sie das in der Antwort enthaltene Array von passenden Flügen an die Elternkomponente weiter. Ergänzen Sie dabei die `App`, damit diese das Array auch in Empfang nehmen kann.